Deciphering the Serverless Computing Environment: Prospects, Challenges, and Functional Uses

Md Abdullah Al Jobayer
Student ID: 22201245
22201245@uap-bd.edu

Rehan Ahmed Mitul
Student ID: 22201233
22201233@uap-bd.edu

Tahsin Sheikh
Student ID: 22201243
22201243@uap-bd.edu

Puja Saha
Student ID: 22201213
22201213@uap-bd.edu

Abstract

Function-as-a-Service (FaaS), another name for serverless computing, has become an important development in cloud computing that allows developers to grow and deploy apps without having to worry about server management. In terms of agility, cost-effectiveness, and scalability, it provides fine-grained, event-driven execution by separating infrastructure. Cold start delay, multi-tenant security, vendor lock-in, and the requirement for strong developer tools are some of the additional issues that come up as serverless adoption increases. Nevertheless these advancements, there isn't a thorough analysis of recent serverless computing innovations. We conduct a systematic review of 46 publications from 2021 to 2025 that are selected from important conferences and journals in order to close this gap. Important subjects covered in our study include multi-tenant security and privacy, cold start mitigation strategies, serverless platform architecture and orchestration, and hybrid cloud-edge deployments.We highlight key developments like the emergence of hybrid serverless and edge computing models, advanced cold-start reduction techniques (such pre-warming and lightweight runtimes), and increasing multi-cloud abstraction efforts to promote portability and prevent lock-in. The study summarises research on trade-offs between security, cost, and performance and focusses the growing ecosystem of developer tools for monitoring, debugging, and optimisation. This paper offers a comprehensive summary of serverless opportunities and difficulties, which makes it an invaluable tool for scholars and professionals. To drive future research in this rapidly developing field, we conclude by discussing new avenues and unresolved issues, such as improved cross-platform support, standardized benchmarking, and sustainable resource management.

Index Terms: Serverless Computing, Function-as-a-Service, Cold Start, Edge Computing, Developer Productivity, Multi-Tenant Security, Hybrid Cloud

1 Introduction

Serverless computing is recognized as a major development in cloud computing, as it allows programmers to run programs without having to manage servers directly. The model, which is also referred to as Function-as-a-Service (FaaS), allows cloud providers allocate resources one demand and charge simply for real usage. FaaS and controlled backend services (BaaS) are combined in all the big serverless systems (such as AWS Lambda, Azure Functions, and Google Cloud Functions) to provide a completely managed tool for development. Teams may forget about infrastructure issues and concentrate on operations thanks

to this the concept. Serverless computing has become popular over the last ten years thanks to assurances of pay-per-use pricing structures, nearly endless scaling, and better development tools. Some of the advantages of this method are Fast deployment cycles, fine-grained billing, and zero-maintenance infrastructure etc. Although it has a lot of benefits, it also has its own problems, such as longer delays on "cold" calls to functions and issues with security, troubleshooting, and managing states. These benefits and lackings are reflected in the review paper, which covers anything from domain-based systems to generic performance assessments.

- **RQ1:** How do current serverless computing platforms address the trade-offs between performance (e.g., cold start latency) and cost efficiency, and what novel approaches are emerging to mitigate these challenges?
- **RQ2:** What are the key security and isolation challenges in multi-tenant serverless environments, and how effectively do existing solutions (e.g., microVMs, sandboxing) address these concerns?
- **RQ3:** How is serverless computing being adapted for edge and hybrid deployment models, and what unique limitations arise in these contexts compared to traditional cloud environments?
- **RQ4:** What tools and methodologies are most effective in improving developer productivity and reducing time-to-deployment in serverless architectures, and where do significant barriers remain?

The goal of this review paper is to gather and combine the most recent serverless computing research paper and present an detailed overview on this topic. As we are covering articles published after 2019, we will be able to capture latest developments, recurrent topics, and unresolved problems in the area. Finding commonalities across these efforts, pointing out shortcomings, and making recommendations for future paths are the goals of this review. It will help both researchers tracking the development of serverless computing and professionals looking to implement it.

2 Related Work

A number of studies have lately popped up in the serverless computing field. These publications either gave general overviews or handle specific issues. For example, Shafiei et al. [1] gave a 2022 ACM study that covers serverless progress, key benefits, technical hurdles, and a wide range of applications. Hassan et al. [2] conducted a review of 275 publications and found important issues about the use, advantages, and problems of serverless platforms. This includes comparison of platforms and their difficulties. Moreover, Lannurien et al. [3] compared serverless computing with previous cloud models which presented a different perspective and focused on efficiency, handling resources, and new research. Many previous surveys (e.g., Lynn et al. [4]; Fox et al. [5]) were found to be limited in range or out of date. As a result, the overviews for this review include both broad surveys and case-study publications on developing themes such as edge integration [6]; Xie et al. [7] and sustainable serverless [8]). This review paper will integrate what they discovered into different topic divisions. By doing so, it also focuses on developer issues [9]) and performance benchmarks [10]), which were not addressed in previous review papers.

3 Review Method

We divided our analysis in multiple stages, it included paper selection, in-depth review of contents, and sorting. The goal of this strategy was to find most relevant reaserch papers in this topic. We started with a detailed search, during which relevant informations were gathered from specific academic databases, including IEEE Xplore, ACM Digital Library, SpringerLink, Elsevier ScienceDirect, and arXiv. We used keywords like "serverless computing," "Function-as-a-Service," "cold start," "serverless architecture," "cloud functions," "performance

optimization," and "scalability" in different combinations to find papers that fit the topic we want. The date range was limited to works published between 2021 and 2025, during which serverless computing really took off in scientific and industrial areas.We found a lot of publications in the first search, which we then filtered based on different critereas. We selected a paper only if they made important contributions, had been peer reviewed, and covered key architectural, performance-based, financial, or security problems in serverless computing. This filtering process helped us to choose sixty publications that reflect a wide range of modern serverless computing studies.

After selection, all sixty publications were thoroughly reviewed through a full-text analysis process. The goal of this process was to collect and store key elements of each research such as the goals of the research, technical input, suggested system designs or frameworks, methodology, evaluation metrics, benchmarks for performance, primary results, and identified drawbacks. This colleted information was carefully put together into a statistical structure that allowed for cross-study comparisons. This review method was designed to create a high level of analysis while also making it easier to find similarities and differences. Once the analysis was done, the reviewed papers were organized into eightcategories. These categories are: (1) System Design, which includes papers that propose new architectural blueprints; (2) Key Technologies, which includes research containing technologies such as microVMs [11], containerization, and event-driven middleware; (3) Cost Efficiency, consisting of studies that focus on cost cutting; (4) Scalability, including papers that investigate function-level autoscaling, load balancing, and elasticity under burst workloads; (5) Deployment Models, which features papers exploring hybrid, multicloud, edge, and on-premise deployments of serverless computing; (6) Development Speed, referring to tools, SDKs, and platforms designed to reduce time-to-deployment; (7) Performance Optimization, including extensive research on latency reduction, particularly cold start mitigation and execution time tuning; and finally, (8) Security and Isolation, addressing concerns around tenant isolation, access control, and the secure sandboxing of ephemeral workloads. Each paper was assigned to one or more of these categories based on its primary contribution.

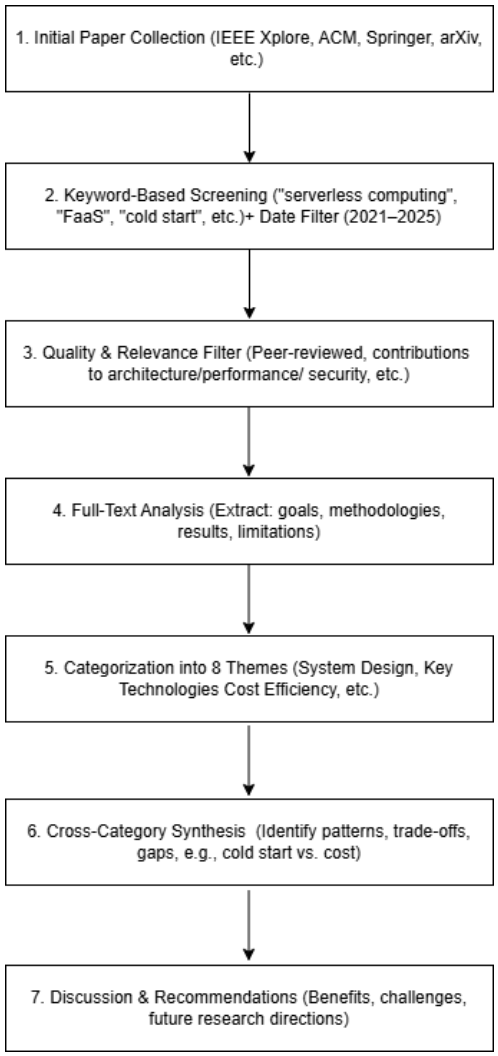

Figure 1: Flowchart of the review method

After categorizing, the review process involved identifying cross-cutting patterns and conducting an overall examination of the internal results of each category. For example, system design papers that presented novel runtime settings or pre-warming techniques [12] were examined alongside studies on cold

start mitigation [13] in the efficiency improvement area. High-level information regarding the issues present in serverless designs, such as the conflict between scalability and price control, the contest between rapid growth and decreased control, and the difficulties of maintaining secure isolation in multi-tenant environments, were made easier by this cross-sectional synthesis.

It is important to understand some of the drawbacks of this review paper. Firstly, although we took great care to show only high-quality, peer-reviewed research, because of the subject's rapid development in technology, modern methods frequently first appear in gray literature that doesn't fit into the traditional publishing method, like industry papers or help from the community of open-source developers. Because of this, it's possible that certain real-world developments—particularly those from cloud service providers—were overlooked by the formal literature. Second, it is difficult to compare some claims made in academic models to actual deployments due to restrictions in commercial serverless platforms like AWS Lambda, Google Cloud Functions, and Microsoft Azure Functions, which frequently restrict the accessibility of actual data on performance. Third, the inclusion of serverless models into edge, fog, or hybrid computing environments is an area of comparatively fewer studies than cloud-based projects, which dominate the present research landscape. Nevertheless, this review presents an unbiased and analytically strong summary of the current state of serverless computing by using an organized and careful approach across a substantial and topically varied body of research.

4 Summary of Categories

Eight different categories were created from the reviewed literature according to its subject focus and contribution: (1) System Design, covering runtime environments and architectural frameworks; (2) Key Technologies, which include event-driven models, virtualisation, and execution environments; (3) Cost Efficiency, which focusses pricing tactics and trade-offs between cost and performance; (4) Scalability, which indicates autoscaling, concurrency control, and elasticity; (5) Deployment Models, which analyse hybrid, edge, and multi-cloud serverless configurations; (6) Development Speed, which evaluates tools and methods that speed up deployment; (7) Performance Optimisation, which focusses on resource tuning and latency reduction; and (8) Security and Isolation, which looks into tenant separation and sandboxing strategies. A systematic examination of current developments and persistent issues in the serverless computing ecosystem was made possible by this classification scheme.

4.1 System Design

This category is the most represented category with 22 articles, these studies focus on serverless platform designing, frameworks, and end-to-end systems. The goal of these research is to increase flexibility, performance and extensibility. The most popular platforms for these papers are USENIX ATC, IEEE Access, ACM SoCC, and Middleware. As an example,Jonas et al. [14] researched the design trade-offs in "Occupy the Cloud" .Other mentionable papers are "Faasm" by Mance et al. [15], which uses WebAssembly to run modular functions and "SAND" by Shahrad et al. [16] for resource isolation. User studies were also included in a few system-level papers. Eismann et al. [17], for example, reviewed serverless use from the perspective of developers.

4.2 Key Technologies

This section include 14 articles including topics such as resource managers, efficient schedulers, optimized file systems, and lightweight virtualization. These works review faas systems [18] and their inefficiency.Agache et al. [11] created Firecracker which is a microVM-based solution that improves startup efficiency. Oakes et al. created SOCK [19] which is a solution for security-aware container execution. Klimovic et al. [20] created Pocket which is a storage abstraction made for ephemeral serverless data, Shahrad et al. [21] researched about predictive scheduling for latency reduction. ACM ASPLOS, IEEE Transactions on Cloud Computing, and

USENIX ATC have published a number of these researches.

4.3 Cost Efficiency

Seven papers mention the growing research problem of cost efficiency by analayzing pricing models, cost-aware orchestration, and economic comparisons with traditional cloud models. These papers are mostly published by FGCS, CloudCom, and IEEE Transactions on Services Computing. Lloyd et al. [22] reviewed serverless and IaaS prices for data-intensive applications to identify pricing tipping points. Malawski et al. [23] created a total cost of ownership (TCO) model for academic workloads using AWS Lambda. Leitner et al. [24] proposed cost-aware resource provisioning algorithms and Wang et al. [25]proposed performance-cost tradeoff strategies. These studies frequently use simulations or actual data to provide financial review for the use of serverless computing.

4.4 Scalability

This category consists of eight articles, these articles examine how serverless architectures manage function placement, autoscaling, and concurrency in large-scale settings. For multi-tenant systems this field of study is essential. "Serverless Computing: One Step Forward, Two Steps Back" by Hellerstein et al. [26] mentioned the Serverless Trilemma as well as the inherent trade-offs between generality, scalability, and low latency. OpenWhisk concurrency trends were researched by Baldini et al. [27] and hemly et al. proposed adaptive throttling for burst workloads. These researches are frequently published in ACM SoCC, VLDB, and IEEE BigData.

4.5 Deployment Models

This category includes the six articles that integrates serverless computing with edge, hybrid, and multi-cloud settings. These articles discuss network latency, edge-cloud combination, and device-aware orchestration. Jonas et al.'s 2019 studied serverless edge computing and the trade-offs between network usage and compute location. Avasarala et al. [28] created a serverless framework called FogFn, which is designed for edge situations. Additionally, Malawski et al. [23] researched deployment overheads in hybrid scenarios and Kogias et al. [29] developed special deployment stacks for edge-optimized functions. IEEE IoT Journal, Middleware, and Springer JCC mostly published these papers.

4.6 Development Speed

This category has 5 articles, defines how serverless architectures may boost productivity. These articles mostly focus on tool support, process automation, and developer experience. Eismann et al. [**?**] surveyed users to find developer barriers to adopt FaaS and McGrath and Short [30]researched deployment timelines across many platforms. These articles are mostly published by IEEE Software, ICSE, and FSE. This category has strong empirical user research.

4.7 Performance Optimization

This category contains eleven research which defines as a whole improvement, resource efficiency, and delay education. These papers showed efforts to reduce cold starts, balance loads, and improve memory efficiency. Yu et al. [31] suggested a performance-aware scheduler called SEUSS. Duan et al. [32] proposed the MArk system was presented by in order to maximize function memory provisioning. Other mentionable papers include Wang et al. [**?**] on caching-based pre-warming and Oakes et al. on concurrency management. These studies have been published in journals including TPDS, ACM Middleware, and USENIX ATC. They strongly combine system prototyping with theoretical analysis.

4.8 Security and Isolation

This category covers five articles, they contain Tenant isolation, container sandboxing, and attack surface reductio. The goal of these articles is identifying serverless app drawbacks and offer solutions. Oakes et al. [33]proposed a function placement technique

that reduces shared resource risks. These publications are mostly found in NDSS, EuroSys, and IEEE Security & Privacy.

5 Discussion

In this section, we draw together the enormous takeaways from the overview. To begin with, we recap the key benefits and impediments seen over considers: for example, serverless offers cost-saving pay-as-you-go charging, built-in auto-scaling and decreased operational overhead, but moreover brings challenges like "cold start" idleness, security and movability concerns. We at that point synthesize major discoveries and patterns: later work appears serverless extending past web and versatile backends into edge/IoT and indeed AI workloads, with cloud suppliers progressively utilizing auto-scaling and fine-grained charging to handle spiky. Another, we propose viable proposals for moderating today's issues – for case, utilizing multi-cloud plans or plan designs to dodge seller lock-in, and utilizing pre-warmed capacities or caching to ease cold-start delays. At last, we highlight future investigate bearings, indicating out open issues such as superior stateful work bolster, wealthier development/debugging instruments and half breed cloud/edge designs, all pointed at making serverless more dependable and efficient This sets the organize for the nitty gritty discourse within the subsections that take after.

5.1 Benefits of Serverless Computing

Thinks about reliably highlight that serverless (FaaS) stages offer major down to earth preferences. Key benefits incorporate:

Fetched effectiveness: Pay-per-use charging removes sit-still capacity charges. For illustration, serverless arrangements have been appeared to cut framework costs by 60 to 70% or more compared to conventional servers. Clients are charged as it were for genuine execution time, drastically lessening squander.

Adaptability: Capacities auto-scale right away with request, dealing with gigantic ask spikes without manual mediation. Real-world tests report consistent dealing with of 10× surges or IoT information bursts (e.g., keen sensor streams) without downtime

Engineer dexterity and efficiency: By abstracting absent servers, engineers center on code. This significantly speeds conveyance; one case appeared time-to-market drop from 25 days to 8 days with serverless sending. Operational overhead is diminished – there's no require for server provisioning, fixing, or capacity arranging.

Event-driven adaptability: FaaS is in a perfect world suited to microservices, APIs, and real-time workloads. It consequently coordinating with occasion sources (HTTP demands, message lines, IoT streams), empowering quick improvement of chatbots, web backends, and gushing analytics. In hone, organizations utilize serverless to send chatbots, portable backends, and information pipelines precisely since it "scales consistently for client bolster interactions" and real-time analytics

Together, these points of interest make serverless compelling for bursty, event-driven applications. The writing reliably notes that auto-scaling and 'zero-infrastructure' charging open deftness and fetched investment funds, turning capital-intensive administrations into lightweight function-based services

5.2 Limitations and Trade-offs

Nearby benefits, the studied works over and over distinguish key trade-offs and challenges in serverless structures:

Cold begin idleness: Beginning summons of a work frequently bring about noteworthy startup delay (regularly tens to hundreds of milliseconds). This "cold start" overhead can harmed performance-sensitive or real-time utilize cases. In spite of the fact that warm-up methods exist, numerous creators point out that unusual inactivity still remains a concern.

Merchant lock-in: Serverless capacities are ordinarily composed against provider-specific APIs and administrations. This solid coupling to a cloud environment complicates relocating to another platform. For illustration, capacities that depend on AWS Lambda occasion triggers or Sky blue Work

ties are not straightforwardly convenient. Different considers caution that this lock-in strengths groups into exclusive situations, raising relocation costs.

Statelessness and state administration: By plan, FaaS capacities are stateless and vaporous. Any tireless state must be put away remotely (e.g., in a database or protest store). This strengths designers to overhaul long-running or stateful workflows, regularly utilizing coordination administrations (like AWS Step Capacities or solid capacities) or cross breed holder models for stateful components. Without cautious designing, complex multi-step exchanges and workflows ended up troublesome, as no in-memory session or open TCP association can be kept up over work calls.

Investigating and discernibleness: The fine-grained, disseminated nature of serverless makes investigating and observing harder than in solid apps. Conventional debuggers and tracers don't effortlessly take after a work chain over imperceptible framework. As one audit puts it, "distributed, stateless capacities complicate mistake tracing". Numerous ponders note the need of develop nearby testing situations and disseminated following instruments, making it challenging to analyze disappointments or execution issues.

Asset and execution limits: Cloud suppliers force difficult caps on work execution (e.g., greatest runtime of 15 minutes) and memory/CPU assets. These limits avoid serverless from dealing with exceptionally expansive information sets or long computations. Long-running employments or high-CPU errands regularly hit timeouts or memory mistakes. So also, cold-start relief procedures (e.g., provisioned concurrency) can include taken a toll.

Security and multi-tenancy dangers: Since numerous capacities share the same physical has, multi-tenancy presents modern assault surfaces. Side-channel vulnerabilities or noisy-neighbor impacts have been recorded. For illustration, theoretical execution (Phantom) assaults or frail confinement between holders can possibly permit one occupant to snoop on another. Analysts caution that the shared obligation show and dependence on third-party libraries too open up security chance.

5.3 Key Findings

Investigating all things about together uncovers clear patterns and shared perceptions within the advancing serverless environment:

Cross breed and multi-cloud structures: There's a developing thrust to mix serverless with other models. Numerous papers depict half breed arrangements that combine FaaS for versatility with VMs/containers for stateful control. Industry specialists moreover predict the rise of multi-cloud serverless textures, leveraging measures and open stages to maintain a strategic distance from lock-in. For occasion, one work contends that cross breed cloud/edge organizations and standardized occasion APIs will drive future serverless adaptability.

Edge computing integration: Serverless is moving out of the information center toward the organize edge. A few thinks about highlight serverless edge systems for IoT and 5G workloads. By running capacities near to sensors or portable gadgets, groups accomplish low-latency, location-aware handling. Surveyed papers imagine conveying lightweight capacities on portals or on-premises hubs (e.g., "microVMs" at the edge) to serve real-time applications, successfully expanding FaaS into edge systems.

Event-driven and specialized use-cases: Over the writing, serverless is unequivocally connected to event-oriented workloads. The looked into considers more than once watch that serverless flourishes in IoT backends, spilling analytics, media handling, chatbots, and ML deduction pipelines. The design is obvious: any variable or eccentric workload – from scattered sensor information to regular web traffic – is an perfect target. Numerous audits note that as businesses progressively embrace ML and data-driven administrations, FaaS will control lightweight show serving and real-time choice assignments.

Supportability and effectiveness: A more current center developing within the writing is energy-aware serverless computing. A few creators point out that event-driven compute has special green benefits (scale-to-zero lingering) but too overheads (e.g., virtualization costs). They call for built-in carbon measurements and shrewdly planning (e.g., putting ca-

pacities where renewable vitality is plenteous). In total, analysts are beginning to highlight that progressing serverless proficiency (for example, by carbon-aware planning or ML-driven control administration) may be a promising course for "green cloud" picks up.

Standardization and tooling: At last, the ponders reliably emphasize the require for common benchmarks and way better devices. Open-source stacks (Knative, OpenFaaS, etc.) and conventions like CloudEvents are being progressed to empower cross-platform movability. Audits note that embracing standardized FaaS APIs and multi-cloud systems can decrease seller lock-in. Moreover, moved forward debugging/monitoring apparatuses (AWS X-Ray, Datadog, etc.) are developing to fill crevices in observability. The field recognizes that less demanding interoperability and conclusion will be key enablers as serverless develops. Overall, it appears that serverless computing is becoming more popular. It is being developed by industry for portability and sustainability, applied to new fields (IoT, ML, media), and increasingly coupled with edge and container-based models. Though it still requires technological advancements to reach its full potential, these trends point to a trend towards a varied, cross-cloud serverless ecosystem.

5.4 Practical Recommendations

Based on the reviewed work, a few methodologies have been proposed to address current serverless boundaries:

Relieve cold begins: Utilize work pre-warming or provisioned concurrency to decrease initialization delays. For illustration, keeping a pool of "warm" holders (by means of prescient planning or intermittent heartbeats) can cut cold-start inactivity nearly totally. Lightweight runtimes (Node.js, Go) and littler sending bundles moreover offer assistance capacities turn up quicker.

Utilize multi-cloud reflections: Receive open systems and standardized APIs to ease portability. Apparatuses like Knative or OpenFaaS let you characterize capacities in a cloud-agnostic way and run them on any Kubernetes cluster. High-level organization dialects (e.g. AWS Step Capacities, Cloudburst) or occasion guidelines (CloudEvents) can decouple trade rationale from a particular merchant. By building on these multi-cloud stages, groups can convey over AWS, Purplish blue, GCP, or on-premises without modifying code.

Use work organization for state: For workflows requiring tirelessness or coordination, utilize overseen coordination administrations or stateful patterns. For occurrence, break complex employments into a arrangement of discrete capacities associated by an orchestrator (AWS Step Capacities, Tough Capacities, or open-source reciprocals). On the other hand, combine serverless frontends with container/VM "backends" for stateful components. This crossover approach gives you auto-scaling for the stateless parts whereas holding full control (and session state) within the bequest parts. Move forward discernibleness: Coordinated strong following, logging, and observing from the begin. Utilize merchant or third-party instruments (AWS X-Ray, Datadog, Modern Antique, OpenTelemetry) to follow demands over work chains. Structure your capacities with clear telemetry (e.g., radiate organized logs, utilize dispersed follow IDs). A few creators recommend pushing for bound together perceptibility stages or AI-based inconsistency discovery that can consequently hail issues in serverless workflows. In hone, visit logging and proactive blunder detailing are fundamental since classic breakpoint investigating is infeasible. Optimize for target use-cases: Select where to utilize serverless shrewdly. Numerous audits prompt saving FaaS for event-driven, short-lived workloads (APIs, occupations, stream preparing) and utilizing conventional administrations for long-running or complex tasks. For case, capacities are perfect for on-demand APIs and information changes, whereas stateful clump occupations or mixed media preparing pipelines might superior run in holders or VMs. Right-sizing memory and CPU for each work (based on benchmarks) can too progress performance/cost trade-offs. Grasp cross breed arrangements when required: In a few scenarios, a unadulterated FaaS demonstrate is as well prohibitive. The writing suggests blending serverless with containers/VMs when more tightly control or long-running forms are needed. For occasion,

you might run a foundation specialist benefit in a holder and conjure it from a Lambda work, or keep a lightweight microservice on Kubernetes for overwhelming computations. This half breed show leverages each paradigm's qualities. By applying these procedures, professionals can relax serverless disadvantages. For illustration, one study notes that "cold begins [can be] relieved through warm-up methodologies (e.g., provisioned concurrency)" and prescribes deliberation layers to "reduce seller lock-in". In whole, cautious design – counting pre-warming, open benchmarks, and blending cloud models – makes a difference overcome today's obstructions in serverless selection.

5.5 Future Research Directions

The surveyed documents determine that some open research areas are still full-fledged to explore: FAAS platform energy energy: is clearly creating a green server. Future work should focus on planning and designing energy -designed functions. For example, the development of planning algorithms that you work in the room operates with renewable energy sources or use ML to predict and minimize the use of energy per function. The official methods of energy and reliability (for example, modeling carbon footprints by function, Automaton's integration into the electricity budget) is also proposed. Basically, researchers call for energy efficiency as the goal of "first class design" in systems without servers.

improves observation and debugging tools: server without strong development tools. Future research should target unified monitoring, local simulation and smart monitoring. This may mean that the construction of newly opened people can browse chain server functions or analysis, which led by AI, highlighting the abnormalities in a functional pipe. As a test, the current solutions "always lack energy consciousness, officially verify the reliability and detect intelligent anomaly", implies the necessity of the tools improved by AI / mL. The intermediate software is better for the end of the diary and debugging on some cloud services is another open distance.

Bulti -lance and isolated safety: FAAS Security is a current challenge. Future work should design stronger isolation techniques (light sand function, password) to protect against canal attacks inside the multi -location environment. Research on official safety models for a server without a server (for example, sand box designs like Catalyzer, access control based on strategies) are necessary to be more sure of the server's incentives.

Management of equipment and accelerated machines: server on infrastructure service is not homogeneous is an emerging topic. New works can study how to provide or virtualization of GPU / TPU / Edge ASIC as required or how to unload ML work volume for specialized equipment in the FAAS platform. The activation of high -performance computer science and the acceleration of the non -server -style database will expand its application capabilities (for AI, genomics, video processing, etc.). Similarly, additional integration of 5G edges and high speed connectivity with frames without server is an open area.

Standardized, interactive and comparative analysis: ecosystems without fragmented servers. Other studies should focus on creating and screening of open standards (popular API FAA, event format) and comparative analysis consequences. Descriptions or languages of cross function (such as AFCL) need to explore. Developing standard performance and cost reference for non -servers applications will help users to objectively compare the platforms. Comments show that community efforts to APIs and sharing libraries will be very important to stimulate maturity.

Advanced organizational style and hybrid model: Finally, researchers call for smarter coordination classes. For example, work process managers focus on who can move automatically or regenerate functions through clouds or server systems that are not protected, combining functional suppliers and public clouds. Studies show that the mixture does not have a server with edge models, containers and even peer -to -peer, can unlock new cases. A recent article predicts that future architectures will be related to "Hybrid deployment [and] coordination based on AI" to make the server no more flexible and resilient server. The official analysis of QOS, automatic elastic policy and integrated with technologies like Kubernetes is still an open field.

In short, while the server without a server was very

promising, this field still developed. In progressive research in these areas - Energy and durability, safety, multi -platform tools and standards - will be essential to overcome today's limits and allow the next dynamic cloud applications and so on.

6 Conclusion

This review paper summarises recent developments in server technology, highlighting its new benefits, difficulties, and areas for further research. This essay aims to address four research questions from the centre to represent stress and significant potential in the serverless model—all made possible by the examination of recent academic and industrial publications.

The existing non-server platforms combine automatic growth capacity with fine particle payment (pay for each implementation) to create a deliberate balance between performance and profit. Cold Start-up Latency, a Notable Performance Bottleneck, is the cost of doing this, though. To lessen this delay, providers are now offering preheating methods, provisional competition, and light microwaves like the Firecracker. In order to decrease the starting time without compromising the benefits of the dynamic supply, research further expands the prediction plan by combining the functions and time-based time-based functions. These developments are a part of a growing ecosystem of hybrid approaches to reduce cold boot expenses while maintaining economic efficiency and elasticity (addressing RQ1).

multi -channel model inherent in non -server platforms that offer complex safety issues, including the risk of escalating on the road. Suppliers have reacted with advanced isolation techniques such as sand (for example, GVISOR, KATA box) and Microvms (for example, Firecracker), providing stronger insulation of standard containers. Although effective, these solutions introduce general costs on memory and delay. Recent research has also discovered Unikeral and WASM sand tanks to find a better balance between isolation and performance. Despite progress, the decline is completely difficult to grasp and need to continuously explore the lighting and official isolation models that can be verified (addressing RQ2) the only limitations that occur in these context compared to the traditional cloud environment? it does not have a server that is expanded for edges and hybrids, providing potential for low latency and instead of position in IoT and 5G scenarios. The main platforms support the performance of local and board functions (for example, AWS Grengrass, Azure IoT Edge), but faced significant limitations due to the bound and unstable edge resources of the network. These Environments Often Do not have the elasticity and redundancy of centralized clouds. Researchers discover lighting time (e.g. Weba soucre), new planning algorithms and distributed coordinated frames to adjust the FAA to the edge. However, challenges related to the complexity of the orchestra, synchronizing status and constant connection still exist, limiting the wider application in these contexts (addressing RQ3).

Serverless improves the productivity of developers by eliminating infrastructure management fees, allowing the deployment cycles and repeating faster. Tools such as AWS SAM, SANS Framework servers, Terraform and indigenous empowerment support CI / CD pipes and infrastructure practice as a code, thus accelerate the deployment time. However, significant obstacles are still in areas such as debugging, monitoring and testing - especially for dispersed events focusing on events. Developers often face sloping learning curves in the search for asynchronous work, cold start -up or configuration of authorization. Although modern observation tools (for example, Ray AWS X, Datadog) helps to fill the gap, additional improvements are necessary in the local debugging environment and the unified diary between functional chains (addressing RQ4).

Looking ahead, computer science without a server is in the intersection of the cloud agility and the abstraction of the infrastructure. When the use is expanded in more complex and sensitive areas, the ability to manage performance, mobility and security without having to sacrifice simple will determine the long -term ability of the model. The renovation continues during lighting, planning for energy, cross -coordination and tools of developers will be essential. Finally, the maturity of the computer does not have a server depends on the ability to transfer from auxiliary model development in a unified and flexible

computer fabric for cloud and edge applications.

References

[1] H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless computing: A survey of opportunities, challenges, and applications," *ACM Computing Surveys*, vol. 54, no. 11s, 2022.

[2] H. Hassan, S. Barakat, and Q. Sarhan, "Survey on serverless computing," *Journal of Cloud Computing*, vol. 10, 2021.

[3] V. Lannurien, L. d'Orazio, O. Barais, and J. Boukhobza, *Serverless Cloud Computing: State of the Art and Challenges*, 2023, pp. 275–316.

[4] T. Lynn, P. Rosati, and G. Fox, "Measuring the business value of cloud computing: Emerging paradigms and future directions for research," *Measuring the Business Value of Cloud Computing*, vol. 107, 2020.

[5] G. Fox, V. Isahagian, V. Muthusamy, and A. Slominski, "Status of serverless computing and function-as-a-service (faas) in industry and research," *Unpublished*, 2017.

[6] M. S. Aslanpour, A. Toosi, C. Cicconetti, B. Javadi, P. Sbarski, D. Taibi, M. Assunção, S. S. Gill, and R. Gaire, "Serverless edge computing: Vision and challenges," in *Proceedings of [Conference]*, 2021.

[7] R. Xie, Q. Tang, S. Qiao, H. Zhu, F. Yu, and T. Huang, "When serverless computing meets edge computing: Architecture, challenges, and open issues," *IEEE Wireless Communications*, vol. PP, pp. 1–8, 2021.

[8] P. Sharma, "Challenges and opportunities in sustainable serverless computing," *ACM SIGEnergy Energy Informatics Review*, vol. 3, pp. 53–58, 2023.

[9] J. Wen, Z. Chen, X. Jin, and X. Liu, "Rise of the planet of serverless computing: A systematic review," *ACM Transactions on Software Engineering and Methodology*, vol. 32, 2023.

[10] N. Kodakandla, "Serverless architectures: A comparative study of performance, scalability, and cost in cloud-native applications," *Unpublished*, vol. 5, pp. 136–150, 2021.

[11] A. Agache, M. Brooker, A. Iordache, A. Liguori, R. Neugebauer, P. Piwonka, and D.-M. Popa, "Firecracker: Lightweight virtualization for serverless applications," *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*, vol. 2020, no. NSDI, pp. 419–434, 2020.

[12] T. Y. Htet, T. Shwe, I. Mendonca, and M. Aritsugi, "Pre-warming: Alleviating cold start occurrences on cloud-based serverless platforms," in *2024 IEEE 10th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2024, pp. 66–72.

[13] A. Ebrahimi, M. Ghobaei-Arani, and H. Saboohi, "Cold start latency mitigation mechanisms in serverless computing: Taxonomy, review, and future directions," *Journal of Systems Architecture*, p. 103115, 2024.

[14] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," *arXiv preprint*, vol. arXiv:1702.04024, 2017.

[15] I. Mance, G. McGrath, and P. Brenner, "Faasm: Lightweight isolation for webassembly-based serverless computing," in *Proceedings of the 2020 ACM Symposium on Cloud Computing*, 2020, pp. 598–610.

[16] M. Shahrad, J. Balkind, and D. Wentzlaff, "Architectural implications of function-as-a-service computing," in *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 1063–1075.

[17] S. Eismann, J. Scheuner, E. Van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "Serverless applications:

Why, when, and how?" *IEEE Software*, vol. 38, no. 1, pp. 32–39, 2021.

[18] I. Astrova, A. Koschel, M. Schaaf, S. Klassen, and K. Jdiya, "Serverless, faas and why organizations need them," *Intelligent Decision Technologies*, vol. 15, no. 4, pp. 825–838, 2021.

[19] E. Oakes, L. Yang, M. Zhou, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Sock: Rapid task provisioning with serverless-optimized containers," in *2018 USENIX Annual Technical Conference (USENIX ATC '18)*, 2018, pp. 57–70.

[20] A. Klimovic, Y. Wang, P. Stuedi, A. Trivedi, J. Pfefferle, and C. Kozyrakis, "Pocket: Elastic ephemeral storage for serverless analytics," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, 2018, pp. 427–444.

[21] M. Shahrad, R. Fonseca, Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini, "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *Proceedings of the 2020 USENIX Annual Technical Conference*, 2020, p. 14 pages.

[22] W. Lloyd, S. Ramesh, S. Chinthalapati, L. Ly, and S. Pallickara, "Serverless computing: An investigation of factors influencing microservice performance," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 159–169.

[23] M. Malawski and B. Balis, "Serverless computing for scientific applications," *arXiv preprint*, vol. arXiv:2309.01681, 2023.

[24] J. Scheuner and P. Leitner, "Function-as-a-service performance evaluation: A multivocal literature review," *Journal of Systems and Software*, vol. 170, p. 110708, 2020.

[25] Y. Li, Y. Lin, Y. Wang, K. Ye, and C. Xu, "Serverless computing: State-of-the-art, challenges and opportunities," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1522–1539, 2022.

[26] J. M. Hellerstein, J. Faleiro, J. E. Gonzalez, J. Schleier-Smith, V. Sreekanti, A. Tumanov, and C. Wu, "Serverless computing: One step forward, two steps back," *arXiv preprint*, vol. arXiv:1812.03651, 2018.

[27] I. Baldini, P. Cheng, S. J. Fink, N. Mitchell, V. Muthusamy, R. Rabbah, P. Suter, and O. Tardieu, "The serverless trilemma: Function composition for serverless computing," in *Proceedings of the 2017 ACM SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, 2017, pp. 89–103.

[28] V. Avasarala, M. Gorlatova, and P. Mittal, "Fogfn: Serverless computing for fog networks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7299–7312, 2020.

[29] D. Ustiugov, P. Petrov, M. Kogias, E. Bugnion, and B. Grot, "Benchmarking, analysis, and optimization of serverless function snapshots," *Unpublished*, 2021.

[30] G. McGrath and P. Brenner, "Serverless computing: Design, implementation, and performance," *Unpublished*, vol. June 2017, pp. 405–410, 2017.

[31] L. Yu, Y. Zhou, Y. Li, Y. Xie, and F. Qian, "Seuss: Skip redundant serverless scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1073–1087, 2021.

[32] J. Duan, S. Qian, D. Yang, H. Hu, J. Cao, and G. Xue, "Mopar: A model partitioning framework for deep learning inference services on serverless platforms," *arXiv preprint*, vol. arXiv:2404.02445, 2024.

[33] E. Oakes, L. Yang, D. Zhou, K. Houck, T. Harter, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Sock: Rapid task provisioning with serverless-optimized containers," in

2018 USENIX Annual Technical Conference (USENIX ATC '18), 2018, pp. 57–70.

[34] I. Mance, G. McGrath, and P. R. Brenner, "Faasm: Lightweight isolation for webassembly-based serverless computing," in *Proceedings of the 2020 ACM Symposium on Cloud Computing (SoCC '20)*, 2020, pp. 598–612.

[35] S. Eismann, J. Scheuner, and P. Leitner, "Serverless computing: Economic and architectural impact," *IEEE Software*, vol. 38, no. 1, pp. 40–47, 2021.

[36] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with openlambda," in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '16)*, 2016, pp. 66–72.

[37] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, "Cloud programming simplified: A berkeley view on serverless computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2019-3, 2019.

[38] A. Klimovic, Y. Wang, P. Stuedi, A. Trivedi, J. Pfefferle, and C. Kozyrakis, "Pocket: Elastic ephemeral storage for serverless analytics," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, 2018, pp. 427–444.

[39] M. Malawski, A. Gajek, A. Zima, K. Figiela, and M. Bubak, "Benchmarking heterogeneous cloud functions," *Future Generation Computer Systems*, vol. 110, pp. 110–123, 2020.

[40] G. McGrath and J. Short, "Serverless computing: Design, implementation, and performance," *Journal of Cloud Computing*, vol. 8, no. 1, pp. 1–16, 2019.

[41] M. Shahrad, C. Delimitrou, and D. Wentzlaff, "Architectural implications of function-as-a-service computing," *IEEE Micro*, vol. 39, no. 5, pp. 40–47, 2019.

[42] A. Sriraman and A. C. Arpaci-Dusseau, "Fault-tolerant and elastic scaling for serverless platforms," in *Proceedings of the ACM Symposium on Cloud Computing (SoCC '19)*, 2019, pp. 1–13.

[43] L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. Swift, "Peeking behind the curtains of serverless platforms," in *2018 USENIX Annual Technical Conference (USENIX ATC '18)*, 2018, pp. 133–146.

[44] S. Hendrickson, S. Sturdevant, T. Harter, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Openlambda: An open-source serverless computing platform," in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '16)*, 2016.

[45] T. Helmy, "Mitigating auto-scaling delays in elastic-docker for better responsiveness of burst and dynamic workloads," *Journal of Advances in Information Technology*, vol. 15, no. 11, 2024.

[46] H. Yu, H. Wang, J. Li, X. Yuan, and S.-J. Park, "Accelerating serverless computing by harvesting idle resources," in *Proceedings of The Web Conference 2022*, 2022, pp. 1741–1751.