**Introduction**

Serverless computing refers to a computing model that empowers developers to develop and execute applications without the need to provision, configure, or manage infrastructure. By delegating infrastructure management to a third party, serverless computing enables developers to focus efforts on creating code that directly adds value to the business. While the concept of serverless computing can refer to a broad range of services, the type of serverless computing discussed in this paper executes functions in response to predefined events or triggers. Also called "functions as a service," this type of serverless computing features short-lived functions that define applications that are executed within a serverless infrastructure.

IDC defines serverless computing as a model of computing that:

- Manages the provisioning and dynamic allocation of infrastructure resources as needed
- Uses discrete functions that execute in response to events or triggers that are defined by each application
- Charges customers by means of a utility pricing model similar to pay-as-you-go pricing models for cloud computing more generally
- Exhibits high availability

Serverless computing provides a fully managed deployment environment and encourages a microservices-style architecture (through the use of functions) that allows developers to rapidly deploy applications. Moreover, serverless computing optimizes application performance by managing the selection of the infrastructure, compute, storage, and networking resources required by each application. Because the selection of resources is dynamically managed (and abstracted from the user) for each application, serverless computing avoids overprovisioning or underprovisioning applications, thereby optimizing application performance and minimizing infrastructure-related costs.
Because serverless computing executes applications on demand instead of running applications persistently, IT resources are used only in response to the realization of customer-defined events or triggers. The model of serverless computing whereby applications are executed on demand, using event-based triggers, on fully managed cloud-based infrastructures delivers economic, operational, technical, and experiential benefits to customers and developers.

In a worldwide IDC survey of more than 3,000 developers, 55.7% of respondents professed to using or having firm plans to implement serverless computing on public cloud infrastructure, as illustrated in Figure 1.

Figure 1: *Current and Future Use of Serverless Computing*

80.9% of IDC survey respondents are educating themselves about, evaluating, planning to implement, or currently using serverless technology.

Source: IDC, 2018

IDC's survey also found that respondents are educating themselves about and evaluating serverless computing. This data suggests that developers perceive serverless functions favorably and are actively taking steps to evaluate the functionality of serverless computing.
Benefits of Serverless Computing

Serverless computing delivers the following benefits to customers and developers:

» **Reduction of operational overhead.** Serverless computing reduces operational overhead by freeing development teams from the burden of managing infrastructure and the logic responsible for its implementation. Additionally, serverless computing allows teams to outsource competencies that are not central to their core business, such as provisioning containers or managing high availability. By transferring the operational responsibility of managing infrastructure to a third-party cloud provider, serverless computing empowers developers to focus their efforts on designing and developing applications. The ability of serverless computing to empower developers to focus solely on application development is one of the key benefits of the computing model because it improves developer productivity, accelerates the development process, and enhances the delivery of value to customers with more effective and more powerful applications.

» **Lowering of capital and operational expenses.** One of the principal advantages of serverless computing consists of the economic benefits specific to its lowering of capital and operational costs for application development and deployment. For starters, the managed quality of serverless deployments means that customers are not required to allocate capital expenses to develop or deploy applications. Moreover, serverless computing uses a utility pricing model that requires customers to pay for only the resources that they consume. Because functions are short lived in duration, customers typically pay a fraction of the cost of hosting a persistent application that features a plethora of dormant code.

» **Simplification of development experience.** Another key benefit of serverless computing is its simplification of the development experience. The requirement that developers write applications in terms of functions that execute in accord with the realization of an event means that developers can focus their development initiatives on the composition and management of discrete functions. This ability to circumscribe the scope of application development to discrete functions without worrying about overhead simplifies and streamlines both the initial and the ongoing development and deployment of applications.

» **Use of microservices-style application architectures.** Microservices allow an application to be built as a suite of loosely coupled independent components. Because serverless applications are built using a microservices architecture, developers concern themselves only with the individual functions that define the serverless application. This requirement for developers to focus on individual functions means that developers are far removed from trudging through a monolithic application, for either application design or updates. Serverless applications are microservices based by design and, in this respect, are representative of a thoroughly modern application architecture.

» **Acceleration of development.** Serverless computing accelerates software development by empowering developers to use functions, in conjunction with APIs, to rapidly build solutions to contemporary business problems. The speed with which developers can use serverless computing to build applications, and then iterate on them through user feedback, increases the likelihood that a serverless application will deliver minimally viable solutions that are embraced and refined.
Selecting a Serverless Platform

The serverless market landscape features a multitude of proprietary and open source serverless products. Choosing a serverless product from within this landscape can be challenging, particularly because vendors and organizations make very similar marketing claims about the capabilities of their products. Customers that are considering the selection of a serverless platform should take stock of the following factors:

» **Platform speed**: How fast is the platform? How often do cold starts occur, and what are their effects on application performance?

» **Open versus closed source**: Is the platform based on open standards, or is it proprietary and closed source?

» **Developer tooling**: Are the developer tools used to develop serverless applications intuitive and easy to use? How easy is it to debug applications and obtain visibility into application- or infrastructure-related issues? Can developers use containers to deploy functions or integrate continuous integration/continuous delivery (CI/CD) functionality into serverless applications?

» **User experience**: Does the platform's user experience simplify the experience of developing and running serverless functions? How easy is it to integrate the serverless platform into existing development workflows? How quickly can developers make changes to the deployment of a serverless application?

» **Pricing model**: How transparent is the pricing model of the serverless platform under consideration? Is it easy to understand the cost of the execution of serverless functions? Does the platform provide granular pricing units that ensure that customers pay for only what they use? Are there hidden charges related to variables such as bandwidth consumption?

» **Monitoring**: How easy is it to monitor the execution and progress of serverless functions? Does the platform have a native toolset for application performance monitoring? If not, does the platform integrate with respected third-party monitoring tools?

» **Security**: How easy is it to implement and manage security? What tools and analytics are available for implementing security?

Trends

The landscape of serverless computing features the following notable trends:

**Enhancements in Developer Tooling**

To date, serverless computing has offered limited visibility regarding what happens once a serverless function is executed. For example, if a serverless application does not perform as expected, developers have little in the way of access to monitoring tools that illustrate the underlying causes of an application malfunction. Because the infrastructure is fully managed and distributed, developers may not have direct access to it to determine whether infrastructure is responsible for an error in an application. The industry should expect the serverless landscape to make available more powerful debugging and performance analysis tools soon. Additionally, the serverless computing landscape is likely to deliver enhanced tooling to augment the ability of developers to aggregate functions such that serverless functions can reference other functions as necessary.
Increasing Standardization of Serverless Computing Infrastructures Around Knative Middleware

Since the launch of the first functions-as-a-service platform, the serverless landscape has featured a multitude of new entrants that have rendered the landscape increasingly heterogeneous. The open sourcing of Knative, a set of middleware components that empowers developers to build serverless applications on Kubernetes, promises to standardize serverless computing. Admittedly, Knative is still early in its life cycle, but the speed with which it has received affirmations from prominent public cloud providers suggests that standardized infrastructure for serverless computing may become a reality sooner rather than later. Increased standardization of serverless computing infrastructures is likely to accelerate adoption of serverless computing because standardization facilitates interoperability among serverless computing providers, thereby allaying customer concerns with respect to vendor lock-in.

Considering Cloudflare

Cloudflare offers performance and security solutions designed to make the internet faster and safer for customers. Based on its Anycast network technology, Cloudflare leverages a distributed architecture that features expansive network scalability, ease of use, integrated performance and security solutions, and transparent pricing. The company’s global footprint facilitates the optimization of content delivery by bringing content closer to customers while mitigating security risks and ensuring low latency and high availability.

Serverless computing represents a key component of Cloudflare’s underlying technology stack in the form of Cloudflare Workers, the company’s serverless computing platform. Cloudflare Workers empowers developers to run more of their application logic on the Cloudflare platform by using serverless computing to define events and their associated outcomes. Cloudflare Workers achieves this by intercepting HTTP traffic and responding to requests to handle business logic for which each individual Worker is responsible.

Use Cases

Developers can use Cloudflare Workers to deliver enhanced performance, security, and customization for websites, applications, and APIs on Cloudflare. Cloudflare Workers executes code in correspondence with requests made by end users. After receiving the request event, Cloudflare Workers can run custom JavaScript that either responds directly or makes subsequent requests to the origin server, Cloudflare Cache, or a third-party service.

Cloudflare Workers can make changes to responses based on request events from individual users based on session, location, cookies, headers, or other information. By making the changes on the global Cloudflare network, closer to the user, Cloudflare Workers can deliver enhanced customization to the user without sacrificing performance. For example, Workers can translate web pages based on user preferred language or change the currency displayed on an ecommerce website into a currency that is local to the geography of the website visitor.

Cloudflare Workers can also be used to deliver a secure experience for websites and applications without sacrificing performance. Developers can write serverless code to quickly verify authorization tokens or check for malicious inputs and easily alert, log, or act before requests reach the application origin. Writing security logic on the Cloudflare network also allows for better separation between the application’s core functionality and support code.

As the Cloudflare Workers platform matures, it will likely be possible to write even more complicated serverless applications, in addition to modifying and enhancing existing applications. Developers will be able to write applications that run at Cloudflare’s 150+ datacenters. This will open the door for any number of use cases, especially those requiring low latency responses anywhere in the world.
Technology
Cloudflare Workers is written in JavaScript (in addition to other languages such as C/C++/Rust by using WebAssembly) and leverages Google's V8 JavaScript engine to execute. The primary use of JavaScript democratizes Cloudflare Workers to developers given that most web developers are likely to know JavaScript. Moreover, the selection of the V8 JavaScript engine bolsters the security of Cloudflare Workers because of the intense attention paid to V8 JavaScript security by Google and other parties, in addition to the extra work done by Cloudflare. Another important attribute of Cloudflare Workers is that its architecture (where each Worker uses a sandboxed lightweight runtime) avoids the cold starts that are characteristic of most container-based serverless implementations.

For example, users do not experience latency delays when they implement Cloudflare Workers for the first time or during times of increasing load. That said, Cloudflare has used its Isolate architecture to take a network-based approach to serverless that has the potential to iteratively improve latency by intelligently routing traffic to Cloudflare Workers anywhere in the world. Figure 2 illustrates the life cycle of a Cloudflare Worker in response to HTTP traffic. The diagram elaborates on the positioning of a Cloudflare Worker with respect to incoming web traffic and an end-user device.

Figure 2: Life Cycle of a Cloudflare Worker

Source: Cloudflare, 2018
Network-Based Serverless with Isolates

A key differentiator of Cloudflare’s serverless implementation is how it uses the performance benefits of isolates to take a network-based serverless approach in contrast to a region-based serverless approach. Cloudflare’s network-based serverless approach delivers serverless computing by means of a global network that is not partitioned into cloud regions, as is common among leading cloud providers.

Cloudflare’s model of network-based serverless uses a global network marked by response times that are faster than those of a regional model. Moreover, its network-based serverless model is not susceptible to regional outages and absolves developers of the responsibility of global georeplication and load balancing. Flexible scaling represents another key advantage of a network-based serverless approach insofar as Cloudflare’s global network automates the scaling of compute, networking, and storage in response to customer demands.

Importantly, Cloudflare’s network-based serverless approach simplifies and enhances the developer experience by eliminating the need for developers to architect multiregion serverless implementations for high-availability or redundancy considerations. Architecting a multiregion implementation of serverless computing can be complex and requires explicit attention to the configuration of infrastructure across regions. As such, Cloudflare’s implementation of serverless computing represents a more developer-friendly implementation of serverless computing insofar as developers need not concern themselves with the platform’s underlying infrastructure; instead, they can focus on writing JavaScript (or C/C++/Rust using WebAssembly) code for Cloudflare Workers.

Challenges

Challenges that Cloudflare is likely to encounter include ensuring that its serverless technology keeps pace with a rapidly evolving landscape that features competing serverless platforms from vendors such as Amazon, Microsoft, Google, and Red Hat. For example, Cloudflare will need to ensure that the developer tooling for its serverless technology remains competitive in a technology landscape that is likely to witness significant advancements over the next few years. To that end, Cloudflare would do well to continue to enhance the ability of developers to obtain visibility into the performance of their serverless applications. Because the developer tooling for debugging and managing the performance of serverless applications is still in its infancy across the industry, Cloudflare can not only obtain parity with its competition but also lead and innovate by enhancing its serverless application performance management capabilities.

Additionally, Cloudflare’s serverless technology would benefit from delivering polyglot development functionality for its serverless technology platform that supplements the ability of developers to build serverless applications in JavaScript with other languages. In addition, Cloudflare should consider incorporating additional integrated serverless storage functionality (e.g., object or document storage) beyond the key-value store that is already supported for its Cloudflare Workers platform.

Conclusion

Serverless computing is widely regarded as exemplary of the future of modern application development because of its abstraction of IT infrastructure, acceleration of the development experience, reduction of application complexity, and ability to lower development costs. The lack of a requirement to manage infrastructure means that developers can focus on designing and building applications while offloading tasks such as application deployment and the ongoing operational management of infrastructure to a third-party cloud provider. Cloudflare’s use of serverless technologies, for use cases such as the personalization of web-based user experience and the implementation of security, pushes business logic onto
the Cloudflare network and subsequently reduces the operational footprint of these applications, leading to greater application responsiveness and reduced infrastructure costs.

Importantly, Cloudflare uses its Isolate architecture to deliver a network-based model of serverless computing that differs from a region-based serverless delivery model by leveraging a global network to route requests to its serverless compute platform. Instead of routing serverless computing–related traffic to a designated set of cloud datacenters, Cloudflare uses a distributed network to implement serverless computing by means of its Cloudflare Workers, which intercepts and transforms HTTP traffic to deliver the correct cached version of an application and content to end users. Cloudflare’s network-based serverless computing model simplifies the development experience by eliminating the need for the multiregion configuration of serverless computing employed by region-based serverless deployments. Furthermore, its network-based serverless model boasts immunity from regional outages, faster response times, and flexible scaling in response to global network supply and demand. As such, Cloudflare’s implementation of serverless computing represents an innovative deployment model for serverless computing that claims a multitude of advantages compared with region-based serverless computing deployment and delivery.
MESSAGE FROM THE SPONSOR

Begin Building Serverless Applications on Cloudflare

Build serverless applications on Cloudflare’s global cloud network of 150+ datacenters. Cloudflare Workers provides a lightweight JavaScript execution environment that allows developers to augment existing applications or create entirely new ones without configuring or maintaining infrastructure.

» Sign up for Cloudflare Workers by visiting: https://cfl.re/IDC-Workers-Signup
» Interested in exploring serverless for your business? Get in touch: https://cfl.re/IDC-Cloudflare-Serverless

About the analyst:

Arnal Dayaratna, Research Director, Software Development

Dr. Arnal Dayaratna is Research Director, Software Development, at IDC. Dr. Dayaratna focuses on software developer demographics, modalities of software development, trends in programming languages and other application development tools, and the intersection of these development environments and the many emerging technologies that are enabling and driving digital transformation.