# The State of FaaS: An analysis of public Functions-as-a-Service providers

Nnamdi Ekwe-Ekwe
*School of Computer Science*
*University of St Andrews*
St Andrews, United Kingdom
nnee@st-andrews.ac.uk

Lucas Amos
*lucasamos.dev*
Scotland, United Kingdom
contact@lucasamos.dev

***Abstract*—Serverless computing is a growing and maturing field that is the focus of much research, industry interest and adoption. Previous works exploring Functions-as-a-Service providers have focused primarily on the most well known providers AWS Lambda, Google Cloud Functions and Microsoft Azure Functions without exploring other providers in similar detail. In this work, we conduct the first detailed review of ten currently publicly available FaaS platforms exploring 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.**

***Index Terms*—serverless, faas, functions-as-a-service, serverless computing, state of serverless, state of faas, survey**

## I. Introduction

Serverless is an event-driven computing paradigm allowing end-users to write, deploy and run code in the cloud without managing the underlying infrastructure [1], [2]. In the case of Functions-as-a-Service (FaaS), the user writes their function in a specific language, with their code packaged and deployed on the serverless computing platform. The user allocates resources to the function and chooses a method of invocation, with the function executed in response to requests [3]. Serverless functions are ephemeral in nature and generally stateless, with the function execution and its underlying resources ceasing to exist post invocation. Serverless billing is typically calculated based on a combination of the function's execution time, provisioned resources and number of invocations [4]–[6]. The advantages of serverless computing are becoming increasingly evident. Economically, serverless can allow for significant cost savings with charges incurred based only on when resources are provisioned to run a function [7]. From an application developer perspective, the developer does not need to manage underlying infrastructure, provision hardware, etc. all they need to do is deploy their function and then process requests [8]. Surveys from industry show increasing serverless adoption amongst end-users with about half of customers across the largest public cloud providers using at least one serverless solution in their deployments [9].

The serverless field is still maturing and as such there is limited literature in addition to several open research areas and challenges [1], [3], [10]. In the state of the art, three main public FaaS providers are frequently discussed and analysed. They are AWS Lambda [11], Google Cloud Functions [12] and Microsoft Azure Functions [13]. Studies of serverless adoption and use cases [14], [15] explore important areas such as the most popular languages being used for serverless applications to the most used FaaS platforms, amongst others. However, in recent years there has been an expansion of functionalities offered by several FaaS providers in addition to new FaaS offerings being launched. There has been little to no detailed review of these new FaaS platforms, what their offerings are and how they sit within the overall serverless landscape. Additionally, many new FaaS providers are offering functionality that has deviated from the more traditional Functions-as-a-Service offerings [8], [16], [17] which as of yet, have not been explored in the state of the art.

In this paper, we conduct the first detailed review and analysis of ten publicly available FaaS offerings. We provide an overview of the features offered by each and draw observations from our analysis. The goal of this work is to, (1) provide an overview to the research community and industry practitioners as to the current state of the FaaS landscape and (2), explore their various capabilities in more detail. The structure of this paper is as follows. We start by discussing the methodology we use to obtain the list of the FaaS platforms we review in Section II. We then discuss the overview and capabilities of each platform (Section III) exploring the history of the platform (Section III-A), the languages supported (III-B), invocation types (III-C), resource configurations (III-D), regions supported (III-E) and pricing models (III-F). We then make a number of observations in Section IV, discuss related work (Section V) and then conclude this paper in Section VI.

## II. Methodology

Functions-as-a-Service (FaaS) describes self-contained functions hosted on some compute platform that are invoked by some event and carry out some processing [3], [10], [18], [19]. To create our collection of FaaS providers to evaluate, we conducted internet searches to garner a list of providers offering public FaaS platforms. Similar to the research state of the art, there appears to be no central up-to-date repository of which providers offer FaaS platforms, with information being

TABLE I
FAAS OFFERINGS EVALUATED

| Company | FaaS Offering | Classification |
|---|---|---|
| Alibaba | Alibaba Function Compute [26] | Generalised |
| Amazon | AWS Lambda [11] | Generalised |
| Cloudflare | Cloudflare Workers [27] | Generalised |
| DigitalOcean | Digital Ocean Functions [28] | Generalised |
| Google | Google Cloud Functions [12] | Generalised |
| IBM | IBM Cloud Code Engine [25] | Generalised |
| Microsoft | Microsoft Azure Functions [13] | Generalised |
| Netlify | Netlify Functions [17] | Specialised |
| Oracle | Oracle Cloud Functions [29] | Generalised |
| Vercel | Vercel Functions [16] | Specialised |

highly fragmented across several sources. Our methodology to obtain our list of providers was twofold:

1) *Explore public cloud providers* and research whether or not they offer a FaaS platform. FaaS has grown out of public cloud providers and so we typically see serverless offerings being tied directly to the overall cloud offering. The first known FaaS platform, Lambda, grew out of AWS [20] and so we explored the list of all publicly available cloud providers worldwide to see which offered FaaS platforms to include in our review.
2) *Explore articles/posts written by industry practitioners and developers*: FaaS is being used extensively by industry and as such, many industry practitioners frequently share their experiences/insights using FaaS. For example, Dunelm, a large UK home furnishing retailer is the largest user of Lambda in Europe and frequently writes about its experiences deploying production workloads on Lambda [21], [22]. Such posts were explored to find public FaaS providers being used by the developer community.

From a combination of these two approaches, our methodology returned a list of *ten* FaaS providers. We found that, aside from the three major cloud companies (Amazon, Google and Microsoft), a number of new FaaS providers have emerged in recent years. We also noted that a FaaS provider, IBM, recently deprecated their FaaS offering - IBM Cloud Functions (which has been cited in prior research in the serverless state of the art such as in work [23]. IBM Cloud Functions was deprecated at the end of 2023 [24] and replaced with IBM Code Engine [25]. We classified our list of providers as *generalised* or *specialised* FaaS offerings. Most of these providers (80%) are *generalised*, allowing users to upload *any* kind of code written in a number of languages to the platform, with the code executed in response to events. Most of these generalised providers are delivered as part of an overall larger public cloud offering. The other 20% of those providers are *specialised*, that is, the offering is tightly coupled to deploying more specific kinds of workloads tied to the overall business offering of the provider. The list of providers we explore are shown in Table I which are analyse next.

## III. OVERVIEW AND CAPABILITIES

### A. History

The timeline of the creation of the various FaaS providers in Table I are shown in Figure 1. We garnered this information from a mixture of press releases by the various companies or (where such releases weren't available) from 1) blog posts written by industry practitioners using the FaaS service or 2) the earliest available Release Notes of that FaaS service. AWS provided the first preview of a FaaS offering in 2014 [30] before making their platform Generally Available (GA) in 2015 [20]. Microsoft were next, releasing Azure Functions in 2016 [31]. Google released a preview of Google Cloud Functions in 2016, before their offering became GA two years later [32]. Alibaba Functions Compute [33] and IBM Cloud Functions [34] were launched in 2017. Cloudflare Workers became GA in 2018 [35] with Netlify Functions also released that year [36]. Oracle Functions [29] was released in 2019. Vercel had a serverless offering in 2019 [37], followed by IBM's new serverless offering Code Engine in 2020 (beta) [38]. This became generally available in 2021 [39] with IBM Cloud Functions [24] being deprecated in 2023 [25]. Vercel released a new offering, Vercel Edge Functions [40] in 2022 with DigitalOcean Functions [41] also launching that same year. A new offering by Netlify, Netlify Edge Functions [42] was released in 2023. As mentioned in the previous section, eight out of ten of these providers provide *generalised* serverless compute capabilities. Two of the providers, Vercel and Netlify have FaaS platforms that are more closely tied to the functionality of their overall service. Users of these platforms typically deploy code that is more specifically tied to the overall offering of the company. For example, Vercel is a platform allowing users to deploy web apps and as such, Vercel's Functions are typically used to write functions relating to authentication, image/video processing on web apps, etc. [16]. Netlify is similar to Vercel, also allowing users to deploy web applications, e-commerce platforms, etc. to the web. Their own functions offering is also tied to their web platform offering with functions being used to perform tasks such as updating a database on a form being submitted, sending an email, etc. [43]. We next explore the underlying features of each of these platforms.

### B. Languages supported

There are a wide variety of languages natively supported by the ten FaaS providers (Table II) with additional languages supported via custom runtimes. AWS natively supports 7 languages in addition to other custom runtimes [44]. Microsoft [45] and Alibaba [46] natively support 6 languages as well as the use of custom runtimes for other languages such as Rust, Go, etc. Google [47] and Oracle [48] support 7 and 6 languages respectively but do not support custom runtimes. DigitalOcean [49] and Vercel [50] support 4 languages natively with DigitalOcean not having custom language support. However Vercel has a custom JavaScript "Edge Runtime" available to run lightweight code using a set of WebStandard APIs.

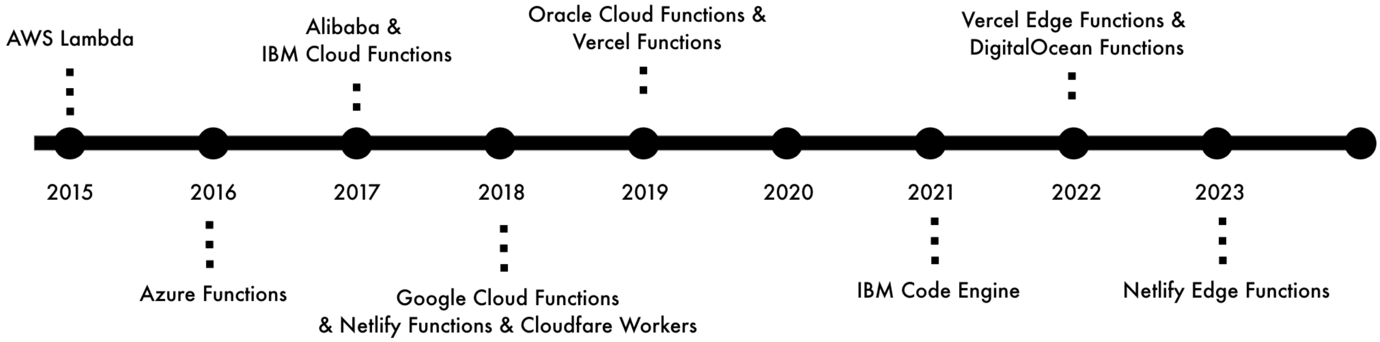


Fig. 1. Timeline of General Availability (GA) of FaaS providers

TABLE II
LANGUAGES SUPPORTED

| | **Alibaba** | **AWS** | **Azure** | **Cloudflare** | **Digital Ocean** | **Google** | **IBM** | **Netlify** | **Oracle** | **Vercel** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Node.js** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Java** | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ |
| **Python** | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| **.NET/C#** | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ |
| **Powershell** | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| **Ruby** | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ |
| **Go** | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| **PHP** | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| **Custom Runtime** | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

TABLE III
RESOURCE CONFIGURATIONS

| | **Alibaba** | **AWS** | **Azure** | **Cloudflare** | **Digital Ocean** | **Google** | **IBM** | **Netlify** | **Oracle** | **Vercel** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Maximum Memory** | 32GB | 10GB | 14GB | 128MB | 1GB | 32GiB | 4GB | 1GB | 2GB | 3GB(S), 128MB(E) |
| **Maximum Tempo-rary Storage** | 10GB | 10GB | 1TB | use-case dependent | NS | NS | NS | NS | NS | NS |
| **Maximum Timeout** | 24h | 15m | $\infty$ | $\infty$ | 15m | 60m/HTTP, 9m/event-driven | 2m | 10s | 5m | 15m(S), $\infty$(E) |

IBM [51], Netlify [52] and Cloudflare [53] only support 2 languages. Cloudflare has a custom runtime for languages that compile to WebAssembly, such as Rust, C++, etc.

### C. Invocation Types

Functions are primarily event-driven with a wide variety of *invocation types* supported. Functions can also be invoked on a schedule as typically seen in a cron job. In Lambda, functions can be invoked by HTTP, Queues, in response to a change in an AWS service or via polling [54]. Multiple triggers to the *same* function are supported - for example a function being invoked by both an HTTP request and change in an S3 bucket, etc. Azure Functions also allow for a wide variety of triggers and bindings [55]. In Azure, a trigger defines how a function is invoked. Unlike Lambda, an Azure function can only have at most **one** trigger. Bindings allow for Azure functions to be connected to other Azure services and receive and/or send data to other services [55]. Google Cloud Functions split their invocation types into two: (1) HTTP and (2) Event triggers. HTTP triggered functions respond only to HTTP events, whilst functions with Event triggers respond to events such as Pub/Sub, from other Google services (via Eventarc), etc. [56]. GCF can only have at most one trigger. Alibaba supports multiple invocation methods (HTTP, via events, etc. [57]) with similar functionality shown in Oracle [58]. IBM [59], DigitalOcean [60], Netlify [52], Vercel [61] and Cloudflare [62] only allow HTTP invocations.

### D. Resource configurations and limits

We have highlighted the resource configurations common to all providers (Memory, Storage and Timeout) in Table III. Please note that "Storage" refers to temporary/ephemeral storage that can be attached to a function, acting as a local disk. Some FaaS providers allow for further configuration parameters that are not common to all of them, such as CPU, concurrency settings, etc. Typically memory size for FaaS is displayed in MB but for readability, we have converted MBs that are $>= 1GB$ equivalent to GBs. Please also note that *NS* in the table means not supported as a configurable option. Edge and Standard functions are also denoted as "E" and "S" in the table. AWS allows the developer to set the memory, timeout and storage size of their Lambda function. The developer can set a maximum memory size and storage of 10GB, and a maximum timeout of 15 minutes [63], compute capacity scales proportionally with memory and cannot be set independently. The maximum number of simultaneous executions for a function can also be set via concurrency options [64]. Alibaba provides more generous maximum memory (32GB) with maximum storage being the same as with Lambda (10GB). The maximum timeout is also the largest of all the FaaS providers at 24h [65]. The number of vCPUs, in addition to the concurrency limit for the function [66] can also be configured with Alibaba. Azure allows the user to set a maximum function memory of 14GB, with storage being 1TB and an unlimited/unbounded maximum timeout [67]. It is worth noting that although a function can technically have an unlimited timeout, Microsoft only guarantees this for up to 60 minutes, stating that activities such as OS/runtime/vulnerability patching, etc. can cancel function execution after this time. Also, due to default settings on Azure Load Balancer, HTTP invoked functions have a maximum timeout of 230s [67]. Google has the most generous maximum memory allocation of all being 32GiB. The maximum timeout for a Google Cloud Function is dependent on its *type* - up to 60 minutes for an HTTP function or 9 minutes for an event-driven function. Similar to Alibaba, Google allows the developer to specify the vCPUs and, like Azure and AWS, the concurrency [68]. Google, unlike the other aforementioned providers, does not allow the user to add/modify additional storage for a function. IBM CodeEngine has a more limited set of configurations and limits open to the end-user. The end-user's function is limited to running for 2m, with maximum memory size being 4GB. The developer can also specify the "scale-down" delay of their function which sets how long an instance remains alive before shutting down - this is at most 6m and is useful to prevent cold start behaviour [69]. DigitalOcean similarly offers a limited set of configuration options. A function on DigitalOcean can only have a maximum memory allocation of 1GB, however a more generous maximum timeout of 15m [70]. Oracle allows for a maximum memory size of 2GB, no option to specify storage, with a maximum timeout of 5m [71]. Cloudflare Workers have a maximum memory of 128MB and have no maximum duration [72]. The concept of storage for a function is slightly different in Cloudflare Workers, with the developer able to make use of different storage types (key-value stores, object stores, queues, etc.) for their function to persist data [73]. Cloudflare however does not provide any option to modify temporary/ephemeral storage for a function. Netlify's Functions have a maximum memory of 1GB and maximum timeout of 10s or 15m depending on the type of function (synchronous function invocations have a 10s timeout; a background function more suited to longer-running tasks is up to 15m [74], [75]). Finally, Vercel has two different maximum memory allocations depending on the type of function being run. Standard functions can be configured with memory up to 3GB, with functions using the Edge Runtime capped at 128MB. Similarly, the maximum timeout of a function is different depending on the type of function with Standard functions being up to 15m, and edge functions having an unlimited timeout (suited for streaming data) [76].

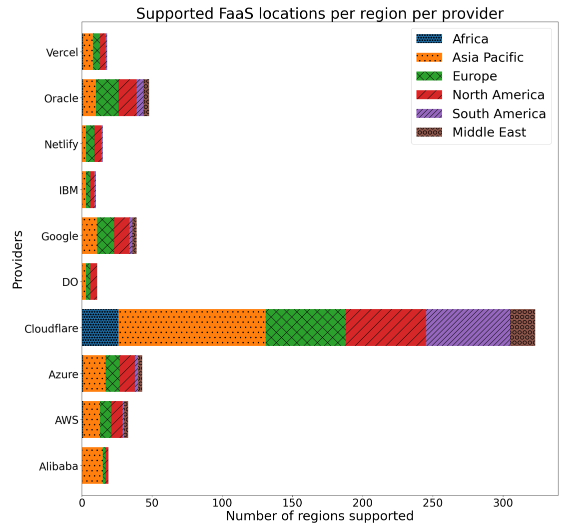


Fig. 2. Supported FaaS locations per region per provider

### E. Regions supported

The various providers support FaaS across their globally distributed locations (Figure 2). Alibaba Functions can be deployed in a total of 19 locations [77] with most of them (15) being in the Asia-Pacific region. It is also worth noting that not all Function features are available universally across every supported region - for example GPU-accelerated Function instances. AWS Lambda is available in 33 locations worldwide on every continent with Europe, North America and Asia-Pacific having the most number of locations supported [78]. Azure Functions are supported in 43 locations [79]. Cloudflare has the largest number of supported locations (323) with functions deployable worldwide [80]. The reason for this significant coverage is due to Cloudflare being an edge-first provider, and as such covers a wide breadth of locations. DigitalOcean supports one of the fewest number of locations (11), with no locations supported in Africa, South America or the Middle East. Google supports 39 locations [81]. IBM supports one less location than DigitalOcean (10) [82]. Netlify supports a subset of AWS's regions as its platform uses AWS as the underlying infrastructure provider [83] (15 locations). Oracle supports 48 locations across the world on every continent [84], [85]. Finally, Vercel supports 18 locations [86].

### *F. Pricing*

The different FaaS platforms have various pricing models. Pricing with FaaS providers is typically made up of two core elements, (1) the number of invocations, and (2) the resources provisioned (CPU/RAM). Other elements such as storage, networking, etc. can differ between providers in the way they are charged (if at all). We report on the core elements that make up the pricing model for each provider next. The common pricing approach amongst FaaS providers is based on (1) the resources consumed and the duration of their use in addition to (2) the number of invocations made. FaaS providers typically charge a fee per GB-second of usage for resources allocated to the function in addition to a flat-fee per $n$ million invocations of a function. The total cost of a function running $P$ can be modelled by Equation 1:

$$P = R_x + I \tag{1}$$

where $R_x$ is the resource consumption charge and $I$ is the invocation charge. For AWS [4], [87], Azure [6] and Oracle [29], $R_x$ is based on the duration of the usage of memory allocated to the function. Google [5], IBM [88], [89], and Alibaba [90] calculate $R_x$ based on both the duration of usage of memory as well as CPUs allocated to the function. Finally, with Cloudflare, $R_x$ is based solely on duration of CPU usage [91]. DigitalOcean does not charge an invocation fee, with only an $R_x$ charge based on duration of memory used [92]. The specialised providers have different pricing models from the generalised FaaS providers. Netlify offers plans for their users that come with a fixed amount of invocations per month with the user automatically upgraded to the next tier when usage exceeds that tier [93]. Vercel has similar elements to Netlify and Cloudflare, offering plans that include 100GB-hours as well as 500K "execution units" for Edge functions [16]. These execution units are equivalent to 50ms of CPU time [94]. Two plans are offered - a standard plan and a pro plan with the user allowed to exceed the pro plan GB-hours/execution units allocations for a fee. Enterprise plans are also offered by Vercel for users whose usage does not fit into either of these two plans. Finally, the providers also include a free tier for end users before charging them for function usage (Table IV).

## IV. Observations

The ten providers we have analysed offer varying services, features, prices and are available in various locations across the world. We note a number of observations from our analysis:

TABLE IV
FaaS free tier

| Company | Free tier offering |
|---|---|
| Alibaba | 1M invocations, 400K GB-seconds |
| Amazon | 1M invocations, 400K GB-seconds |
| Cloudflare | 100K invocations per day, 10ms CPU time per invocation |
| DigitalOcean | 90K GiB seconds |
| Google | 400K GB-seconds, 200K GHz-seconds, 5GB network transfer, 2M invocations |
| IBM | 100K vCPU seconds |
| Microsoft | 1M invocations, 400K GB-seconds |
| Netlify | N/A |
| Oracle | 2M invocations, 400K GB-seconds |
| Vercel | "Hobby" free tier |

*Observation 1*: The FaaS provider landscape has grown, but newer providers are yet to mature to challenge the more established platforms.

Since the turn of the decade (2019/20), we have seen five new FaaS platforms emerge (Figure 1). Whilst this adds new competition to the FaaS landscape, the functionality offered by these platforms are yet to mature. We saw this in Table III, with more limited configuration options (such as memory, timeout, etc.) as well as limits on the languages supported (Table II). These limits were also evident in FaaS support across the various regions with DigitalOcean, one of the newest providers, having some the fewest number of locations supported worldwide. In order for these newer FaaS providers to compete with the likes of AWS, Microsoft, Alibaba and Google, more functionality and flexibility needs to be offered.

*Observation 2*: Whilst the various FaaS providers have many common features/functionalities, some significant differences remain, impacting the types of applications supported.

We noted in our earlier analysis how AWS Lambda allowed for the developer to set multiple triggers simultaneously to the same function (Section III-C). There are some use cases where this could be very useful, for example being able to invoke a function externally via an HTTP call whilst also allowing it to be invoked via an internal service that adds a message to a queue. The use-cases and applications for serverless are varied [95] and so the developer needs to ensure that their chosen FaaS provider can accommodate their application use-case. We highlighted the functionality by AWS Lambda as it was different to all the other providers. These providers either (1) limited the user to choosing at most one invocation type for their function or (2) had a limited set of invocation options supported by default. This was also seen in the case of languages supported with some providers supporting a large number of languages by default and allowing for other languages to be implemented via custom runtimes. These various constraints need to be carefully considered by the application developer when choosing a particular FaaS provider.

*Observation 3*: Some of the FaaS providers are making the edge available for developers to deploy workloads.

Running functions on the network edge, reducing latency and speeding up function serving and handling of requests is now being introduced across several platforms such as AWS, Netlify, Vercel and Cloudflare. The advantages of the edge with its low latency and how this can be leveraged for functions, handling requests close to the users in geographically distributed locations is clear. Functionality that is run on these providers is very web-oriented with functions on Netlify and Vercel typically tied to their web application workloads such as authentication, form processing, etc. Cloudflare is more generalised in not restricting the kinds of workloads that can be run on the edge. From our work, we note how edge locations attract significantly reduced resources (128MB maximum for Vercel and Cloudflare) and so decisions on which functions to run on the edge needs to be carefully considered.

*Observation 4*: JavaScript, Python and Go are the languages most frequently supported by the FaaS providers

JavaScript/Node.js is the only language supported by all the FaaS platforms. This was closely followed by Python and Go that were both tied on 8 providers. Java and C# were both tied on 5, with the other languages being sparsely supported by the platforms. The use of the custom runtimes was supported by only 4 out of the 10 providers. Custom runtimes are certainly advantageous in accommodating for any gaps in native language support but this feature is not well established across the explored FaaS providers. It would be interesting to see the performance of these custom runtimes across the providers that support them and if there are any differences. Works [96], [97] have explored how language affects function performance and price.

*Observation 5*: Resources and configuration options are generally generous amongst providers, with the newer FaaS providers providing more limited resources.

Most of the providers offer generous resource configurations for their users' function workloads. Alibaba overall has the most generous, offering the largest amount of memory and timeout of all the providers (Table III). Google offers a slightly higher memory offering than Alibaba but does not provide any separate storage configuration and has a significantly smaller timeout. Azure offers the highest amount of storage and a high amount of memory per function with a function timeout that is not limited (at least 60m uninterrupted). Cloudflare does allow for an unlimited timeout albeit with a relatively small amount of memory available. The newer providers (DigitalOcean, IBM, Oracle) have the most limited set of configurations available of all the generalised providers. Vercel is quite generous with its function configuration (depending on whether the function is standard or edge) however those functions are tied directly to its overall web offering with Netlify also being quite limited.

*Observation 6*: There are differences amongst the various providers' pricing models, but core elements are common to them all.

In general, pricing follows a duration/resource model, whereby an end-user is charged for the duration of their function and the resources they consume. They are also additionally charged for the number of invocations of that function. Cloudflare is the only provider that bills based on CPU-time rather than duration, a move they argue is advantageous as it does not include "idle time" of the function waiting on network requests/IO [98]. There is work exploring optimising FaaS function placement to take advantage of price [99], [100] however an exploration of these various differences using real-world workloads across the providers reviewed would further add to exploring this area.

## V. Related Work

Several related works exploring the state of the art have been produced by both the research and industry communities. Key surveys [1]–[3], [95], discuss the state of the art of serverless, exploring everything from the evolution and creation of serverless, to common use cases, to the key challenges and opportunities in the field, etc. From industry, we note works such as DataDog's The State of Serverless report [9] also exploring key trends/statistics within Serverless today. Empirical studies understanding and comparing multiple FaaS platforms [23], [101]–[103] have been produced primarily comparing the big three (Azure, AWS and Google) in addition to IBM OpenWhisk. We also note works from industry such as those by Mikhail Shilkov [104] exploring a wide range of topics, such as exploring cold start behaviour across the various platforms, to testing their scalability, etc. Several severless benchmark frameworks [105]–[107] have also garnered empirical evidence on the performance and cost of executing functions on the big three FaaS providers, drawing conclusions from their experiments.

## VI. Conclusion

In this paper, we have provided the first detailed review of all ten currently available public FaaS providers. We have discussed various elements of each provider from their history, invocation types, features/resource configurations supported, regions covered, and pricing models employed. We have also made a number of observations from our research as to the state of public FaaS providers and provided insight as to where further work can be performed. Empirical evaluations of various facets of these additional providers would be advantageous to add to already burgeoning research in this area.

## VII. Acknowledgements

We want to thank the anonymous reviewers for their time and valuable feedback.This study is based on the use of primary sources accessed from several archives. A full list of the sources is available in the reference list.

## References


[1] Z. Li, L. Guo, J. Cheng, Q. Chen, B. He, and M. Guo, "The serverless computing survey: A technical primer for design architecture," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10s, pp. 1–34, 2022.

[2] 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, no. 5, pp. 1–61, 2023.

[3] H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless computing: a survey of opportunities, challenges, and applications," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1–32, 2022.

[4] AWS. (2024) Serverless computing - aws lambda pricing - amazon web services. [Online]. Available: https://aws.amazon.com/lambda/pricing/

[5] Google. (2024) Cloud functions pricing. [Online]. Available: https://cloud.google.com/functions/pricing

[6] Microsoft. (2024) Azure functions pricing. [Online]. Available: https://azure.microsoft.com/en-gb/pricing/details/functions/

[7] G. Adzic and R. Chatley, "Serverless computing: economic and architectural impact," in *Proceedings of the 2017 11th joint meeting on foundations of software engineering*, 2017, pp. 884–889.

[8] Cloudflare. (2024) Why use serverless computing? — pros and cons of serverless. [Online]. Available: https://www.cloudflare.com/en-gb/learning/serverless/why-use-serverless/

[9] Datadog. (2023) The state of serverless — datadog. [Online]. Available: https://www.datadoghq.com/state-of-serverless/

[10] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, "Survey on serverless computing," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–29, 2021.

[11] AWS. (2024) Serverless function, faas serverless - aws lambda. [Online]. Available: https://aws.amazon.com/lambda/

[12] Google. (2024) Cloud functions. [Online]. Available: https://cloud.google.com/functions

[13] Microsoft. (2024) Azure functions – serverless functions in computing. [Online]. Available: https://azure.microsoft.com/en-gb/products/functions

[14] S. Eismann, J. Scheuner, E. Van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "A review of serverless use cases and their characteristics," *arXiv preprint arXiv:2008.11110*, 2020.

[15] I. Pavlov, S. Ali, and T. Mahmud, "Serverless development trends in open source: a mixed-research study," 2019.

[16] Vercel. (2023) Serverless functions overview. [Online]. Available: https://vercel.com/docs/functions/serverless-functions

[17] Netlify. (2024) Functions overview. [Online]. Available: https://docs.netlify.com/functions/overview/

[18] E. Van Eyk, A. Iosup, S. Seif, and M. Thömmes, "The spec cloud group's research vision on faas and serverless architectures," in *Proceedings of the 2nd international workshop on serverless computing*, 2017, pp. 1–4.

[19] E. Van Eyk, L. Toader, S. Talluri, L. Versluis, A. Uță, and A. Iosup, "Serverless is more: From paas to present cloud computing," *IEEE Internet Computing*, vol. 22, no. 5, pp. 8–17, 2018.

[20] AWS. (2015) Aws lambda is generally available — aws compute blog. [Online]. Available: https://aws.amazon.com/blogs/compute/aws-lambda-is-generally-available/

[21] Dunelm. (2023) Dunelm wins "highly commended" for best tech refresh project at uk it awards 2023 — by paul kerrison — dunelm technology. [Online]. Available: https://engineering.dunelm.com/dunelm-wins-highly-commended-for-best-tech-refresh-project-at-uk-it-awards-2023-141bb3fd495e

[22] Aleios. (2023) Serverless: Dun-our-way — aleios. [Online]. Available: https://www.aleios.com/talks/serverless-dun-our-way-mark-white-dunelm

[23] K. Figiela, A. Gajek, A. Zima, B. Obrok, and M. Malawski, "Performance evaluation of heterogeneous cloud functions," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 23, p. e4792, 2018.

[24] IBM. (2024) Deprecation overview - ibm cloud docs. [Online]. Available: https://cloud.ibm.com/docs/openwhisk?topic=openwhisk-dep-overview

[25] ——. (2024) Ibm cloud code engine. [Online]. Available: https://www.ibm.com/products/code-engine

[26] Alibaba. (2024) Function compute: Fully hosted and serverless running environment - alibaba cloud. [Online]. Available: https://www.alibabacloud.com/fr/product/function-compute

[27] Cloudflare. (2024) Cloudflare workers®. [Online]. Available: https://workers.cloudflare.com

[28] D. Ocean. (2024) Run functions on-demand. scale automatically. [Online]. Available: https://www.digitalocean.com/products/functions

[29] O. Cloud. (2024) Cloud functions — oracle united kingdom. [Online]. Available: https://www.oracle.com/uk/cloud/cloud-native/functions/

[30] AWS. (2014) Introducing aws lambda. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2014/11/13/introducing-aws-lambda/

[31] Microsoft. (2016) Announcing general availability of azure functions — microsoft azure blog. [Online]. Available: https://www.digitalocean.com/blog/introducing-digitalocean-functions-serverless-computing

[32] TechCrunch. (2018) Google's cloud functions serverless platform is now generally available — techcrunch. [Online]. Available: https://techcrunch.com/2018/07/24/googles-cloud-functions-serverless-platform-is-now-generally-available/

[33] Alibaba. (2017) Release notes in 2017 - function compute (2.0) - alibaba cloud documentation center. [Online]. Available: https://www.alibabacloud.com/help/en/fc/product-overview/release-notes-in-2017

[34] IBM. (2017) New cloud functions ui enhancements. [Online]. Available: https://www.ibm.com/blog/announcement/new-cloud-functions-ui-enhancements/

[35] Cloudflare. (2018) Introducing cloudflare workers: Run javascript service workers at the edge. [Online]. Available: https://blog.cloudflare.com/introducing-cloudflare-workers

[36] Netlify. (2018) Netlify's aws lambda functions bring the backend to your frontend workflow. [Online]. Available: https://www.netlify.com/blog/2018/03/20/netlifys-aws-lambda-functions-bring-the-backend-to-your-frontend-workflow/

[37] Vercel. (2019) Inspecting serverless functions - vercel. [Online]. Available: https://vercel.com/blog/functions-tab

[38] IBM. (2020) Announcing ibm cloud code engine: The easy way to run your code, containers, and batch jobs - ibm blog. [Online]. Available: https://www.ibm.com/blog/announcement/ibm-cloud-code-engine/

[39] ——. (2021) Ibm cloud code engine is now generally available - ibm blog. [Online]. Available: https://www.ibm.com/blog/announcement/ibm-cloud-code-engine-is-now-generally-available/

[40] Vercel. (2022) Vercel edge functions – vercel. [Online]. Available: https://vercel.com/blog/edge-functions-generally-available

[41] DigitalOcean. (2022) Introducing digitalocean functions: A powerful serverless solution. [Online]. Available: https://www.digitalocean.com/blog/introducing-digitalocean-functions-serverless-computing

[42] Netlify. (2023) Netlify presents: Edge functions general availability. [Online]. Available: https://www.netlify.com/blog/edge-functions-general-availability/

[43] Earthly. (2023) Introduction to netlify cloud functions - earthly blog. [Online]. Available: https://earthly.dev/blog/netlify-cloud-functions/

[44] AWS. (2024) Aws lambda - faqs. [Online]. Available: https://aws.amazon.com/lambda/faqs/

[45] Microsoft. (2024) Supported languages in azure functions — microsoft learn. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/supported-languages

[46] Alibaba. (2024) Function compute:overview. [Online]. Available: https://www.alibabacloud.com/help/en/fc/user-guide/overview-35

[47] Google. (2024) Runtime support — cloud functions documentation. [Online]. Available: https://cloud.google.com/functions/docs/runtime-support

[48] Oracle. (2024) Languages supported by oci functions. [Online]. Available: https://docs.oracle.com/en-us/iaas/Content/Functions/Tasks/languagessupportedbyfunctions.htm

[49] DigitalOcean. (2024) Supported runtimes for digitalocean functions. [Online]. Available: https://docs.digitalocean.com/products/functions/reference/runtimes/

[50] Vercel. (2024) Choosing a runtime. [Online]. Available: https://vercel.com/docs/functions/runtimes

[51] IBM. (2024) Function runtimes — ibm cloud docs. [Online]. Available: https://cloud.ibm.com/docs/codeengine?topic=codeengine-fun-runtime

[52] Netlify. (2024) Get started with functions — netlify docs. [Online]. Available: https://docs.netlify.com/functions/get-started/?fn-language=ts

[53] Cloudflare. (2024) Languages - cloudflare workers docs. [Online]. Available: https://developers.cloudflare.com/workers/reference/languages/
[54] AWS. (2024) using aws lambda with other services - aws lambda. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/lambda-services.html
[55] Microsoft. (2024) Triggers and bindings in azure functions — microsoft learn. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings
[56] G. Cloud. (2024) Cloud functions triggers — cloud functions documentation — google cloud. [Online]. Available: https://cloud.google.com/functions/docs/calling
[57] Alibaba. (2024) Function compute 3.0:trigger overview. [Online]. Available: https://www.alibabacloud.com/help/en/functioncompute/latest/trigger-overview
[58] O. Cloud. (2024) Oci functions concepts. [Online]. Available: https://docs.oracle.com/en-us/iaas/Content/Functions/Concepts/functionsconcepts.htm
[59] IBM. (2023) Configuring custom domain mappings for your function — ibm cloud docs. [Online]. Available: https://cloud.ibm.com/docs/codeengine?topic=codeengine-fun-domainmapping
[60] DigitalOcean. (2024) How to develop functions - digitalocean documentation. [Online]. Available: https://docs.digitalocean.com/products/functions/how-to/develop-functions
[61] Vercel. (2024) Vercel functions quickstart. [Online]. Available: https://vercel.com/docs/functions/quickstart
[62] Cloudflare. (2024) Get started guide - cloudflare workers docs. [Online]. Available: https://developers.cloudflare.com/workers/get-started/guide/
[63] AWS. (2024) Configuring lambda function options - aws lambda. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/configuration-function-common.html
[64] ——. (2024) Lambda function scaling - aws lambda. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html
[65] Alibaba. (2024) Limits on resource usage - function compute (2.0) - alibaba cloud documentation center. [Online]. Available: https://www.alibabacloud.com/help/en/fc/product-overview/limits
[66] ——. (2024) create and update a function - function compute (2.0) - alibaba cloud documentation center. [Online]. Available: https://www.alibabacloud.com/help/en/fc/manage-functions
[67] Microsoft. (2023) Azure functions scale and hosting — microsoft learn. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale
[68] Google. (2024) Configure memory and vcpu limits — cloud functions documentation — google cloud. [Online]. Available: https://cloud.google.com/functions/docs/configuring/memory
[69] IBM. (2023) Limits and quotas for codeengine - ibm cloud docs. [Online]. Available: https://cloud.ibm.com/docs/codeengine?topic=codeengine-limits
[70] DigitalOcean. (2024) How to configure functions :: Digitalocean documentation. [Online]. Available: https://docs.digitalocean.com/products/functions/how-to/configure-functions/
[71] Oracle. (2024) Changing default memory and timeout settings. [Online]. Available: https://docs.oracle.com/iaas/Content/Functions/Tasks/functionscustomizing.htm
[72] Cloudflare. (2024) Limits - cloudflare workers docs. [Online]. Available: https://developers.cloudflare.com/workers/platform/limits/
[73] ——. (2024) Choosing a data or storage product - cloudflare workers docs. [Online]. Available: https://developers.cloudflare.com/workers/platform/storage-options/
[74] Netlify. (2024) Background functions overview — netlify docs. [Online]. Available: https://docs.netlify.com/functions/background-functions/
[75] ——. (2024) Functions overview — netlify docs. [Online]. Available: https://docs.netlify.com/functions/overview/
[76] Vercel. (2024) Choosing a runtime. [Online]. Available: https://vercel.com/docs/functions/runtimes
[77] Alibaba. (2024) Supported regions - function compute (2.0) - alibaba cloud documentation center. [Online]. Available: https://www.alibabacloud.com/help/en/fc/product-overview/region-availability
[78] AWS. (2024) Aws services by region - aws. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/
[79] Microsoft. (2024) Azure products by region — microsoft azure. [Online]. Available: https://azure.microsoft.com/en-gb/explore/global-infrastructure/products-by-region/
[80] Cloudflare. (2024) Cloudflare global network — data center locations — cloudflare. [Online]. Available: https://www.cloudflare.com/en-gb/network/
[81] Google. (2024) Cloud functions locations — cloud functions documentation — google cloud. [Online]. Available: https://cloud.google.com/functions/docs/locations
[82] IBM. (2024) Ibm cloud docs. [Online]. Available: https://cloud.ibm.com/docs/codeengine?topic=codeengine-regions
[83] Netlify. (2024) Optional configuration for functions — netlify docs. [Online]. Available: https://docs.netlify.com/functions/optional-configuration/?fn-language=go#region-3
[84] Oracle. (2024) Oracle functions now generally available. [Online]. Available: https://blogs.oracle.com/cloud-infrastructure/post/oracle-functions-now-generally-available
[85] ——. (2024) Regions and availability domains. [Online]. Available: https://docs.oracle.com/en-us/iaas/Content/General/Concepts/regions.htm
[86] Vercel. (2024) Vercel edge network regions. [Online]. Available: https://vercel.com/docs/edge-network/regions
[87] AWS. (2024) Lambda - how aws pricing works. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/how-aws-pricing-works/lambda.html
[88] IBM. (2024) Pricing for code engine — ibm cloud docs. [Online]. Available: https://cloud.ibm.com/docs/codeengine?topic=codeengine-pricing
[89] ——. (2024) Ibm cloud free tier. [Online]. Available: https://www.ibm.com/cloud/free
[90] Alibaba. (2024) Billable items and billing rules - function compute (2.0) - alibaba cloud documentation center. [Online]. Available: https://www.alibabacloud.com/help/en/fc/product-overview/billing-overview
[91] Cloudflare. (2024) Pricing - cloudflare workers docs. [Online]. Available: https://developers.cloudflare.com/workers/platform/pricing/
[92] DigitalOcean. (2024) Functions pricing. [Online]. Available: https://docs.digitalocean.com/products/functions/details/pricing/
[93] Netlify. (2024) Pricing and plans — netlify. [Online]. Available: https://www.netlify.com/pricing/
[94] Vercel. (2024) Usage in the dashboard. [Online]. Available: https://vercel.com/docs/limits/usage#execution-units
[95] S. Eismann, J. Scheuner, E. Van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "The state of serverless applications: Collection, characterization, and community consensus," *IEEE Transactions on Software Engineering*, vol. 48, no. 10, pp. 4152–4166, 2021.
[96] R. Cordingly, H. Yu, V. Hoang, D. Perez, D. Foster, Z. Sadeghi, R. Hatchett, and W. J. Lloyd, "Implications of programming language selection for serverless data processing pipelines," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2020, pp. 704–711.
[97] D. Jackson and G. Clynch, "An investigation of the impact of language runtime on the performance and cost of serverless functions," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. IEEE, 2018, pp. 154–160.
[98] Cloudflare. (2024) New workers pricing — never pay to wait on i/o again. [Online]. Available: https://blog.cloudflare.com/workers-pricing-scale-to-zero
[99] T. Elgamal, A. Sandur, K. Nahrstedt, and G. Agha, "Costless: Optimizing cost of serverless computing through function fusion and placement," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 300–312.

[100] K. Mahajan, D. Figueiredo, V. Misra, and D. Rubenstein, "Optimal pricing for serverless computing," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[101] K. Chowhan, *Hands-on Serverless Computing: Build, Run and Orchestrate Serverless Applications Using AWS Lambda, Microsoft Azure Functions, and Google Cloud Functions*. Packt Publishing Ltd, 2018.

[102] P. F. Pérez-Arteaga, C. C. Castellanos, H. Castro, D. Correal, L. A. Guzmán, and Y. Denneulin, "Cost comparison of lambda architecture implementations for transportation analytics using public cloud software as a service," *Special Session on Software Engineering for Service and Cloud Computing*, pp. 855–862, 2018.

[103] M. Malawski, K. Figiela, A. Gajek, and A. Zima, "Benchmarking heterogeneous cloud functions," in *Euro-Par 2017: Parallel Processing Workshops: Euro-Par 2017 International Workshops, Santiago de Compostela, Spain, August 28-29, 2017, Revised Selected Papers 23*. Springer, 2018, pp. 415–426.

[104] M. Shilkov. (2024) Serverless — mikhail shilkov. [Online]. Available: https://mikhail.io/tags/serverless/

[105] P. Maissen, P. Felber, P. Kropf, and V. Schiavoni, "Faasdom: A benchmark suite for serverless computing," in *Proceedings of the 14th ACM international conference on distributed and event-based systems*, 2020, pp. 73–84.

[106] T. Yu, Q. Liu, D. Du, Y. Xia, B. Zang, Z. Lu, P. Yang, C. Qin, and H. Chen, "Characterizing serverless platforms with serverlessbench," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 30–44.

[107] M. Copik, G. Kwasniewski, M. Besta, M. Podstawski, and T. Hoefler, "Sebs: A serverless benchmark suite for function-as-a-service computing," in *Proceedings of the 22nd International Middleware Conference*, 2021, pp. 64–78.