Serverless Applications: Why, When, and How?

Simon Eismann, Julius-Maximilian University

Joel Scheuner, Chalmers |University of Gothenburg

Erwin van Eyk, Vrije Universiteit Amsterdam

Maximilian Schwinger, German Aerospace Center

Johannes Grohmann and Nikolas Herbst, Julius-Maximilian University

Cristina L. Abad, Escuela Superior Politecnica del Litoral

Alexandru Iosup, Vrije Universiteit Amsterdam

// Why do so many companies adopt serverless? When are serverless applications well suited? How are they currently implemented? To address these questions, we analyze 89 serverless applications from open source projects, industrial sources, academic literature, and scientific computing—presenting the most extensive study to date. //

Digital Object Identifier 10.1109/MS.2020.3023302
Date of current version: 22 December 2020

EASE OF USE and efficiency are two of the most desirable properties of software services. Ease of use entices more people to try services and allows more individuals to continue using them. Efficiency enables increased and longer use of services and, as the scale of software services has already reached a significant fraction of the world's energy consumption, keeps them sustainable. But ease of use and efficiency have been historically at odds with each other: The easier the software is to use, the fewer hints it can provide to the platform on which it runs, leaving it to solve complex resource management problems (in particular, related to servers). Although not a universal remedy, serverless computing aims to provide both ease of use and efficiency for common software services that run on cloud resources. In this article, we investigate why, when, and how these serverless applications are useful.

Serverless computing is any computing platform that "hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running."[1] "The Mobile Back End for a Social Media App" exemplifies how serverless can be useful. More generally, serverless applications combine managed stateless ephemeral computing solutions [function-as-a-service (FaaS)], such as Amazon Web Services (AWS) Lambda, Azure Functions, and Google Cloud Functions, with fully provider-managed services for messaging, file storage, databases, streaming, and authentication [back end-as-a-service (BaaS)]. Serverless computing is being increasingly adopted by industry[3–5] and studied by academics.[6,7] One crucial reason is that serverless operations empower developers to focus on implementing business logic and to let the cloud providers handle all operational

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. Restrictions apply.

THE MOBILE BACK END FOR A SOCIAL MEDIA APP

In this example application, a social media user wants to publish a status update, which should be seen by all of the user's friends. In 2019, this happened more than 1 billion times per day on social media platforms, such as Facebook, Twitter, and Instagram. Technology wise, this would happen through a four-step process: 1) the user would compose the status update using the mobile clients of the social media platform; 2) then, the user would send it using the mobile client; 3) the platform would orchestrate the operations needed to propagate the update inside the platform and to the user's friends; and 4) each friend would receive the update on their social media clients, for example, on their mobile phones. Step 3 is the "secret sauce" of the social media platform—although the users and their friends never see it, the software and the resources (the servers) on which the software run ensure the technical sustainability of the social media platform.

With conventional technology, for step 3, the platform operators would need not only to develop the logic of routing the status update to each of the user's friends but also to carefully manage the resources (the servers) on which the logic (the software) can run and to make sure the logic runs correctly. Resource management, and, in particular, resource provisioning and allocation, are long-lasting hard problems in computing. Obstacles, such as scaling the resources proportionally to the number of users (and their friends), are exacerbated by the fine-grained nature of each operation. Running the logic is also challenging when subject to strict performance requirements—after all, the user expects all their friends to see the status update immediately and to reply to it in a matter of seconds.

With serverless technology, the cloud provider abstracts away the server management, provisioning servers with fine granularity, on demand, and with a pay-per-use model. To benefit from this, the serverless software gets transformed. Simplified, the process of posting updates proceeds as follows. The users post status updates using their mobile clients as an HTTP request to the API Gateway (which operates as *serverless request routing*, that is, a specialized function that routes only requests and runs very efficiently without further developer intervention). Step 3 proceeds: the API Gateway triggers a lambda function (serverless compute), which queries the user's friends from a DynamoDB table (serverless storage) and publishes the status update to friends using the Simple Notification Service (SNS) (serverless publish/subscribe). Finally, SNS generates push notifications with the status update for the user's friends. Tens of other serverless functions get invoked, to authenticate the user, to authorize posting messages, to copy data between various locations, and so on. All of these are orchestrated on resources managed by the cloud operator. The serverless functions are fine-grained, which leads to higher scalability than the coarser conventional approaches but at the cost of more complex orchestration. For further examples of serverless applications, we refer to our technical report.[2]

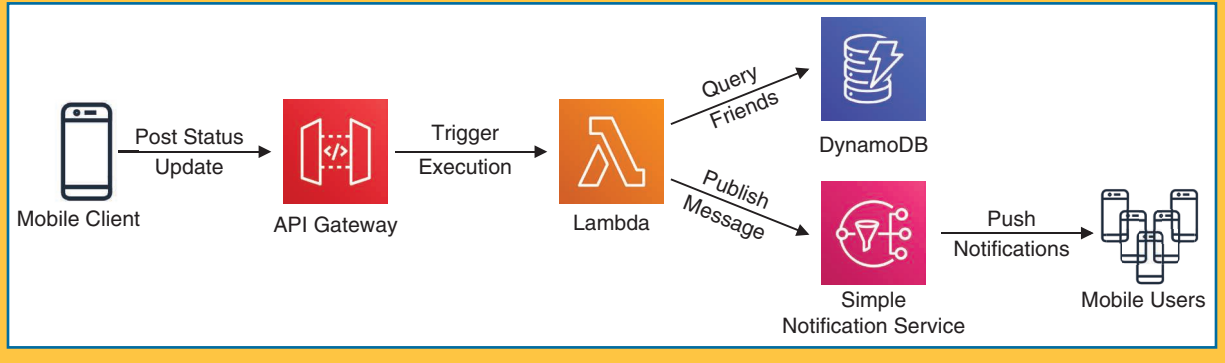

FIGURE S1. An example serverless application: the mobile back end for a social media app.

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. | Restrictions apply.

concerns, such as deployment, resource allocation, and autoscaling.[1]

However, only a few, and sometimes conflicting, reports address important questions, such as 1) "Why are practitioners choosing to build serverless applications?"; 2) "When are serverless applications well suited?"; and 3) "How are serverless applications implemented in practice?" For example, there are reports of significant cost savings by switching to serverless applications[8,9] but also articles suggesting there is higher cost in some scenarios compared to traditional hosting.[10] Having concrete information on these topics would be valuable for managers to guide decision making on whether a serverless application can be a suitable solution for a specific use case. The Standard Performance Evaluation Corporation Research Cloud Group suggests that surveys of real-world serverless computing are needed to understand architectural, implementation, and deployment patterns emerging in the "serverless soup."[6] Leitner et al.[5] also discuss that empirical studies about serverless use are required to guide software developers in building serverless solutions.

Addressing the need for a comprehensive empirical study of serverless applications, we collected a total of 89 descriptions of existing cases. We use diverse sources, for example, GitHub projects, blog posts, scientific publications, and talks at industry conferences. We analyze every serverless application regarding 24 characteristics (see "Methodology for Serverless Application Collection and Characterization" for a description of our methodology). Based on these data, we investigate why practitioners are choosing to adopt the serverless paradigm, when serverless applications are used, and how they are implemented in practice. The percentage values for the characteristics presented in the remainder of this article exclude any serverless application for which we could not determine the respective trait.

> Although not a universal remedy, serverless computing aims to provide both ease of use and efficiency for common software services that run on cloud resources.

Why Are Practitioners Choosing to Adopt Serverless?

Pioneers and journalists seem to agree on several potential benefits of serverless applications: reduced operation effort, faster development due to the heavy use of BaaS, and near-infinite scalability. Many also discuss significant cost savings from switching to serverless. However, not all of these benefits hold in general. For example, cost savings have come under scrutiny.[10] To understand why practitioners choose to adopt serverless, we investigated the descriptions and documentation of applications in our data set.

For 27 of these applications, we cannot discern the motivation for going serverless. Of the remaining 62 applications, 47% chose serverless to save costs. Serverless holds this financial promise due to its pay-per-use model for irregular or bursty workloads, which would have low resource utilization and thus higher cost with traditional hosting options. This concurs with our observation that 84% of the serverless applications have bursty workloads.

Another reason for serverless adoption, which applies for 34% of the applications in our data set, is that developers no longer need to bother with operational concerns, such as deployment, scaling, or monitoring, and instead can focus on developing new features. This was not only reported for utility functionality, such as a continuous integration/continuous delivery pipeline or malware detection, where traditionally servers need to be maintained for only a few and irregular executions. Interestingly, it was also reported for user-facing application programming interfaces (APIs) serving large-scale traffic, where one would expect other concerns, such as performance or availability to take priority.

A third reason (also 34%) is the scalability of serverless applications. Although traditional applications can also be scalable, serverless ones offer near-infinite, out-of-the-box scalability with minimal engineering effort. This is because the FaaS implementation commonly used for serverless computing is fine-grained and thus conveniently parallel.

Based on our data set, the most commonly reported reasons for the adoption of serverless are to save costs for irregular or bursty workloads, to avoid operational concerns, and for built-in scalability. Other reasons, such as improved performance and faster time to market, are

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. Restrictions apply.

METHODOLOGY FOR SERVERLESS APPLICATION COLLECTION AND CHARACTERIZATION

We collected descriptions of serverless application from four sources: open source projects, academic literature, industrial literature, and scientific computing. Next, we randomly assigned two independent reviewers to analyze each serverless application based on a set of fixed characteristics. In the following discussion and consolidation phase, we considered and resolved any differences between the two resulting characteristic reviews. For the scientific applications, a different approach was necessary, as many of them were not publicly available yet. Therefore, these applications were reviewed by a single domain expert, who is either involved in the development of the applications or in direct contact with the development team. If the information to determine a characteristic for a serverless application was not available, we labeled the characteristic as "Unknown." The percentage of "Unknowns" ranges from 0 to 19%, with two outliers at 25 and 30% for the characteristics presented in this article. These "Unknowns" were excluded from the percentage values presented in this article. For a more detailed breakdown of our results and a more in-depth description of our methodology, we refer to our technical report.[2]

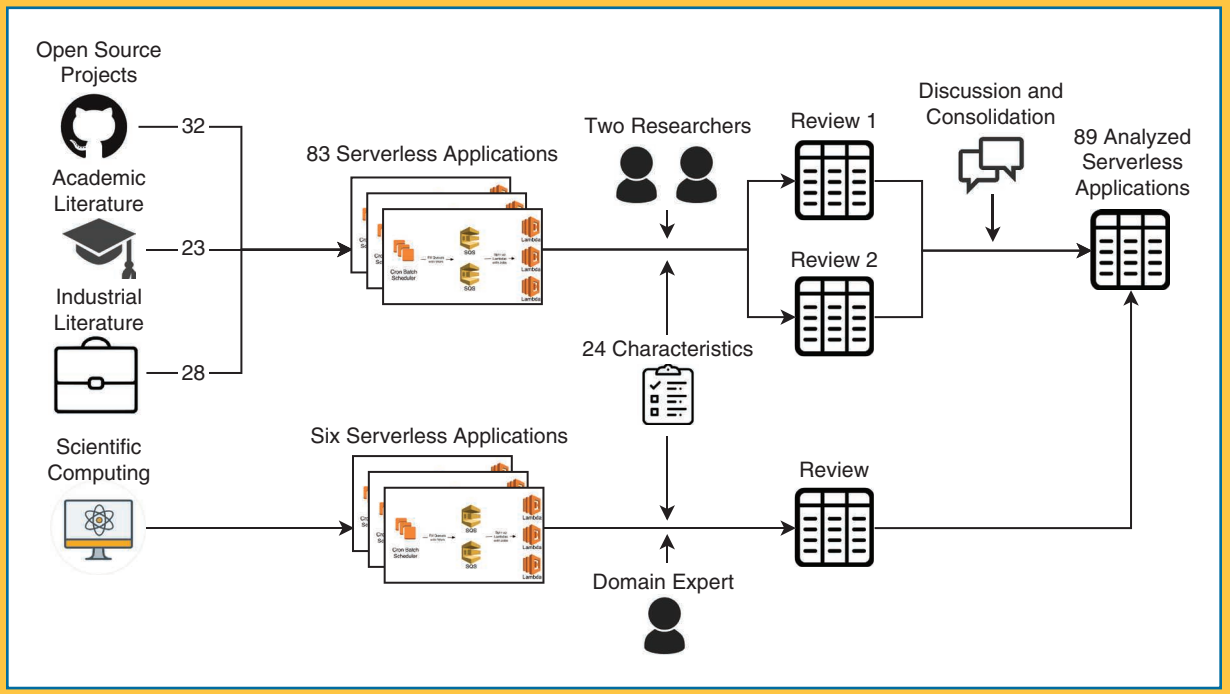

FIGURE S2. Methodology for serverless application collection and characterization.

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. | Restrictions apply.

less common (19 and 13%, respectively). These findings are mostly in accordance with a recent community survey where more than 160 participants[9] reported that the positive impacts of adopting serverless are the event-driven architecture (51%), cost of resources (44%), speed of development (36%), flexibility of scaling (31%), and application performance (19%).

When Are Serverless Applications Used?

When starting a project, managers need to decide which technology stack is best suited. Currently, managers are faced with the difficult question of if serverless is well suited for a specific application. A common assumption is that serverless is best suited for utility functionality and less applicable for latency-critical, high-volume core functionality. For example, Netflix uses AWS Lambda for utility functionality, such as video encoding, file backup, security audits of EC2 instances, and monitoring. However, the core functionality, such as the website/app back-end or video delivery, is still running on traditional infrastructure as a service cloud services.[11]

Contrary to this popular belief, we found that serverless applications are not limited to utility functionality. Our data indicate that many of them implement it (39%), but also that many of them implement core functionality (42%) and scientific workloads (16%). This is consistent with our finding that a substantial number of serverless applications (39%) experience high traffic intensity, a high proportion complemented by on-demand applications experiencing low traffic intensity (47%) and by scheduled applications (17%).

A common argument against serverless applications is that cold starts make them unsuitable for applications with latency requirements. On the contrary, we find that they are used for latency-critical tasks, despite the cold starts affecting tail latencies. Concretely, 38% of the surveyed serverless applications have no latency requirements. However, 32% of them have latency requirements for all functionality, 28% have partial latency requirements, and 2% even have real-time requirements.

Another argument is that current serverless platforms can be unsuited for long-running tasks or ones with large data volumes.[7] Our data set supports this hypothesis as 69% of the surveyed serverless applications have a data volume of less than 10 MB and 75% have an execution time in the range of seconds. To overcome this limitation, the area needs further innovation.

To summarize, serverless applications are most commonly used for short-running tasks with low data volume and bursty workloads. However, contrary to popular belief, these applications are also widely used for latency-critical, high-volume core functionality. Overall, there are examples of them across all application types, requirements, and workloads.

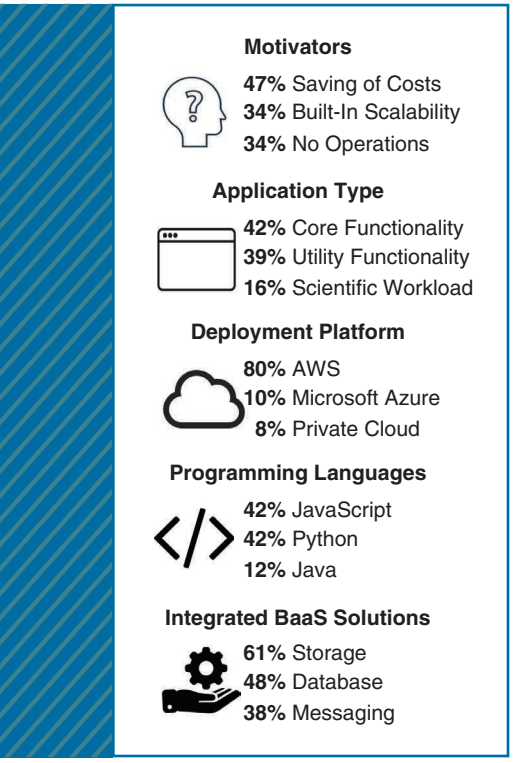

FIGURE 1. The key findings of our study. The results are limited to the top three values, and a single application can have multiple values for motivators, programming languages, and integrated BaaS solutions. For more detailed results, we refer to our technical report.[2]

How Are Serverless Applications Implemented in Practice?

When implementing a serverless application, engineers face several technology and architecture decisions, such as selecting the cloud platform, serverless platform, programming language, BaaS options, and appropriate granularity level for serverless functions. Based on our collection, we identify the most popular approaches to implementing serverless applications.

Among the surveyed applications, AWS is by far the most popular deployment platform, chosen by 80% of the applications (see Figure 1) The other cloud vendors are less represented, with Azure at 10%, IBM at 7%, and Google Cloud at 3%. We see two potential reasons for this vast lead: 1) AWS introduced their FaaS platform (AWS Lambda) two years before the other cloud vendors, so it is likely to be the most mature, and 2) AWS has the largest market share of general cloud computing,[12] which gives it a larger existing user base that can move applications to serverless. About 8% of the surveyed

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. Restrictions apply.

serverless applications run in private clouds; however, these are mostly applications from academia and scientific computing.

The low adoption of private clouds is in stark contrast to the large number of existing open source FaaS frameworks.[6] Part of the appeal of the serverless application model is the automation of operational concerns. We hypothesize that the increase in such concerns, which comes with maintaining a fleet of servers and an open source FaaS framework, is deterring the adoption of these frameworks. Additionally, most serverless applications make use of managed services (storage, databases, messaging, logging, streaming, and so on), which are not available directly in a private cloud environment.

Serverless platforms support popular programming languages. In our study, we were able to determine the language for 67 of the 89 serverless applications. JavaScript (42%) and Python (42%) were by far the most popular languages. Some applications are also written in Java (12%), C/C++ (11%), or C# (8%), while only a few use Go (5%) or Ruby (2%). Traditionally, interpreted languages, such as JavaScript and Python, have lower cold-start times, that is, the time required to initialize a new instance, than compiled languages. However, as the technology matures, this difference seems to level out,[13] and there are multiple new approaches for cold-start mitigation. For example, AWS introduced the reserved-capacity feature, which provides prewarmed function instances to avoid cold starts. The adoption of such cold-start mitigation strategies could lead to an increased usage of compiled languages.

Cloud providers offer managed services as part of BaaS, such as messaging, file storage, databases, streaming, logging, and machine learning. In our study, we found that the most popular external services are cloud storage (for example, S3) and databases (for example, Dynamo DB). As serverless functions are ephemeral and stateless, they need to rely on external services to persist data and manage state. Many serverless applications also use some form of managed messaging (38%), such as managed pub/sub (17%), streaming (11%), or queues (10%). Messaging services are popular because serverless functions rarely communicate via external calls, as this results in double billing.[14] The popularity of messaging services is consistent with our finding that serverless applications mostly use event-based architectures (60%) or workflow engines (38%) to coordinate multiple functions. It is interesting to note that only 12% of serverless applications use no BaaS solutions, highlighting the symbiosis between FaaS and BaaS.

The appropriate granularity of serverless functions is currently debated.[15] Opinions range from wrapping each program function or each API endpoint to the full-scale conversion of each microservice into serverless functions. We find that 82% of the surveyed serverless applications consist of five functions or less and that 93% consist of 10 functions or fewer. With one exception, the remaining applications consist of fewer than 20. Therefore, the granularity of a serverless function is more akin to a full microservice or an API endpoint in our data set. Consequently, the term *serverless function* might be somewhat of a misnomer as they are not related to the general programming concept of functions.

To summarize, serverless applications are mostly implemented on AWS, in either Python or JavaScript, and commonly make use of cloud storage, managed databases, and messaging services. Also, serverless functions are not comparable to programming functions in terms of size, according to our data set.

Based on our analysis of 89 serverless applications, we find that the most commonly reported reasons for the adoption of serverless are to save costs for irregular or bursty workloads, to avoid operational concerns, and for the built-in scalability. Serverless applications are most commonly used for short-running tasks with low data volume and bursty workloads but are also frequently used for latency-critical, high-volume core functionality. They are mostly implemented on AWS, in either Python or JavaScript, and make heavy use of BaaS. We see this study as a step toward a community-wide policy of sharing and discussion of serverless applications. Such a catalog could stimulate a new wave of serverless designs, facilitate meaningful tuning and performance benchmarking, and overall prove useful for both industry and academia.

References

1. P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Commun. ACM*, vol. 62, no. 12, pp. 44–54, Nov. 2019. doi: 10.1145/3368454.
2. S. Eismann et al., "A review of serverless use cases and their characteristics," SPEC RG, Gainesville, Tech. Rep., Aug. 2020. Accessed: May 2020. [Online]. Available: https://bit.ly/36Ntvgy.
3. "$7.72 billion function-as-a-service market 2017." Businesswire.com. https://bwnews.pr/2VBDBgC (accessed Feb. 2017).
4. "Accelerating with serverless!" Medium.com. https://bit.ly/35bcps3 (accessed Apr. 2020).

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. | Restrictions apply.

ABOUT THE AUTHORS

SIMON EISMANN is currently a Ph.D. student at the chair of software engineering at the University of Würzburg, Würzburg, Germany. His research interests include cloud computing and serverless and performance analysis/modeling. Eismann received his M.S. from the University of Würzburg. He is a Member of IEEE. Further information about him can be found at https://se.informatik.uni-wuerzburg.de/software-engineering-group/staff/simon-eismann/. Contact him at simon.eismann@uni-wuerzburg.de.

JOEL SCHEUNER is a Ph.D. student at the joint division of software engineering at Chalmers University of Technology and the University of Gothenburg, Gothenburg, Sweden. His research interests include cloud computing, performance engineering, and software engineering. Scheuner received his M.S. in software systems from the University of Zurich. Further information about him can be found at https://joelscheuner.com/. Contact him at scheuner@chalmers.se.

ERWIN VAN EYK is a Ph.D. student at Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, and the chair of the Standard Performance Evaluation Corporation Research Cloud Group Serverless activity. His research interests include efficient and automated cloud solutions, including serverless computing. Van Eyk received an M.Sc. from TU Delft, The Netherlands, in 2019 for work on cloud computing and serverless workflows. Further information about him can be found at https://erwinvaneyk.nl. Contact him at erwinvaneyk@gmail.com.

MAXIMILIAN SCHWINGER is a Ph.D. student at the chair for software engineering at the University of Würzburg, Würzburg, Germany, and is working for the German Aerospace Center as a software and systems engineer. His research interests include high-performance computing, cloud-based computing, and scientific computing in the domain of satellite-based Earth observation. Schwinger received a diploma in computer science from TU Munich. Further information about him can be found at https://www.dlr.de/eoc/de/desktopdefault.aspx/tabid-5242/8788_read-17897/. Contact him at maximilian.schwinger@dlr.de.

5. P. Leitner, E. Wittern, and J. Spillner, "A mixed-method empirical study of Function-as-a-Service software development in industrial practice," *J. Syst. Softw.*, vol. 149, pp. 340–359, Mar. 2019. doi: 10.1016/j.jss.2018.12.013.
6. E. Van Eyk et al., "The SPEC-RG reference architecture for FaaS: From microservices and containers to serverless platforms," *IEEE Internet Comput.*, vol. 23, no. 6, pp. 7–18, Nov. 2019. doi: 10.1109/MIC.2019.2952061.
7. J. Hellerstein et al., "Serverless computing: One step forward, two steps back," in *Proc. 9th Biennial Conf. Innov. Data Syst. Res. (CIDR)*, Jan. 2019. [Online]. Available: http://bit.ly/TwoStepsBackward
8. G. Adzic and R. Chatley, "Serverless computing: Economic and architectural impact," in *Proc. 11th Joint Meeting Found. Softw. Eng.*, Sept. 2017, pp. 884–889. doi: 10.1145/3106237.3117767.
9. E. Levinson, "Serverless community survey 2020: Day1." [Online]. http://bit.ly/SerServGit
10. A. Eivy, "Be wary of the economics of serverless cloud computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 6–12, Apr. 2017. doi: 10.1109/MCC.2017.32.
11. J. Demian. "Serverless case study: Netflix." https://bit.ly/3d4iEAZ (accessed July 2018).

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. Restrictions apply.

ABOUT THE AUTHORS

JOHANNES GROHMANN is currently a Ph.D. student at the chair of software engineering at the University of Würzburg, Würzburg, Germany. His research interests include serverless and cloud computing, and performance model learning and analysis. Grohmann received his M.S. from the University of Würzburg. He is a Member of IEEE. Further information about him can be found at https://se.informatik.uni-wuerzburg.de/software-engineering-group/staff/johannes-grohmann/. Contact him at johannes.grohmann@uni-wuerzburg.de.

CRISTINA L. ABAD is an associate professor at Escuela Superior Politecnica del Litoral, Guayaquil, Ecuador, where she leads the Distributed Systems Research Lab. Her research interests lie at the intersection of distributed systems and performance engineering. Abad received her Ph.D. in computer science from the University of Illinois at Urbana-Champaign. She is a Member of IEEE, the Association for Computing Machinery, the Standard Performance Evaluation Corporation, and the Advanced Computing Systems Association. Further information about her can be found at https://sites.google.com/fiec.espol.edu.ec/cv-cabad/english. Contact her at cabad@fiec.espol.edu.ec.

NIKOLAS HERBST is a research group leader at the chair of software engineering at the University of Würzburg, Würzburg, Germany. His research interests include predictive data analysis, elasticity in cloud computing, autoscaling and resource management, performance evaluation of virtualized environments, and autonomic and self-aware computing. Herbst received his Ph.D. from the University of Würzburg. He serves as elected vice-chair of the Standard Performance Evaluation Corporation Research Cloud Group. He is a Member of IEEE. Further information about him can be found at https://se.informatik.uni-wuerzburg.de/software-engineering-group/staff/nikolas-herbst/. Contact him at nikolas.herbst@uni-wuerzburg.de.

ALEXANDRU IOSUP is a full professor and university research chair at Vrije Universiteit (VU) Amsterdam, Amsterdam, The Netherlands, and member of the Young Royal Academy of Arts and Sciences of The Netherlands. His research interest is massivizing computer systems, at the intersection of distributed systems, performance engineering, and software engineering. He serves as the chair of the Massivizing Computer Systems research group at VU and of the Standard Performance Evaluation Corporation Research Cloud Group. He is a Member of IEEE, the Association for Computing Machinery, and the Standard Performance Evaluation Corporation. Further information about him can be found at http://atlarge.science/aiosup/. Contact him at a.iosup@vu.nl.

12. "Gartner says worldwide IaaS public cloud services market grew 31.3% in 2018." Businesswire.com. https://bwnews.pr/2ZcI7o4 (accessed July 2019).
13. N. Malishev, "AWS Lambda cold start language comparisons, 2019 edition." Gitconnected.com. Accessed: Sept. 2019. [Online]. Available: https://bit.ly/ColdStartComp
14. I. Baldini et al., "The serverless trilemma: Function composition for serverless computing," in *Proc. ACM SIGPLAN Int. Symp. New Ideas, New Paradigms, Reflections Program. Softw.*, Oct. 2017, pp. 89–103. doi: 10.1145/3133850.3133855.
15. J. Spillner, C. Mateos, and D. Monge, "Faaster, better, cheaper: The prospect of serverless scientific computing and HPC," in *Proc. Latin Amer. High Perform. Comput. Conf.*, Dec. 2017, pp. 154–168. doi: 10.1007/978-3-319-73353-1_11.

Authorized licensed use limited to: University of L'Aquila. Downloaded on December 30,2023 at 14:14:05 UTC from IEEE Xplore. | Restrictions apply.