DOI:10.1145/3587249

Dispelling the confusion around serverless computing by capturing its essential and conceptual characteristics.

BY SAMUEL KOUNEV, NIKOLAS HERBST, CRISTINA L. ABAD, ALEXANDRU IOSUP, IAN FOSTER, PRASHANT SHENOY, OMER RANA, AND ANDREW A. CHIEN

Serverless Computing: What It Is, and What It Is Not?

FULL AUTOMATION OF IT infrastructure and the delivery of efficient IT operations as billed services have been long-standing goals of the computing industry since at least the 1960s. A newcomer—serverless computing—emerged in the late 2010s with characteristics claimed to be different from those of established IT services, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) clouds. Even though serverless computing has gained significant attention in industry and academia over the past five years, there is still no consensus about its unique distinguishing characteristics and precise understanding of how these characteristics differ from classical cloud computing.

What is serverless computing, and what are its implications?

Market analysts are agreed that serverless computing has strong market potential, with projected compound annual growth rates (CAGRs) varying between 21% and 28% through 2028[4,25,33,35,49] and a projected market value of $36.8 billion[49] by that time. Early adopters are attracted by expected cost reductions (47%), reduced operation effort (34%), and scalability (34%).[17] In research, the number of peer-reviewed publications connected to serverless computing has risen steadily since 2017.[46] In industry, the term is heavily used in cloud provider advertisements and even in the naming of specific products or services.

Yet despite this enthusiasm, there exists no common and precise understanding of what serverless is (and of what it is not). Indeed, existing definitions of serverless computing are largely inconsistent and unspecific, which leads to confusion in the use of not only this term but also related terms such as cloud computing, cloud-native, Container-as-a-Service (CaaS), Platform-as-a-Service (PaaS), Function-as-a-Service (FaaS), and Backend-as-a-Service (BaaS).[12]

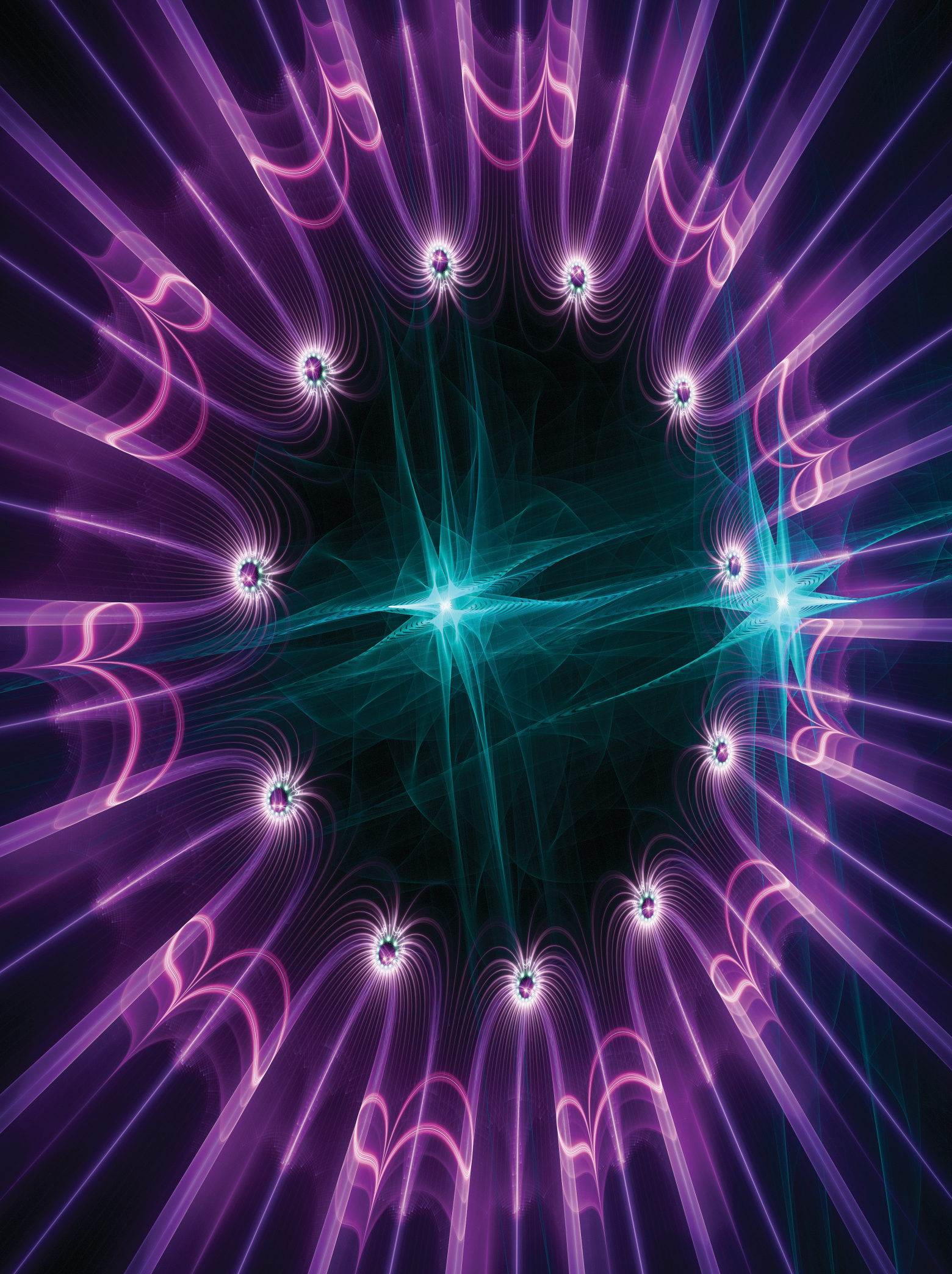
As an extended discussion during a 2021 Dagstuhl Seminar[2] and our analysis of existing definitions of serverless computing reveal, current definitions focus on a variety of aspects, from abstractions to practical concerns, from computational to financial, from separation of concerns to how concerns should be enacted, and so on.

These definitions do not provide consensus, and they are omissive in essential points or even diverge. For example, they do not agree on whether serverless is solely a set of require-

» key insights

- **Serverless computing means full automation and fine-grained utilization-based billing.**
- **Serverless computing has a well-defined and unique place in computing history.**
- **Serverless computing supports diverse applications, from enterprise automation to scientific computing.**

IMAGE BY SIRIPONG JITCHUM



ments from the user's perspective or it should also mandate specific implementation choices on the provider side, such as implementing an autoscaling mechanism to achieve elasticity. Similarly, they do not agree on whether serverless is just the operational part, or it should also include a specific programming model, interface, or calling protocol. These and related aspects make serverless computing an interesting object of study for academics, complementing the economic and industrial interest, but an object whose current definition is fraught with confusion.

In this article, we seek to dispel this confusion among others by proposing a refined definition capturing the essential conceptual characteristics of serverless computing as a paradigm, while putting the various terms around it into perspective. We start by providing an analogy to intuitively illustrate serverless computing and how it compares to classical cloud computing. We then examine how the term serverless computing, and related terms, are used today. We explain the historical evolution leading to serverless computing, starting with mainframe virtualization in the 1960s through to grid and cloud computing all the way up to today. We review existing cloud computing service models (including IaaS, PaaS, SaaS, CaaS, FaaS, and BaaS), and for each of them, we discuss which aspects determine if a given platform can be considered as serverless computing or not. Finally, we review serverless computing applications, discuss open challenges, and provide an outlook on the future of the serverless paradigm.

Our goal is to help improve communication among researchers and practitioners, reducing confusion and misunderstandings due to the lack of understanding of the underlying concepts and their historical evolution. Conceptual understanding of the state of the art coupled with clear and consistent terminology provide a basis for supporting interoperability between emerging platforms as well as for future research driving the further advancement of the field.

Figure 1. Analogy between IT services and moving homes.

TECHNICAL BOX

Serverless through an Analogy: Services for Moving Homes

We explain serverless computing via an analogy. Moving homes is a familiar, if not always welcome, task in our lives. Household items must be packaged, packages delivered to their destination, and operational and even legal issues resolved. These demanding and often complex tasks, like computing, offer many opportunities for outsourcing and specialization. As illustrated in Figure 1, we group these opportunities into three broad classes.

One approach to moving homes, representing self-hosting in our analogy, is to move everything independently. Many students take this approach. Here, we must disassemble and package all objects, load them into the car, plan the route, and drive the whole way.

There may be borders on the way, in which case we must do paperwork. This approach gives us maximal flexibility and control, but as it costs considerable resources in planning and executing, it is inefficient for one-time movers. The inefficiency arises from not exploiting the many opportunities for process learning and economies of scale.

A second approach, which represents classical cloud computing (IaaS/PaaS) in our analogy, is to hire a traditional moving company. They specialize in loading and driving to the destination cheaply. They are efficient in what they do but leave little room for customization and defer all operational details other than packing and driving the van to the client. They have several kinds of crates, but we must disassemble our furniture, pack it into crates, and unpack and rebuild it at the destination. We pay by crate size, regardless of its actual occupancy. Odd-sized objects are not allowed, and it is our responsibility to ensure grandma's precious mirror stays intact. Individual operations do not appear in the final bill. Furthermore, the processes happen relatively slowly, and changes or additional requests can take days to be acknowledged. Bottom line: We can move cheaply and retain some control, but we do not receive a detailed bill, pay for all operations regardless of their usefulness, and are responsible for everything but loading, relocating, and unloading crates.

Serverless computing is commonly understood as an approach to developing and running cloud applications without requiring server management.

The third and simplest approach to moving homes, which we suggest is akin to serverless computing, is to hire professional movers. The movers know how to plan and transport loads. They handle legal paperwork and take responsibility if objects break. They have a broad collection of appropriately sized boxes, which they select on our behalf, and they pack objects into those boxes efficiently and safely. They know about various kinds of furniture, so there is no risk that they may break them apart instead of merely disassembling them, or that they may not be able to put them back together. They handle fragile objects with care and sign off on expensive antiques. They can configure Internet and cable in the new location. They know plants require air and water. All these operations are recorded and appear explicitly in the final bill. They perform these processes rapidly and are responsive to additional requests and changes. Moreover, the movers can pack several jobs together, leading to important economies of scale without breaking things. We give up control but receive better service with less effort.

The Many Definitions of Serverless

Serverless computing is commonly understood as an approach to developing and running cloud applications without requiring server management.[12] The term became popular after Amazon Web Services (AWS) introduced AWS Lambda in 2014.[1] Since then, a number of serverless computing platforms have appeared in industry.

Serverless computing, however, still lacks a precise definition. To underline this observation, we show in Figure 2 similarities and differences among six oft-cited definitions selected as representative examples. While there are further definitions from different communities,[30,32] they share some similar elements and characteristics as the definitions we show. On the one hand, the definitions differ in their scope and level of detail; on the other hand, they appear to make some inconsistent assumptions, which may leave the impression of contradictory viewpoints. We see that while certain characteristics recur, only two are to be found in all six existing definitions, and even then, with different emphasis. Other characteris-

tics that appear as fundamental to some definitions do not occur at all in others (for example, autoscaling/elasticity, which is not mentioned in Definitions 2 and 3). In the following, we take a closer look at the recurring concepts to extract the essential features that distinguish serverless computing from classical cloud computing.

We start by noting that the term "serverless" does not imply that no servers are used; it rather refers to a key characteristic of serverless computing—that cloud application developers need not concern themselves with managing and operating servers: a feature sometimes referred to as NoOps (for "no operations")[8,29] or Zero Server Ops.[12] Although this may seem intuitive, many definitions describe this feature in rather imprecise and potentially confusing ways. For example, the phrasing "developers need not concern themselves with provisioning or operating servers" (Definition 4) could naturally be interpreted as referring to the management of datacenter infrastructure (that is, the physical servers including hardware, operating systems, and storage).

Thus, one may ask: How is this differ-

Figure 2. Popular definitions of serverless computing (color-coding highlights wording related to different characteristics, citation counts obtained in January 2023 via Google Scholar).

1. Baldini, et al., Serverless Computing: Current Trends and Open Problems (667 cit.), Springer, Dec 2017 [8]:

> "Serverless computing is a term coined by industry to describe a programming model and architecture where small code snippets are executed in the cloud without any control over the resources on which the code runs… the developer has control over the code they deploy into the Cloud, though that code has to be written in the form of stateless functions. The developer does not worry about the operational aspects of deployment and maintenance of that code and expects it to be fault-tolerant and auto-scaling. In particular, the code may be scaled to zero where no servers are actually running when the user's function code is not used, and there is no cost to the user… The version of serverless that explicitly uses functions as the deployment unit is also called Function-as-a-Service (FaaS)."

2. Varghese and Buyya, Next Generation Cloud Computing: New Trends and Research Directions (731 cit.), Elsevier FGCS, Feb 2018 [48]; see also Buyya et al., A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade (294 cit.), ACM CSUR, Nov 2018 [9]:

> "Serverless…simply means that a server is not rented as a conventional cloud server and developers do not think of the server and the residency of applications on a cloud. From a developers' perspective challenges such as the deployment of an application on a VM, over/under provisioning of resources for the application, scalability and fault tolerance do not need to be dealt with… In this novel approach, functions (modules) of the application will be executed when necessary without requiring the application to be running all the time. Sometimes this is also referred to as Function-as-a-Service or event-based programming."

3. van Eyk et al., Serverless is More: From PaaS to Present Cloud Computing (135 cit.), IEEE IC, May 2018 [47]:

> "Serverless Computing is a form of cloud computing which allows users to run event-driven and granularly billed applications, without having to address the operational logic. Function-as-a-Service (FaaS) is a form of serverless computing where the cloud provider manages the resources, lifecycle, and event-driven execution of user-provided functions."

4. Hellerstein et al., Serverless Computing: One Step Forward, Two Steps Back (360 cit.). CIDR, Jan 2019 [23]:

> "Serverless computing offers the attractive notion of a platform in the cloud where developers simply upload their code, and the platform executes it on their behalf as needed at any scale. Developers need not concern themselves with provisioning or operating servers, and they pay only for the compute resources used when their code is invoked... Serverless is not only FaaS. It is FaaS supported by a "standard library": the various multi-tenanted, autoscaling services provided by the vendor. In the case of AWS, this includes S3 (large object storage), DynamoDB (key-value storage), SQS (queuing services), SNS (notification services), and more."

5. Castro et al., The Rise of Serverless Computing (196 cit.), CACM, Dec 2019 [10]:

> "Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running. This definition captures the two key features of serverless computing: (a) Cost—billed only for what is running (pay-as-you-go)...; serverless essentially supports "scaling to zero" and avoids the need to pay for idle servers. (b) Elasticity—scaling from zero to "infinity." … The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests. Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing."

6. Jonas,…, Patterson et al., Cloud Programming Simplified: A Berkeley View on Serverless Computing (485 cit.), arXiv, Feb 2019 [26], refined in a follow up publication by the same authors What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing (75 cit.), CACM, May 2021 [41]

> "In serverless computing, programmers create applications using high level abstractions offered by the cloud provider... They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as Backend-as-a-Service (BaaS). Managed cloud function services are also called Function-as-a-Service (FaaS) and collectively Serverless Cloud Computing today = FaaS + BaaS. Three essential qualities of serverless computing are: 1. Providing an abstraction that hides the servers and the complexity of programming and operating them. 2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources. 3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite."

Legend: 6x NoOps 6x Function-as-a-Service 5x pay-per-use 4x autoscaling/elasticity 3x Backend-as-a-Service 2x event-driven arch.

ent from what classical cloud computing is about? The answer is that, when speaking of not having to provision and operate servers, one means virtualized servers (not physical servers) and their operational aspects, such as fault tolerance or autoscaling. More specifically, the NoOps property refers to the abstraction of the complexity of the cloud execution environment (virtual machines, containers, or operating systems) and associated operational concerns, such as resource management, container/instance life cycle, elasticity/autoscaling, fault-tolerance, security, system monitoring and accounting, among others.[2,8,12] This is in contrast to classical cloud computing, as adopted by the market, which has mostly been focused on IaaS clouds, where cloud users must explicitly allocate (that is, lease, reserve) resources, such as virtual machines or containers, and are then responsible for their management and operation.[6] We note that under NoOps, the business logic part of the application, including function and workflow composition, remains a task of the application developer except for very restricted frameworks such as MapReduce.[47]

The second fundamental characteristic most definitions mention is what they refer to as the "pay-as-you-go" cost/billing model (for example, see Definitions 5 and 6).[29] This characteristic is related to, but distinct from, the NoOps property: Given that in serverless computing users do not explicitly allocate and release resources (which are indeed hidden from them), it only makes sense to charge based on the time that their applications are executing and actively consuming resources rather than, as in classical cloud platforms (for example, Amazon EC2), based on what resources a user has allocated, even if currently idle.[32,41]

Again, the wording used in most definitions is confusing. For one, the terms "pay-as-you-go" (Definitions 5 and 6) or "pay-per-use" have long been used to describe classical reservation-based cost models.[6,9,32] When speaking of being "billed only for what is running (pay-as-you-go)" (Definition 5) or "pay only for the compute resources used when their code is invoked" (Definition 4), one may again not see the difference here from classical cloud computing where similar wording is used; for example, the authors of the 2009 highly cited Berkeley definition of cloud computing[6] also use the term "pay-as-you-go" and define it as "the ability to pay for use of computing resources on a short-term basis as needed" while also stating that "it involves metering usage and charging based on actual use" and explicitly mentioning AWS as "a true pay-as-you-go service." While some definitions (for example, Definition 6) explicitly differentiate from classical reservation-based cost models, the formulations used in most definitions are rather imprecise, which may lead to confusion as further described in the following.

Indeed, another source of ambiguity is that phrases like "pay only for the compute resources used when their code is invoked" (Definition 4), "billed only for the time the code is running" (Definition 5), and "event-driven and granularly billed applications" (Definition 3) leave much room for interpretation; for example, does the bill include the launching of the environment (container, runtime) where the application code is to execute? Is an event-driven application model generally assumed (as suggested by Definitions 2 and 3 but not by others)? What about the storage space used by applications? Data stored in a database or in a message queue consumes space even when no application code is running.

Related to these observations is the fact that definitions differ somewhat in how they describe the purpose and scope of serverless platforms. All definitions speak explicitly of FaaS as the most common form of serverless computing since the introduction of AWS Lambda in 2014.[1] Definition 2 is particularly rigid in its assumption that serverless = FaaS, while Definition 4 states explicitly that serverless is not only FaaS but rather "FaaS supported by a standard library." BaaS is only mentioned explicitly in Definitions 5 and 6; however, while the former considers BaaS as a separate category bearing close resemblance to serverless computing,[10] the latter explicitly includes BaaS as part of the serverless paradigm in addition to FaaS.[26,41] While not mentioned explicitly in Definitions 1 and 5, their authors consider the boundaries defining serverless computing to also overlap with classical terms such as SaaS and PaaS.[8,10] Further, some definitions (in particular, Definitions 2 and 3 but also the one provided in Li et al.[30] and Mampage et al.[32]) assume an event-driven application architecture. While this fits classical FaaS platforms, an event-driven architecture appears to not be fundamental to serverless computing.[10,26,41] Overall, the purpose, scope, and programming model seem to be areas of uncertainty when it comes to serverless computing definitions.

Finally, four of the definitions mention elasticity (or automatic scaling, also referred to as autoscaling) as a defining characteristic of serverless computing. This, again, is potentially confusing since the term elasticity has been listed as an essential characteristic of cloud computing from the beginning[6,9,24,34] with, for example, the NIST Definition of Cloud Computing[34] speaking of rapid elasticity defined as "capabilities [that] can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand." Google App Engine, one of the earliest PaaS cloud offerings, offered automatic scaling from the early days of cloud computing.[6] A highly cited paper from 2013 defined elasticity as "the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible."[24] Even back in 2009, Armbrust et al.[6] spoke of elasticity as the "ability to add or remove resources at a fine grain ... and with a lead time of minutes rather than weeks allowing one to match resources to workload much more closely." Thus, when speaking (in the context of serverless computing) of "elasticity—scaling from zero to infinity" (Definition 5) or of "automatic, rapid, and unlimited scaling resources up and down to match demand closely, from

...as-a-Service

BaaS = Backend-as-a-Service
CaaS = Container-as-a-Service
FaaS = Function-as-a-Service
IaaS = Infrastructure-as-a-Service
PaaS = Platform-as-a-Service
SaaS = Software-as-a-Service

zero to practically infinite" (Definition 6), one may again be confused, as similar wording has been used in the cloud literature for more than a decade. Indeed, Jonas et al.[26] admit that "without a quantitative and broadly accepted technical definition or metric—something that could aid in comparing or composing systems—elastic will remain an ambiguous descriptor." The idea of automatic scaling is not new, although classical cloud platforms (IaaS, PaaS) provided limited support for it and implementing autoscaling has been a complex task commonly left for the cloud user to configure and manage.[9,10,41]

In our view, the essential point about elasticity in serverless computing is that the responsibility for it is entirely offloaded to the cloud provider, leaving the developer free from having to define autoscaling rules or configure orchestration frameworks to implement autoscaling.[26,32,47] However, this feature is already captured as part of the NoOps property, as elasticity is a classical operations task. Therefore, we argue that details about autoscaling/elasticity should not be part of the definition because it is just one technical aspect, which is not even used by some providers, either due to their technological choices or because their users do not need it.

The preceding analysis leads us to conclude that existing definitions of serverless computing fail to capture the central aspects of this new technology in a clear, unambiguous, and consistent manner.

Understanding the Essence of Serverless Computing

The question of just what is serverless computing and how it differs from classical cloud computing was discussed extensively at a Dagstuhl Seminar we organized in 2021,[2] bringing together around 50 experts from academia and industry, representing three communities of experts in computer systems, software engineering, and performance engineering. The discussions at the seminar sparked an effort to provide a new refined definition of serverless computing coupled with a long-term perspective on how the serverless paradigm fits in the space of existing and emerging cloud computing service models. The initial results of this effort based on discussions at the seminar were included in the seminar report[29] but have not been published in a peer-reviewed publication so far. The effort was continued after the seminar and eventually led to our refined definition and perspective that we present in this article.

Our Refined Definition of Serverless

Serverless computing is a cloud computing paradigm encompassing a class of cloud computing platforms that allow one to develop, deploy, and run applications (or components thereof) in the cloud without allocating and managing virtualized servers and resources or being concerned about other operational aspects. The responsibility for operational aspects, such as fault tolerance or the elastic scaling of computing, storage, and communication resources to match varying application demands, is offloaded to the cloud provider. Providers apply utilization-based billing: they charge cloud users with fine granularity, in proportion to the resources that applications actually consume from the cloud infrastructure, such as computing time, memory, and storage space.

Our refined definition of serverless is shown in the sidebar here. In developing this definition, we sought to strike a balance between generality, to cover the serverless technologies of today—and, hopefully, also the future—and *concreteness*, to make clear how the serverless paradigm differs from classical cloud computing. The resulting definition, we believe, is formulated at a level of abstraction that can remain valid as novel serverless platforms continue to emerge in the next decade and beyond.

Note that our definition avoids terms such as "pay-per-use," "pay-as-you-go," or "infinite/rapid elasticity." As was noted earlier, such terms have been used from the beginning of cloud computing and thus have little differentiating power for the new serverless paradigm. Furthermore, the meaning assigned to those terms has varied significantly in the past years, often causing confusion.[24]

To address this issue and the other concerns about the ambiguity of the wording used in existing definitions (expressed previously), we propose the alternative term "utilization-based billing." We believe the specific wording used in the definition to describe this term better captures the full range of current serverless billing models while also providing flexibility to accommodate other models that may emerge in the future.[48] Computing time, memory, and storage space are mentioned as examples of possible resources; however, novel billing models could be based on any form of computing, memory, storage, or networking resources for both quantity and quality (for example, speed).[32] For instance, Google Cloud Functions charges for cloud speed in GHz-seconds, where the clock speed and the number of vCPUs allocated to a function are scaled relative to a function's memory.

Finally, based on the previous discussion, no specific requirements about elasticity/autoscaling are included in the definition; elastic scaling is only mentioned as an example of a major operational aspect that in serverless computing is delegated to the cloud provider. Indeed, under the assumption of NoOps and utilization-based billing, the user should not care about how the cloud provider manages the infrastructure internally, including details about the level and granularity of autoscaling used to achieve elasticity.[32] Theoretically speaking, even if the cloud provider would apply a brute force approach heavily overprovisioning resources to avoid ever needing to scale, under utilization-based billing, the user would not be concerned given that no costs are incurred for overprovisioned resources when they are not actively used.

The many faces of serverless: Cloud service models and their relation to serverless. *FaaS* can be seen as the most prominent example of serverless computing nowadays; one way to define it is as "a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests."[10] Current FaaS platforms have a relatively narrow scope, focusing on small, stateless, and event-driven functions. Those assumptions make it easy for FaaS cloud providers to implement autoscaling in a generic manner and to provide a fine-

granular utilization-based cost model that bills customers based on the actual time functions are running.[8,26] We expect the current assumptions of the FaaS model (small, stateless, and event-driven units of computation) to eventually be relaxed as platforms evolve to support a wider set of applications.[29,50]

BaaS. Our broad definition of serverless computing includes modern BaaS offerings, which are focused on specialized cloud application components, such as object storage, databases, and messaging.[26] Examples of BaaS offerings include AWS' Simple Storage Service (object storage) and DynamoDB (key-value database) or Google' Cloud Firestore (NoSQL document database) and Cloud Pub/Sub (publish/subscribe messaging middleware).

CaaS is a cloud service model that allows users to deploy and manage containers in the cloud.[48] A container can be seen as a light-weight execution environment, typically run inside a VM on a server in the cloud infrastructure.[9] Whether a CaaS platform can be considered as serverless or not depends on the level of abstraction and automation it provides. Examples of CaaS platforms include Amazon Elastic Container Service (AWS ECS), Google Kubernetes Engine (GKE), and Azure Container Instances (ACI). While these platforms can be configured to use container orchestration services taking care of container management and operational tasks, they often do not completely abstract the underlying server layers such as VMs and operating systems.[12] Thus, they cannot be considered as being fully serverless. In recent years, some serverless CaaS platforms have emerged including Google Cloud Run, AWS Fargate, and Azure Container Apps.

PaaS, a concept realized in platforms such as Cloud Foundry, Heroku, and Google App Engine, was originally defined by NIST as "the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations."[34] Thus, the classical PaaS definition neither requires not forbids application developers having control over the deployment and configuration of the hosting environment. Consequently, whether a PaaS can be considered as serverless depends on the specific abstractions and automation that it provides to application developers. Classical PaaS offerings like early versions of Microsoft Azure had serverless elements but did not completely abstract servers and operational aspects, and they therefore cannot be considered as being fully serverless. Others like Google App Engine, specialized for Web applications, were close to the serverless paradigm from the early days of cloud computing, and they quickly evolved into serverless PaaS offerings.[6]

We argue that details about autoscaling/elasticity should not be part of the definition of serverless computing because it is just one technical aspect, which is not even used by some providers.

SaaS refers to the end-user applications deployed in a cloud platform and delivered as services over the Internet.[6,34] Therefore, although SaaS abstract the cloud execution environment, strictly speaking, the term serverless is not applicable here since it describes characteristics of the cloud platform as opposed to the deployed applications running on it. On the other hand, some SaaS offerings support the execution of user-provided functions tightly coupled to a specific application domain. Such offerings can be seen as specialized forms of serverless computing platforms, such as the Google Workspace Marketplace in Google Workspace.[10]

In *IaaS*—the classic and most widespread cloud service model—the cloud user typically manages virtualized servers and resources provisioned by the cloud provider; the user is assumed to have control over operating systems, storage, and possibly network components.[34] Thus, by definition, IaaS platforms are not serverless.

Boundaries of serverless computing. In our preliminary definition,[29] we introduced serverless computing as "... a cloud computing paradigm offering a high-level application-programming model ..." In our refined definition presented here, we slightly changed the formulation, introducing serverless as "... a cloud computing paradigm encompassing a class of cloud computing platforms ..." We believe this formulation better captures the evolution and diversity of serverless offerings that emerged in the past decade since 2014.

Indeed, modern serverless platforms do not necessarily lock developers into a specific application programming model. On the one hand, the programming model itself often comes from the entire cloud platform (for example, AWS, Google Cloud Platform, or Microsoft Azure), as in practice, there are typically many bindings between serverless components (for example, FaaS functions) and a diverse set of vendor specific cloud services, some of which may not be serverless. On the other hand, FaaS platforms, the most popular form of serverless computing, are nowadays becoming increasingly diverse and open; instead of prescribing a specific application programming model, they offer an agile infrastructure that can be managed and scaled dynamically for any task. For example, platforms like AWS Lambda support the deployment of "custom runtime" functions that can be written in any language as well as read-only container images allowing developers to deploy libraries and code in any Linux compatible language or tool. Similarly, Google Cloud Run supports deploying and autoscaling containerized applications developed using any programming language or operating system libraries with the possibility to even deploy the user's own binaries. As serverless computing is further adopted by cloud providers, we expect that serverless offerings will continue to become increasingly diverse and open with the boundaries between different cloud service models increasingly diminishing.[8,50]

In summary, the serverless ecosystem includes a growing set of technologies and evolving service models (for example, FaaS, BaaS, some CaaS/PaaS/SaaS). Serverless computing is a high-level, broadly applicable term, which can be applied at many levels, including functions, containers, middleware, and backend services, as we discuss later. But it also refers to a specific technological evolution, which is the transition of cloud computing, as used and adopted by the market, to its second phase characterized by a shift of focus from the use of low-level VM-based interfaces, where VMs are managed by the cloud user, to high-level application-oriented interfaces, where servers are abstracted and managed by the provider.[41]

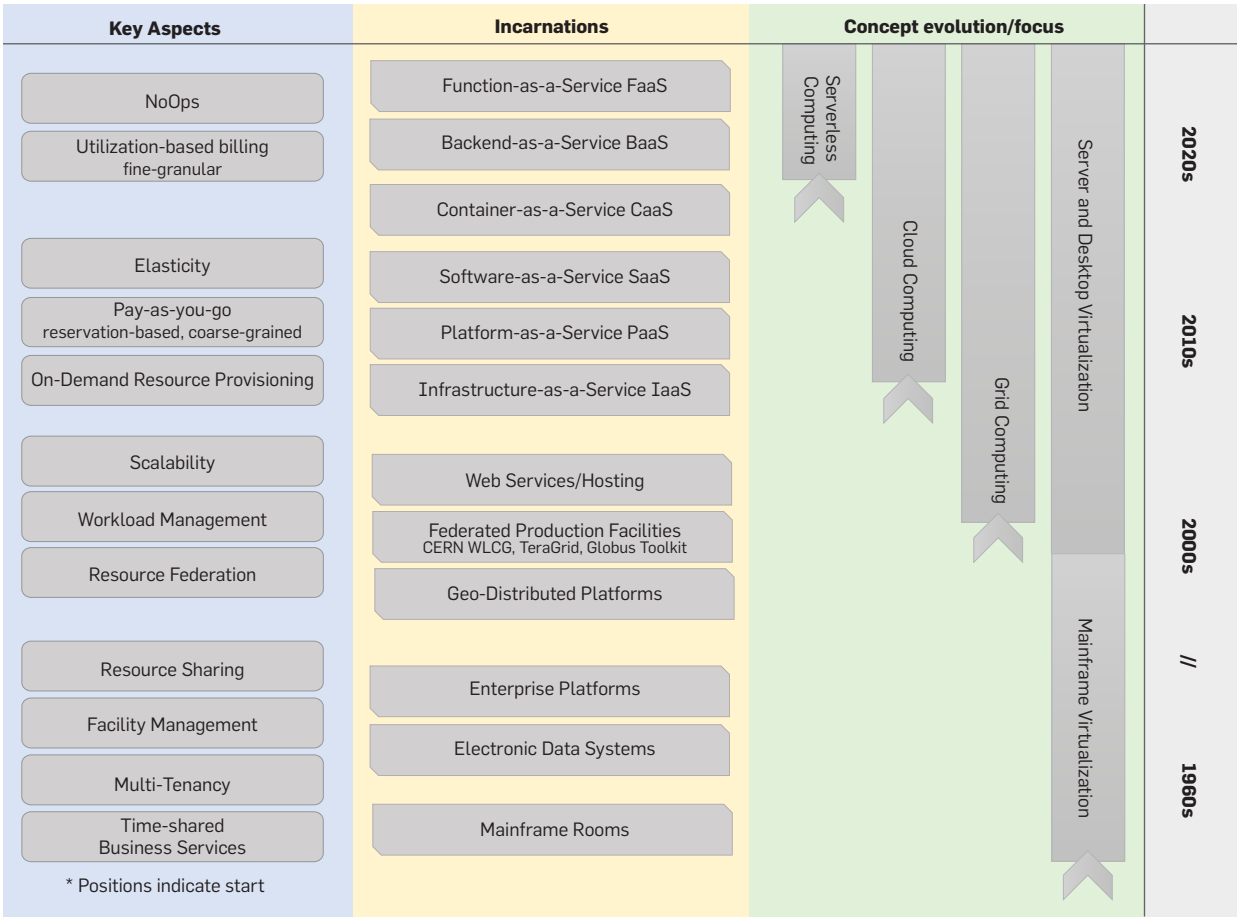
TECHNICAL BOX

Historical Perspective, Concept Evolution, Key Aspects, and Their Incarnations

Serverless computing is the latest step in a progression of computing utilities. Figure 3 depicts important steps in this technological evolution.

Computing platforms in the 1950s introduced computing technology with the IBM 701 leased for use. Others include the USA NSA ROGUE and ROB

Figure 3. Sixty years of technological evolution toward serverless computing.



ROY, and the U.S. Department of Defense's Whirlwind/SAGE. By the 1960s, computers were used for business, science, and other societal applications.

JCR Licklider at ARPA/IPTO proposed interconnecting computing centers for public collaboration (in 1963[31]), Martin Greenberger at MIT made the case for computing as utility (in 1964[22,a]), and Douglas Parkhill proposed computing as home-utility (in 1966[36]). These ideas converged, realized in centralized mainframes providing time-shared business services, multi-tenancy, and mainframe virtualization (for example, CTSS, DTSS, and PLATO in academia, and the IBM STRETCH). At the end of the 1960s, the ARPANET laid the foundation for networked communication, and pioneer Leonard Kleinrock predicted "as [computer networks] grow up and become more sophisticated, we will probably see the spread of 'computer utilities' which, like present electric and telephone utilities, will service individual homes and offices across the country."[27]

In the 1970s and 1980s computers became dramatically cheaper, smaller and as a result exploded in number (minicomputers, PCs). The obvious need was networking. TCP/IP emerged in 1983, enabling applications such as FTP and email (an independently developed extension of earlier ideas, for example, of 1960s messaging and 1970s PLATO mailing[14]) and forming the basis for today's Internet and Web applications.

The 1980s saw a new dynamic of resource integration and sharing in large-scale computing, which would become fruitful through the 1990s, 2000s, and beyond. Digital data representation unified information processing and communications technologies. Local networks enabled systems such as Condor and Utopia—automated approaches for workload management and resource sharing.

With the wide deployment of high-speed networks in the 1990s, academia started to play again a seminal role. U.S. gigabit testbeds that integrated resources at multiple sites spurred pioneering meta-computing approaches[45] in which resources were federated to enable new applications. Large-scale experiments such as the IWAY, which in 1994 built a geo-distributed software platform across USA to showcase support for over 50 research groups and application types,[19] enabled testing various designs for workload management and resource federation at scale. The Globus Toolkit provided a much-used reference architecture and implementation. These ideas resulted in federated, production-grade grid computing facilities for science, such as the massive, global-scale WLCG used primarily by CERN physicists and the cross-disciplinary USA TeraGrid. Concurrently, commercial services rapidly improved in availability, performance, and diversity (for example, AOL and CompuServe in the U.S., Minitel in France, building on Gopher and BBS technologies for information search and sharing).

In the late 2000s, cloud computing emerged as on-demand, elastic resource provisioning coupled with a convenient payment model. These offerings quickly diversified to provide several levels of computing abstraction—machine level (IaaS), middleware level (PaaS), and application level (SaaS), which enabled cloud applications of extraordinary complexity and scale.[6]

The success of these technologies set the stage for even more flexible and capable use of computing services—lower complexity, incremental cost (pay-as-you-go), and the latter coupled with elastic scaling. CaaS emerged to offer a finer-grained, more lightweight alternative to IaaS' VM-based virtualization.[48] Similarly, BaaS and FaaS emerged as back-end and front-end services in between PaaS and SaaS.[41] These opportunities and technological incarnations, and a shift in software development processes (toward DevOps), are essential progress vectors that led to the emergence of serverless computing.

Serverless computing is a high-level, broadly applicable term, which can be applied at many levels, including functions, containers, middleware, and backend services.

Outlook and Future Challenges

Serverless computing has emerged as an active area of research with ongoing projects tackling a range of topics to address open challenges in the field.

Performance challenges in serverless computing include cold-start latencies and autoscaling, next to emerging as-

a "Computing services and establishments will begin to spread throughout every life-sector ... medical-information systems, ... centralized traffic control, ... catalogue shopping from ... home, ... integrated management-control systems for companies and factories." Greenberg.[22]

pects such as workflow and dataflow management, and rethinking resource management techniques for fine-grained utilization-based billing. Cold-start costs are incurred whenever resources are first allocated for some purpose; autoscaling is required if workloads vary in size over time. If demand drops to zero, a serverless computing platform must choose between maintaining idle application resources (with associated memory costs) or scaling them to zero (incurring cold-start costs). For serverless applications with time-varying workloads, proactive autoscaling approaches are needed to eliminate scale-up latency in the face of load spikes. Right-sizing capacity allocations to the incoming workloads continues to be a challenge.[16] The cold-start problem is being tackled by approaches like letting users pay for reserved functions, reusing and snapshotting techniques[42] that can be combined with sticky routing approaches to encourage container reuse,[3,7] and enhanced caching approaches.[20] The autoscaling problem is being worked at multiple layers like increasing the worker pool size for the full platform or for a specific application,[43] and also scaling data-storage and providing serverless support for different storage technologies,[28] caches,[39] and databases.[38]

Security is another important issue in serverless computing. Since serverless applications from multiple third-party users execute on a shared platform, it is essential to provide isolation across computations from independent users. In scenarios where the application comprises a chain of functions that can invoke one another, it is important to ensure security by preventing unauthorized invocations of functions from third-party code. Because serverless ecosystems use services that break traditional security enclaves, solutions could consider dynamic levels of trust and some solutions could adopt zero-trust computing. Approaches to improve serverless security include global policies enforced by providers,[5] information flow tracking and control,[13] and novel access control models.[40]

From a *programming* standpoint, new paradigms and tools are being developed to simplify the implementation of serverless applications. So too are tools to support the migration of existing applications to the serverless model,

The question of whether a concrete platform is serverless or not may not always have a binary—yes or no—answer.

which often requires refactoring existing code into smaller modules or functions that can be deployed using the serverless paradigm. Tools to support such decomposition can significantly reduce the effort needed to port current applications. Approaches in this area include adopting familiar parallel programming models[51] and proposing new domain-specific languages.[37]

For cloud *hardware architectures,* just as IaaS spawned a variety of hardware virtualization features, serverless presents new challenges and opportunities for cloud hardware architectures. Short-running serverless invocations present challenges for CPU and especially accelerator architectures tuned for long runs by single applications. Fine-grained resource accounting presents challenges for performance variability—a difficult problem if it shows through into billed costs. Large working sets and cold start pose new performance optimization challenges for memory and storage hierarchies, and they introduce new avenues for cross-application performance interference. While there has been some prior work in trying to understand how serverless (and other novel) workloads are affected by existing hardware designs,[21] more work is needed to better identify potential low-level improvements and optimizations.

While the FaaS model is a popular method for implementing serverless *applications*, many new workloads are being adapted to the serverless paradigm, which often come with new requirements and will require that serverless frameworks evolve to meet them. For example, AI workloads such as machine learning training and model serving via machine learning inference have been adapted to the serverless model, as have long-lived computations such as scientific and high-performance computing,[17] and games with modifiable virtual environments.[15] Prominent examples of new workload classes adopting the serverless model are shown in the Technical Box on Serverless Applications. Looking further afield, we see opportunities for serverless actor frameworks and for a serverless, low latency, tuple spaces layer that could power new applications by enabling fast, cheap, and reliable communications among cloud functions.

Other challenges of growing concern relate to distribution, federation,

and heterogeneity, with solutions likely starting with rethinking traditional resource management approaches for serverless needs and later spurring more creative solutions. At the envisioned scale of serverless, these could have important economic and climate implications. New low-latency applications, data sovereignty concerns, and specialized hardware are among the factors driving interest in federated serverless computing platforms that link geographically distributed resources.[11] Fine-grained telemetry and other meta-data streams raise important opportunities related to operations, life cycle, and governance with many stakeholders.

TECHNICAL BOX

Serverless Applications

Many fields of inquiry emerge contentiously from existing fields, distinguishing themselves at first by the ability to solve new problems with their proposed techniques, rather than through precise terminology. Examples include systems biology emerging from molecular biology, modern optoelectronics emerging from microwave technology, and computer ecosystems emerging from traditional systems. We argue that serverless computing already supports diverse and specific applications. A 2021 survey of serverless use cases[17,18] identified a wide variety of applications that use cloud functions (FaaS) as critical building blocks in domains ranging from mobile (for implementing core backend functionality) to scientific (for example, metadata extraction[44]). The associated scalability and performance requirements range from ultra-low-latency, as when automating the operational tasks of complex distributed applications to deliver some DevOps and NoOps processes like autoscaling, to high-throughput, as when processing IoT streaming data. These constitute applications that have a fully serverless or partially *serverless architecture*.

For a concrete example, consider anomaly detection in industrial sensor data, an essential task for maintaining large-scale industrial infrastructure at companies like AirFrance-KLM, Shell, and Tata. Such applications can ingest high-velocity data via a scalable messaging system like Apache Kafka, with one or more serverless cloud functions consuming the data in micro batches. These functions can apply simple threshold-based rules or more complex machine learning models to identify anomalous behavior which can then be communicated in real time to an operator via email or SMS using a serverless notification service (like AWS SNS).

Not all serverless applications involve a radical departure from past cloud computing approaches, because some of the first and most important cloud services have been serverless from the start. For example, pay-per-byte-stored serverless object storage systems like AWS S3 do not expose to the user any notion of the number, characteristics, and locations of the physical servers on which data is stored. Consequently, they are in practice seen as infinite storage that can be easily integrated into other solutions, from web applications to Big Data pipelines.

Going beyond cloud functions and storage, the concept of "serverless" can be applied to other systems and scenarios:

- *Serverless databases* (relational, noSQL, or object stores) are offered by many cloud providers. These systems alleviate the user from capacity planning and autoscale as needed, including going into hibernation to scale down to zero after a period of inactivity.
- *Serverless SQL-as-a-Service* products like AWS Athena and Databricks Serverless SQL can be used to query data from a data lake. Results can then be visualized via interactive dashboards by using a serverless visualization solution like AWS Quicksight. This combination enables business intelligence analytics without the need for an always-on data warehouse or for managing visualization software.
- For processing Big Data in the cloud, consumers are turning to *serverless Big Data processing* like AWS Serverless EMR. These services autoscale to meet demand, relieving users from the potential problems of over- or under-provisioning their compute clusters.
- *Serverless edge computing* products like Cloudflare workers and AWS Lambda Edge can be used to provide low latency computing, suitable for many IoT use cases like applying real-time computer vision algorithms.

We expect the concept of serverless will continue to expand to other software pipeline elements and systems. For example, Big Data ETL and CI/CD processes are also being moved to serverless solutions. As these are not long-running services, they can benefit greatly from the flexibility of serverless pricing. Even *serverless caching services* (for example, see goMomento.com) and *serverless streaming services* (for example, AWS MSK Serverless) have started to appear, continuing with the trend toward simplifying the management of software infrastructure.

Conclusion

Serverless computing does not impose an exhaustive list of specific requirements for cloud platforms but rather reflects an evolutionary and gradual process in the advancement of the latter. The question of whether a concrete platform is serverless or not, may not always have a binary—yes or no—answer. Indeed, it may well be that some aspects of a platform exhibit a higher degree of serverless characteristics, while others can be better classified as traditional cloud computing. We expect that serverless offerings will continue to become increasingly diverse and open with the boundaries between different cloud service models increasingly diminishing.

As serverless computing continues to unfold, we expect to see an accelerated shift of focus of cloud platforms and services:

- from low-level VM-based interfaces to high-level interfaces that hide the cloud execution environment with its hardware and software stack (physical machines, VMs, and containers);
- from explicit allocation of resources (for example, VMs, containers) by cloud users to automatic resource allocation, based for example on fine-grained autoscaling mechanisms;
- from cloud users being responsible for configuring and managing operational aspects (like instance deployment/life cycle, elastic scaling, fault tolerance, monitoring, and logging) to offloading such responsibilities to the cloud provider;
- from coarse-grained to fine-grained multi-tenant multiplexing and resource sharing;
- from reservation-based pay-as-you-go billing models to real pay-per-use

models based on actual resource utilization with no costs being charged for idle resources;

▸ from coarse-grained (for example, VM-hours) to fine-grained resource usage accounting and pricing (for example, execution time in 0.1s units); and,

▸ from cloud users having more control of the execution environment to cloud users having less control.

For some serverless platforms like FaaS, one could consider the shift of focus as having already occurred (although being restricted to the relatively narrow scope of today's FaaS platforms). For others like BaaS, the shift of focus is ongoing with many parts still being in the early stages of adoption or barely starting, such as for complex data processing pipelines (for example, graph processing at scale). C

References

1. AWS. re:Invent: Getting started with AWS Lambda, 2014; https://bit.ly/44mcw2r
2. Abad, C., Foster, I.T., Herbst, N., and Iosup, A. Serverless computing (Dagstuhl Seminar 21201). *Dagstuhl Reports 11*, 4 (2021), 34–93; https://doi.org/10.4230/DagRep.11.4.34
3. Agache, A. et al. Firecracker: Lightweight virtualization for serverless applications. In *Proceedings of the 17th USENIX Symp. Networked Systems Design and Implementation*, 2020, 419–434.
4. Allied Market Research. Serverless Market Global, 2022; https://www.alliedmarketresearch.com/serverless-architecture-market.
5. Alpernas, K. et al. Secure serverless computing using dynamic information flow control. In *Proceedings of 2018 OOPSLA*; https://doi.org/10.1145/3276488
6. Armbrust, M. et al. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report. 2009. University of California, Berkeley.
7. Aumala, G., Boza, E., Ortiz-Avilés, L., Totoy, G., and Abad, C. Beyond load balancing: Package-aware scheduling for serverless platforms. In *Proceedings of the 19th IEEE/ACM Intern. Symp. Cluster, Cloud and Grid Computing*, 2019, 282–291; https://doi.org/10.1109/CCGRID.2019.00042
8. Baldini, I. et al. *Serverless Computing: Current Trends and Open Problems*. Springer, Singapore, 2017, 1–20; https://doi.org/10.1007/978-981-10-5026-8_1
9. Buyya, R. et al. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Comput. Surv. 51*, 5, (Nov 2018), Article 105; https://doi.org/10.1145/3241737
10. Castro, P., Ishakian, V., Muthusamy, V., and Slominski, A. The Rise of Serverless Computing. *Commun. ACM 62*, 12 (Dec. 2019), 44–54; https://doi.org/10.1145/3368454
11. Chard, R. et al. FuncX: A federated function serving fabric for science. In *Proceedings of the 29th Intern. Symp. High-Performance Parallel and Distributed Computing*, 2020, 65–76.
12. CNCF. CNCF WG-Serverless Whitepaper v1.0, 2018; https://bit.ly/40JYnsA
13. Datta, P., Kumar, P., Morris, T., Grace, M., Rahmati, A, and Bates, A. Valve: Securing function workflows on serverless computing platforms. In *Proceedings of the Web Conference* (Taipei, Taiwan, 2020). ACM, New York, NY, USA, 939–950; https://doi.org/10.1145/3366423.3380173
14. Dear, B., Ed. *The Friendly Orange Glow: The Untold Story of the PLATO System and the Dawn of Cyberculture*. Pantheon Books, 2017.
15. Donkervliet, J., Trivedi, A., and Iosup, A. Towards supporting millions of users in modifiable virtual environments by redesigning minecraft-like games as serverless systems. In *Proceedings of the 12th USENIX Workshop on Hot Topics in Cloud Computing*. USENIX Assoc. 2020; https://www.usenix.org/conference/hotcloud20/presentation/donkervliet
16. Eismann, S., Bui, L., Grohmann, J., Abad, C., Herbst, N., and Kounev, S. Sizeless: Predicting the optimal size of serverless functions. In *Proceedings of the 22nd Intern. Middleware Conf.* (Québec City, Canada, 2021) ACM, New York, NY, USA, 248–259; https://doi.org/10.1145/3464298.3493398
17. Eismann, S. et al. The state of serverless applications: Collection, characterization, and community consensus. *IEEE Trans. Software Engineering* (2021), 1–1. https://doi. org/10.1109/TSE.2021.3113940
18. Eismann, S. et al. Serverless applications: Why, when, and how? *IEEE Software 38*, 1 (2021), 32–39; https://doi.org/10.1109/MS.2020.3023302
19. Foster, I.T., Geisler, J., Nickless, B., Smith, W., and Tuecke, S. Software infrastructure for the I-WAY metacomputing experiment. *Concurr. Pract. Exp. 10*, 7 (1998), 567–581.
20. Fuerst, A. and Sharma, P. FaasCache: Keeping serverless computing alive with greedy-dual caching. In *Proceedings of the 26th ACM Intern. Conf. Architectural Support for Programming Languages and Operating Systems*. ACM, 2021, New York, NY, USA, 386–400; https://doi.org/10.1145/3445814.3446757
21. Gan, Y. et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems. In *Proceedings of the 24th Intern. Conf. Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA, 2019). ACM, New York, NY, USA, 3–18; https://doi.org/10.1145/3297858.3304013
22. Greenberger, M. The computers of tomorrow. *Atlantic Monthly* (1964).
23. Hellerstein, J.M. et al. Serverless computing: One step forward, two steps back. In *Proceedings of the 9th Biennial Conf. on Innovative Data Systems Research*, 2019; http://cidrdb.org/cidr2019/papers/p119-hellerstein-cidr19.pdf
24. Herbst, N., Kounev, S., and Reussner, R. Elasticity in cloud computing: What it is, and what it is not. In *Proceedings of the 10th Intern. Conf. Autonomic Computing*. USENIX Assoc. (San Jose, CA, USA, 2013), 23–27; https://bit.ly/3AKntwT
25. Industry ARC. Serverless architecture market, 2022; https://bit.ly/3oZdwt5.
26. Jonas, E. et al. Cloud programming simplified: A Berkeley view on serverless computing. CoRR, 2019; http://arxiv.org/abs/1902.03383
27. Kleinrock, L. The first message transmission. Technical Report, 2019; https://go.icann.org/3Hycxq6
28. Klimovic, S., Wang, Y., Stuedi, P., Trivedi, A., Pfefferle, J., and Kozyrakis, C. Pocket: Elastic ephemeral storage for serverless analytics. In *Proceedings of the 13th USENIX Symp. Operating Systems Design and Implementation* (Carlsbad, CA, USA, Oct. 8–10, 2018). A.C. Arpaci-Dusseau and G. Voelker, Eds. USENIX Assoc., 427–444; https://www.usenix.org/conference/osdi18/presentation/klimovic
29. Kounev, S. et al. Toward a definition for serverless computing. *Serverless Computing* (Dagstuhl Seminar 21201), C. Abad, I.T. Foster, N. Herbst, and A. Iosup, Eds. Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 34–93; https://doi.org/10.4230/DagRep.11.4.34
30. Li, Z., Guo, L., Cheng, J., Chen, Q., He, B., and Guo, M. The serverless computing survey: A technical primer for design architecture. *ACM Comput. Surv. 54*, 10s, Article 220 (Sept. 2022).
31. Licklider, JCR. Intergalactic Computer Network. Technical Report, 1963.
32. Mampage, A., Karunasekera, S., and Buyya, R. A holistic view on resource management in serverless computing environments: Serverless computing: What it is, and what it is not? Taxonomy and future directions. *ACM Comput. Surv. 54*, 11s, (Sept. 2022), Article 222; https://doi.org/10.1145/3510412
33. Markets and Markets. Serverless architecture market, 2022; https://bit.ly/3HrW0E7.
34. Mell, P.M. and Grance, T. SP 800-145. The NIST Definition of Cloud Computing. Technical Report, 2011. Gaithersburg, MD, USA.
35. Mordor Intelligence. Serverless computing market, 2022; https://bit.ly/3oVGKc5.
36. Parkhill, D.F. *Challenge of the Computer Utility*. Addison-Wesley, 1966.
37. Patterson, L. et al. HiveMind: A hardware-software system stack for serverless edge swarms. In *Proceedings of the 49th Annual Intern. Symp. Computer Architecture* (New York, NY, USA, 2022). ACM, New York, NY, USA, 800–816; https://doi.org/10.1145/3470496.3527407
38. Poppe, O. et al. Moneyball: Proactive auto-scaling in Microsoft Azure SQL database serverless. In *Proceedings of VLDB Endow. 15*, 6 (Feb. 2022), 1279–1287.
39. Romero, F. et al. FaaST: A transparent auto-scaling cache for serverless applications. In *Proceedings of the ACM Symp. Cloud Computing* (Seattle, WA, USA, 2021). ACM, New York, NY, USA, 122–137; https://doi.org/10.1145/3472883.3486974
40. Sankaran, A., Datta, P., and Bates, A. Workflow integration alleviates identity and access management in serverless computing. In *Proceedings of the 2020 Computer Security Applications Conf.* (Austin, TX, USA). ACM, New York, NY, USA, 496–509; https://doi.org/10.1145/3427228.3427665
41. Schleier-Smith, J. et al. What serverless computing is and should become: The next phase of cloud computing. *Commun. ACM 64*, 5 (May 2021), 76–84; https://doi.org/10.1145/3406011
42. Silva, P., Fireman, D., and Pereira, T.E. Prebaking functions to warm the serverless cold start. In *Proceedings of the 21st Intern. Middleware Conf.* (Delft, Netherlands, 2020). ACM, New York, NY, USA, 1–13; https://doi.org/10.1145/3423211.3425682
43. Singhvi, A., Balasubramanian, A., Houck, K., Shaikh, M.D., Venkataraman, S., and Akella, A. Atoll: A scalable low latency serverless platform. In *Proceedings of the 2021 ACM Symp. Cloud Computing* (Seattle, WA, USA). ACM, New York, NY, USA, 138–152; https://doi.org/10.1145/3472883.3486981
44. Skluzacek, T.J., Wong, R., Li, Z., Chard, R., Chard, K., and Foster, I. A serverless framework for distributed bulk metadata extraction. In *Proceedings of the 30th Intern. Symp. High-Performance Parallel and Distributed Computing*, 2021, 7–18.
45. Smarr, L. and Catlett, C.E. Metacomputing. *Commun. ACM 35, 6* (1992), 44–52; https://doi.org/10.1145/129888.129890
46. Spillner, J. and Al-Ameen, M. Serverless literature dataset. Zenodo dataset (3rd revision), 2019; https://doi.org/10.5281/zenodo.1175423.
47. van Eyk, E., Toader, L., Talluri, S., Versluis, L., Ută, A., and Iosup, A. Serverless is more: From PaaS to present cloud computing. *IEEE Internet Computing 22*, 5 (2018), 8–17; https://doi.org/10.1109/MIC.2018.053681358
48. Varghese, B. and Buyya, R. Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems 79* (2018), 849–861; https://doi.org/10.1016/j.future.2017.09.020
49. Verified Market Research. Serverless architecture market size, 2022; http://bit.ly/3rLRprh.
50. Wen, J., Chen, Z., Jin, X., and Liu, X. Rise of the planet of serverless computing: A systematic review. *ACM Trans. Softw. Eng. Methodol.* (Jan. 2023); https://doi.org/10.1145/3579643
51. Zhang, W., Fang, V., Panda, A., and Shenker, S. Kappa: A programming framework for serverless computing. In *Proceedings of the 11th ACM Symp. on Cloud Computing*, 2020. ACM, New York, NY, USA, 328–343.

Samuel Kounev, University of Würzburg, Germany.

Nikolas Herbst, University of Würzburg, Germany.

Cristina L. Abad, ESPOL, Ecuador.

Alexandru Iosup, VU Amsterdam. The Netherlands.

Ian Foster, Argonne National Lab and University of Chicago, IL, USA.

Prashant Shenoy, University of Massachusetts, Amherst, MA, USA.

Omer Rana, Cardiff University, U.K.

Andrew A. Chien, The University of Chicago and Argonne National Lab, USA.

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/serverless-computing-what-it-is