Aleksi Pekkala

Migrating a web application to serverless architecture

Master's Thesis in Information Technology

February 12, 2018

University of Jyväskylä

Department of Mathematical Information Technology

Author: Aleksi Pekkala

Contact information: alvianpe@student.jyu.fi

Supervisor: Oleksiy Khriyenko

Title: Migrating a web application to serverless architecture

Työn nimi: Web-sovelluksen siirtäminen serverless-arkkitehtuuriin

Project: Master's Thesis

Study line: Master's Thesis in Information Technology

Page count: 25+0

Abstract: This document is a sample gradu3 thesis document class document. It also functions as a user manual and supplies guidelines for structuring a thesis document.

The abstact is typically short and discusses the background, the aims, the research methods, the obtained results, the interpretation of the results and the conculsions of the thesis. It should be so short that it, the Finnish translation, and all other meta information fit on the same page.

The Finnish tiivistelmä of a thesis should usually say exactly the same things as the abstract.

Keywords: serverless, FaaS, architecture, cloud computing, web applications

Suomenkielinen tiivistelmä: Tämä kirjoitelma on esimerkki siitä, kuinka gradu3-tutkielmapohjaa käytetään. Se sisältää myös käyttöohjeet ja tutkielman rakennetta koskevia ohjeita.

Tutkielman tiivistelmä on tyypillisesti lyhyt esitys, jossa kerrotaan tutkielman taustoista, tavoitteesta, tutkimusmenetelmistä, saavutetuista tuloksista, tulosten tulkinnasta ja johtopäätöksistä. Tiivistelmän tulee olla niin lyhyt, että se, englanninkielinen abstrakti ja muut metatiedot mahtuvat kaikki samalle sivulle.

Sen tulee kertoa täsmälleen samat asiat kuin englannikielinen abstrakti.

Avainsanat: serverless, FaaS, arkkitehtuuri, pilvilaskenta, web-sovellukset

Contents

1 Introduction

Cloud computing has in the past decade emerged as a veritable backbone of modern economy, driving innovation both in industry and academia as well as enabling scalable global enterprise applications. Just as the adoption of cloud computing continues to increase, the technologies in which the paradigm is based have continued to progress. Recently the development of novel virtualization techniques has lead to the introduction of *serverless computing*, an architectural pattern based on ephemeral cloud resources that scale up and down automatically and are billed for actual usage at a millisecond granularity. The main drivers behind serverless computing are to both reduce operational costs by more efficient cloud resource utilization and to improve developer productivity by shifting provisioning, load balancing and other infrastructure concerns to the platform. (Buyya et al. 2017)

As an appealing economic proposition, serverless computing has attracted significant interest in the industry. This is illustrated for example by its appearance in the 2017 Gartner Hype Technologies Report (Walker 2017). By now most of the prominent cloud service providers have introduced their own serverless platforms, promising capabilities that make writing scalable web services easier and cheaper (e.g. Amazon 2018; Google 2018; IBM 2018; Microsoft 2018). A number of high-profile use cases have also been presented in the literature (CNCF 2018). Baldini, Castro, et al. (2017) however note a lack of corresponding degree of interest in academia despite a wide variety of technologically challenging and intellectually deep problems in the space.

One of the open problems identified in literature concerns the discovery of serverless design patterns: how do we compose the granular building blocks of serverless into larger systems? (Baldini, Castro, et al. 2017) Varghese and Buyya (2018) contend that one challenge hindering the widespread adoption of serverless will be the radical shift in the properties that a programmer will need to focus on, from latency, scalability and elasticity to those relating to the modularity of an application. Considering this and the paradigm's unique characteristics and limitations, it's unclear to what extent our current patterns apply and what kind of new patterns are best suited to optimize for the features of serverless computing. The object of this thesis is to fill the gap by re-evaluating existing design patterns in the serverless context

and proposing new ones through an exploratory migration process.

1.1 Research problem

The research problem addressed by this thesis distills down to 4 different questions:

1. Why should a web application be migrated to serverless?
2. What kind of patterns are there for building serverless web application backends?
3. Do the existing patterns have gaps or missing parts, and if so, can we come up with improvements or alternative solutions?
4. How does migrating a web application to serverless affect its quality?

The first two questions are addressed in the theoretical part of the thesis. Question 1 concerns the motivation behind the thesis and introduces serverless migration as an important and relevant business problem. Question 2 is answered by surveying existing literature for serverless patterns as well as other, more general patterns thought suitable for the target class of applications.

The latter questions form the constructive part of the thesis. Question 3 concerns the application and evaluation of surveyed patterns. The surveyed design patterns are used to implement a subset of an existing traditional web application in the serverless architecture. In case the patterns prove unsuitable for any given problem, alternative solutions or extensions are proposed. The last question consists of comparing the migrated portions of the app to the original version and evaluating whether the posited benefits of serverless architecture are in fact realized.

1.2 Outline

The thesis is structured as follows: the second chapter serves as an introduction to the concept of serverless computing. The chapter describes the main benefits and drawbacks of the platform, as well as touching upon its internal mechanisms and briefly comparing the main service providers. Extra emphasis is placed on how the platform's limitations should be taken into account when designing web application backends.

The third chapter consists of a survey into existing serverless design patterns and recommendations. Applicability of other cloud computing, distributed computing and enterprise integration patterns is also evaluated.

The fourth chapter describes the process of migrating an existing web application to serverless architecture. The patterns discovered in the previous chapter are utilized to implemented various typical web application features on a serverless platform. In cases where existing patterns prove insufficient or unsuitable as per the target application's characteristics, modifications or new patterns are proposed.

The outcome of the migration process is evaluated in the fifth chapter. The potential benefits and drawbacks of the serverless platform outlined in chapter 2 are used to reflect on the final artifact. The chapter includes approximations on measurable attributes such as hosting costs and performance as well as discussion on the more subjective attributes like maintainability and testability. The overall ease of development – or developer experience – is also addressed since it is one of the commonly reported pain points of serverless computing (Eyk et al. 2017).

The final chapter of the thesis aims to draw conclusions on the migration process and the resulting artifacts. The chapter contains a summary of the research outcomes and ends with recommendations for further research topics.

2 Serverless computing

This chapter serves as an introduction to serverless computing. We start by (attempting to define serverless) going over the background and motivations behind the serverless paradigm. This is followed by a look at the basic features of a serverless runtime, as well as the major providers and use cases. The chapter closes with notes on the drawbacks and limitations of serverless, particularly from the point of view of web application backends.

2.1 Defining serverless

Defining serverless computing succinctly can be difficult because of the relative immaturity of the field and the fact that there's significant overlap with other cloud computing terms. What's worse, the term has really come to mean two different but overlapping areas. Earlier usage of the term refers to applications that depend on external cloud services, having their business logic concentrated on the client side – a development model described as Backend-as-a-Service (BaaS) or Mobile-Backend-as-a-Service (MBaaS). More recently the term has been extended to applications whose business logic is located in the cloud, and run in ephemeral compute containers managed by a 3rd party platform. This model is known as Function-as-a-Service (FaaS).

Fundamentally serverless computing is about building and running back-end code that does not require server management or server applications (Roberts 2016). Eyk et al. (2017) further define serverless computing by three key characteristics:

1. Granular billing: the user of a serverless model is charged only when the application is actually executing
2. (Almost) no operational logic: operational logic, such as resource management and autoscaling, is delegated to the infrastructure, making those concerns of the infrastructure operator
3. Event-Driven: interactions with serverless applications are designed to be short-lived, allowing the infrastructure to deploy serverless applications to respond to events, so only when needed

Fox et al. (2017) present a useful short definition of serverless as a cloud-native platform for short-running, stateless computation and event-driven applications which scales up and down instantly and automatically and charges for actual usage at a millisecond granularity.

As a sidenote, the earliest uses of term 'serverless' can be traced back to peer-to-peer software. The name has evolved into a completely different meaning in the current cloud computing context, so we're dismissing these earliest mentions.

The term 'serverless' in itself can be seen a bit disingenuous, since despite the name serverless computing obviously still involves servers. The name – coined by industry – instead implies that there's no longer need to spend time and resources on managing servers. Tasks such as provisioning, maintenance and capacity planning are handled by the serverless platform as needed by current workload. This leaves developers to focus on application logic, providing an abstraction where computation is disconnected from the infrastructure it is going to run on.

We see serverless architectures as making possible the idea of running complex, distributed applications built from relatively simple functions, without requiring the developer (the cloud customer) to manage servers (e.g., through elastic scheduling or auto-scaling) or complex operational aspects (e.g., authentication, authorization, and accounting). Specifically, we see serverless cloud architectures as defined by three key characteristics: (1) Granular billing: the user of a serverless model is charged only when the application is actually executing; (ii) (Almost) no operational logic: operational logic, such as resource management and autoscaling, is delegated to the infrastructure, making those concerns of the infrastructure operator; and (iii) Event-Driven: interactions with serverless applications are designed to be short-lived, allowing the infrastructure to deploy serverless applications to respond to events, so only when needed. (Eyk et al. 2017)

We see FaaS as tightly integrated with the notion of executing cloud functions, and, as such, a core component of serverless architectures. Useful especially for applications with narrow functionality, e.g., mobile applications, the (Mobile-)Backend-as-a-Service ((M)BaaS) focuses more than FaaS on operational logic, providing a more configurable, vendor- specific framework with many built-in components that simplify operation, such as user manage-

ment and data-storage[13]. Thus, we see BaaS as a more tailored, vendor-specific approach to FaaS. (Eyk et al. 2017)

FaaS in tandem with BaaS, good as a glue between external services

definition/separation is not clear-cut; for example Baldini, Castro, et al. (2017) sees MBaaS as a wholly different paradigm. Eyk et al. (2017) in turn note that some PaaS/SaaS offerings such as Auth0 and FaunaDB, although not generally considered serverless, exhibit the main characteristics of serverless.

Fundamentally serverless computing is about building and running back-end code that does not require server management or server applications. In the serverless paradigm applications are built of small, stateless *cloud functions* that are uploaded to a *serverless platform*. It is then up to the platform to execute, scale an handle billing in response to the exact demand needed at the moment.

why short running?

Historical/evolutionary views: 1) on-premises infra -> grid -> IaaS -> PaaS -> serverless, 2) bare metal -> VMs -> containers -> functions (Fox et al. 2017), 3) monolith -> SOA -> microservices -> serverless.

Unlike SaaS or PaaS that are always running, but scale on-demand, serverless workloads run on-demand, and consequently, scale on-demand. (Fox et al. 2017)

How does serverless relate to concepts such as FaaS, BaaS/MBaaS like Auth0 and Firebase, SOA, microservices, event-driven, virtualiation, containers, cloud-native, ...? Cloud-ready/cloud-native as defined by Pozdniakova and Mazeika (2017).

2.2 Background

The vision of computing as an utility can be dated back to at least 1969 and ARPANET. (Buyya et al. 2009)

Cloud computing is by now a well-established paradigm that enables organizations to flex-

ibly deploy their software systems over a pool of computing resources. (Jamshidi, Ahmad, and Pahl 2013)

The motivations behind cloud migration in general. Jamshidi, Ahmad, and Pahl (2013) identify cost saving, scalability, and efficient utilization of resources as well as flexibility as key drivers to migrate application to the cloud. Balalaie, Heydarnoori, and Jamshidi (2016) talk about the reasons behind migrating systems to cloud-native architectures.

Youseff, Butrico, and Silva (2008) with a detailed ontology of cloud computing. The recent evolution of cloud computing has borrowed its basics from several other computing areas and systems engineering concepts. Cluster and Grid Computing on one hand, and virtualization on the other hand are perhaps the most obvious predecessor technologies that enabled the inception of cloud computing. However, several other computing concepts have indirectly shaped today's cloud computing technology, including peer-to-peer (P2P) computing, SOA and autonomic computing.

Serverless platforms promise new capabilities that make writing scalable mi- croservices easier and cost effective, positioning themselves as the next step in the evolution of cloud computing architectures. (Baldini, Castro, et al. 2017)

reason behind cloud migration, why is cloud important? organizations migrate to cloud because of A, B, C... note reasons why NOT to migrate to cloud and whether they also apply to serverless.

Another particularly important reason is... Particular importance in the current energy-constrained environment. Data centers use on the order of 1-2% of global energy consumption, but server utilization rates are really low. (Horner and Azevedo 2016)

historically cloud has been driven by virtualization, IaaS to PaaS to CaaS...

serverless takes these benefits to the extreme... how does serverless work and lean on the existing technology, compare to other platforms

2.3 Use cases

Introduce the main/intended use cases for serverless, as well as the more esoteric applications in literature.

Malawski et al. (2017) find that while serverless infrastructures are designed mainly for processing background tasks of Web and IoT applications, the simple mode of operation makes this approach easy to use and promising in scientific workflows too. Jonas et al. (2017) argue that a serverless execution model with stateless functions can enable radically-simpler, fundamentally elastic, and more user-friendly distributed data processing systems. Spillner, Mateos, and Monge (2018) also find that in many domains of scientific and high-performance computing, solutions can be engineered based on simple functions which are executed on commercially offered or self-hosted FaaS platforms.

Introduce edge computing as a particularly strong driver for serverless. Glikson, Nastic, and Dustdar (2017) propose the novel paradigm of Deviceless Edge Computing that extends the serverless paradigm to the edge of the network, enabling IoT and Edge devices to be seamlessly integrated as application execution infrastructure. Nastic et al. (2017) present a novel approach implementing cloud-supported, real-time data analytics in serverless edge-computing applications. Baresi, Filgueira Mendonça, and Garriga (2017) propose a serverless architecture at the edge, bringing a highly scalable, intelligent and cost-effective use of edge infrastructure's resources with minimal configuration and operation efforts.

Fouladi et al. (2017) present a serverless video-processing framework. Yan et al. (2016) present the architecture and prototype of a chatbot using a serverless platform, where developers compose stateless functions together to perform useful actions. Ishakian, Muthusamy, and Slominski (2017) evaluate the suitability of a serverless computing environment for the inferencing of large neural network models. Ast and Gaedke (2017) describe an approach of how to utilize serverless computing to enable self-contained web components by deploying Web Component business logic as cloud-hosted functions.

2.4 Runtime

Briefly describe the inner workings of a FaaS runtime. Describe the two supported execution models, synchronous and asynchronous, and how they relate to application design. The former is used to build a typical request-response flow, e.g. a REST API endpoint, whereas the latter relates to pub-sub and other event-driven flows. Give examples on the kind of triggers supported by serverless platforms (HTTP calls, messaging, database events, ...).

Spillner (2017b) presents the design and implementation of a research-friendly FaaS runtime. McGrath and Brenner (2017) present the design of a novel performance-oriented serverless computing platform, discussing implementation challenges such as function scaling and container discovery, lifecycle, and reuse. Hendrickson et al. (2016) present OpenLambda, an open-source platform for serverless computation, describing the key aspects of serverless computation and presenting numerous research challenges that must be addressed in the design and implementation of such systems.

2.5 Service providers

Lynn et al. (2017) provide an overview and multi-level feature analysis of seven enterprise serverless computing platforms. Baldini, Castro, et al. (2017) also has a narrower survey of serverless platforms.

2.6 Economics of serverless

Eivy (2017) and Villamizar et al. (2016) both focus on the economic aspects of serverless. Adzic and Chatley (2017) explain how novel design patterns are used to significantly optimize costs – just running traditional web apps inside Lambda containers doesn't necessarily equate to savings. Adzic and Chatley (2017) also report savings between 66 and 95% in two case studies, and present a handly table comparing hosting prices for intermittent service tasks. Spillner (2017a) exploits the control plane of AWS Lambda to implement services practically for free. Leitner, Cito, and Stöckli (2016) present an approach to model deployment costs of AWS Lambda applications in real-time. Kuhlenkamp and Klems (2017)

present another cost-tracing system that enables per-request cost-tracing for cloud-based software services, noting that cost testing should not only rely on isolated tests of single services but consider comprehensive end-to-end cost traces.

2.7 Security

Address the security implications of serverless. Overall the consensus seems to be that compared to services maintaining their own servers and resources, serverless approach reduces the attack surface However, research needs to understand the new security issues introduced by FaaS. For example, because the infrastructure can share resources among cloud-functions, what is the ideal security vs. performance/cost trade-off? (Eyk et al. 2017)

2.8 Drawbacks and limitations

What to take into consideration when migrating to serverless?

Lloyd et al. (2018) analyze serverless performance and elasticity, identifying the cold start phenomenon. Differences in runtimes/languages, and larger library dependencies lead to slower starts. Oakes et al. (2017) address the problem by caching package dependencies on platform-level.

Baldini, Cheng, et al. (2017) identify three competing constraints in serverless function composition: functions should be considered as black boxes; function composition should obey a substitution principle with respect to synchronous invocation; and invocations should not be double-billed.

Roberts (2016), Adzic and Chatley (2017) and Baldini, Castro, et al. (2017) each list a number of limitations, including lack of strong SLA, vendor lock-in, short life-span, immature local development tools, statelessness and many others.

Kuhlenkamp and Klems (2017) discover two serverless cost tradeoffs: the retry cost effect and the cost ripple effect.

The need for circuit breakers (risk of DDoSing yourself) when interacting with non-serverless

components like a database. Mention novel cloud-native database services like Google's Cloud Spanner and AWS Aurora. Figure out a source for this – Hohpe and Woolf (2004) might have a relevant pattern.

Address maintainability: debugging serverless functions, following the flow of control can be tough.

Composing serverless functions is not like composing regular functions. All the difficulties of distributed computing – message loss, timeouts and others – apply and have to be handled. Possible solutions include retry policies, dead-letter queues and idempotent functions.

3 Serverless design patterns

Survey of serverless design patterns. Baldini, Castro, et al. (2017) put the question as follows:

> Will there be patterns for building serverless solutions? How do we combine low granularity basic building blocks of serverless into bigger solutions? How are we going to decompose apps into functions so that they optimize resource usage? For example how do we identify CPU-bound parts of applications built to run in serverless services? Can we use well-defined patterns for composing functions and external APIs? What should be done on the server vs. client (e.g., are thicker clients more appropriate here)? Are there lessons learned that can be applied from OOP design patterns, Enterprise Integration Patterns, etc.?

3.1 Serverless patterns

A limited number of purely serverless design patters. Sbarski and Kroonenburg (2017) introduce five patterns – Command, Messaging Priority queue, Fan-out and Pipes and filters – but they seem mostly to be reinterpretations of classic Enterprise Integration Patterns.

Common low-level design patterns such as function chaining and fanout in Github.

API Gateway and messaging patterns as described in platforms' own documentation. (Amazon 2018)

Adzic and Chatley (2017) suggest 3 methods for optimizing resource usage: use distributed authorization, let clients orchestrate workflows and allow clients to directly connect to AWS resources. The authors also discuss how paying only for actual utilization has two additional benefits of 1) removing incentives for bundling and 2) removing barriers to versioning: examples of serverless economics affecting architecture. Presenting these (or applications of these) as more formal patterns could be of some value.

Roberts (2016) asks how big can FaaS functions get before they get unwieldy? Assuming

we can atomically deploy a group of FaaS functions what are good ways of creating such groupings - do they map closely to how we'd currently clump logic into microservices or does the difference in architecture push us in a different direction? Extending this further what are good ways of creating hybrid architectures between FaaS and traditional 'always on' persistent server components?

Extending this further what are good ways of creating hybrid architect

Ast and Gaedke (2017) describe self-contained web components with serverless backends.

3.2 Enterprise Integration Patterns

Hohpe and Woolf (2004) present a number of asynchronous messaging architectures in the seminal book on EIP. While predating the whole serverless phenomenon the patterns are still relevant. Hohpe even demonstrated implementing one of his patterns on top of Google's serverless platform in a blog post.

3.3 FaaSification

Spillner (2017c) describes an automated approach to transform monolithic Python code into modular FaaS units by partially automated decomposition. Doesn't really seem suitable for the web application migration process covered in this thesis but worth mentioning.

4 Migration process

Implement a subset of the target app in a serverless style, utilizing the surveyed patterns and keeping log of the tricky parts. In case the patterns prove unsuitable for the given problem, try to come up with an alternative solution.

Describe the web application to migrate.

Decide on the parts to migrate. Should demonstrate the features and limitations of serverless as outlined above. Possible features include

- A simple REST API endpoint to showcase API Gateway and synchronous invocation. Shouldn't require any big changes to application code.
- Interaction between multiple services to demonstrate distributed transactions.
- A scheduled (cron) event.
- Interacting with an external SaaS service like Twilio, Auth0. Demonstrate event-driven invocation.
- others?

5 Evaluation

Evaluation the outcome of migration process. Estimate the effects on performance and hosting costs. Weigh in on maintainability, testability, developer experience etc.

6 Conclusion

What can we conclude about the research questions? Mention limitations and further research directions.

Bibliography

Adzic, Gojko, and Robert Chatley. 2017. “Serverless computing: economic and architectural impact”. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering,* 884–889. ACM.

Amazon. 2018. “AWS Lambda”. Visited on February 1, 2018. https://aws.amazon.com/lambda/.

Ast, Markus, and Martin Gaedke. 2017. “Self-contained Web Components Through Serverless Computing”. In *Proceedings of the 2Nd International Workshop on Serverless Computing,* 28–33. WoSC ’17. Las Vegas, Nevada: ACM. ISBN: 978-1-4503-5434-9. doi:10.1145/3154847.3154849. http://doi.acm.org/10.1145/3154847.3154849.

Balalaie, Armin, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. “Migrating to Cloud-Native Architectures Using Microservices: An Experience Report”. In *Advances in Service-Oriented and Cloud Computing,* edited by Antonio Celesti and Philipp Leitner, 201–215. Cham: Springer International Publishing. ISBN: 978-3-319-33313-7.

Baldini, Ioana, Paul C. Castro, Kerry Shih-Ping Chang, Perry Cheng, Stephen J. Fink, Vatche Ishakian, Nick Mitchell, et al. 2017. “Serverless Computing: Current Trends and Open Problems”. *CoRR* abs/1706.03178. arXiv: 1706.03178. http://arxiv.org/abs/1706.03178.

Baldini, Ioana, Perry Cheng, Stephen J. Fink, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Philippe Suter, and Olivier Tardieu. 2017. “The Serverless Trilemma: Function Composition for Serverless Computing”. In *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software,* 89–103. Onward! 2017. Vancouver, BC, Canada: ACM. ISBN: 978-1-4503-5530-8. doi:10.1145/3133850.3133855. http://doi.acm.org/10.1145/3133850.3133855.

Baresi, Luciano, Danilo Filgueira Mendonça, and Martin Garriga. 2017. “Empowering Low-Latency Applications Through a Serverless Edge Computing Architecture”. In *Service-Oriented and Cloud Computing,* edited by Flavio De Paoli, Stefan Schulte, and Einar Broch Johnsen, 196–210. Cham: Springer International Publishing. ISBN: 978-3-319-67262-5.

Buyya, Rajkumar, Satish Narayana Srirama, Giuliano Casale, Rodrigo N. Calheiros, Yogesh Simmhan, Blesson Varghese, Erol Gelenbe, et al. 2017. “A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade”. *CoRR* abs/1711.09123. arXiv: 1711.09123. http://arxiv.org/abs/1711.09123.

Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. 2009. “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”. *Future Generation Computer Systems* 25 (6): 599–616. ISSN: 0167-739X. doi:https://doi.org/10.1016/j.future.2008.12.001. http://www.sciencedirect.com/science/article/pii/S0167739X08001957.

CNCF. 2018. *Serverless whitepaper.* Technical report. Cloud Native Computing Foundation. Visited on February 7, 2018. https://github.com/cncf/wg-serverless.

Eivy, Adam. 2017. “Be Wary of the Economics of" Serverless" Cloud Computing”. *IEEE Cloud Computing* 4 (2): 6–12.

Eyk, Erwin van, Alexandru Iosup, Simon Seif, and Markus Thömmes. 2017. “The SPEC cloud group’s research vision on FaaS and serverless architectures”. In *Proceedings of the 2nd International Workshop on Serverless Computing,* 1–4. ACM.

Fouladi, Sadjad, Riad S Wahby, Brennan Shacklett, Karthikeyan Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. 2017. “Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads.” In *NSDI,* 363–376.

Fox, Geoffrey C., Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2017. “Status of Serverless Computing and Function-as-a-Service (FaaS) in Industry and Research”. *CoRR* abs/1708.08028. arXiv: 1708.08028. http://arxiv.org/abs/1708.08028.

Glikson, Alex, Stefan Nastic, and Schahram Dustdar. 2017. “Deviceless Edge Computing: Extending Serverless Computing to the Edge of the Network”. In *Proceedings of the 10th ACM International Systems and Storage Conference,* 28:1–28:1. SYSTOR ’17. Haifa, Israel: ACM. ISBN: 978-1-4503-5035-8. doi:10.1145/3078468.3078497. http://doi.acm.org/10.1145/3078468.3078497.

Google. 2018. “Google Cloud Functions”. Visited on February 7, 2018. https://cloud.google.com/functions/.

Hendrickson, Scott, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2016. “Serverless Computation with openLambda”. In *Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing,* 33–39. HotCloud’16. Denver, CO: USENIX Association. http://dl.acm.org/citation.cfm?id=3027041.3027047.

Hohpe, Gregor, and Bobby Woolf. 2004. *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* Addison-Wesley Professional.

Horner, Nathaniel, and Inês Azevedo. 2016. “Power usage effectiveness in data centers: overloaded and underachieving”. *The Electricity Journal* 29 (4): 61–69. ISSN: 1040-6190. doi:https://doi.org/10.1016/j.tej.2016.04.011. http://www.sciencedirect.com/science/article/pii/S1040619016300446.

IBM. 2018. “IBM Cloud Functions”. Visited on February 7, 2018. https://www.ibm.com/cloud/functions.

Ishakian, Vatche, Vinod Muthusamy, and Aleksander Slominski. 2017. “Serving deep learning models in a serverless platform”. *CoRR* abs/1710.08460. arXiv: 1710.08460. http://arxiv.org/abs/1710.08460.

Jamshidi, P., A. Ahmad, and C. Pahl. 2013. “Cloud Migration Research: A Systematic Review”. *IEEE Transactions on Cloud Computing* 1, number 2 (): 142–157. ISSN: 2168-7161. doi:10.1109/TCC.2013.10.

Jonas, Eric, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. 2017. "Occupy the Cloud: Distributed Computing for the 99%". *CoRR* abs/1702.04024. arXiv: 1702.04024. http://arxiv.org/abs/1702.04024.

Kuhlenkamp, Jörn, and Markus Klems. 2017. "Costradamus: A Cost-Tracing System for Cloud-Based Software Services". In *Service-Oriented Computing,* edited by Michael Maximilien, Antonio Vallecillo, Jianmin Wang, and Marc Oriol, 657–672. Cham: Springer International Publishing. ISBN: 978-3-319-69035-3.

Leitner, P., J. Cito, and E. Stöckli. 2016. "Modelling and Managing Deployment Costs of Microservice-Based Cloud Applications". In *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC),* 165–174.

Lloyd, Wes, Shruti Ramesh, Swetha Chinthalapati, Lan Ly, and Shrideep Pallickara. 2018. "Serverless Computing: An Investigation of Factors Influencing Microservice Performance". *The IEEE International Conference on Cloud Engineering (IC2E).* Forthcoming.

Lynn, Theo, Pierangelo Rosati, Arnaud Lejeune, and Vincent Emeakaroha. 2017. "A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms". In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom),* 162–169. IEEE.

Malawski, Maciej, Adam Gajek, Adam Zima, Bartosz Balis, and Kamil Figiela. 2017. "Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions". *Future Generation Computer Systems.* ISSN: 0167-739X. doi:https://doi.org/10.1016/j.future.2017.10.029. http://www.sciencedirect.com/science/article/pii/S0167739X1730047X.

McGrath, G., and P. R. Brenner. 2017. "Serverless Computing: Design, Implementation, and Performance". In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW),* 405–410. doi:10.1109/ICDCSW.2017.36.

Microsoft. 2018. "Microsoft Azure Functions". Visited on February 7, 2018. https://azure.microsoft.com/en-us/services/functions/.

Nastic, S., T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, and R. Prodan. 2017. "A Serverless Real-Time Data Analytics Platform for Edge Computing". *IEEE Internet Computing* 21 (4): 64–71. ISSN: 1089-7801. doi:10.1109/MIC.2017.2911430.

Oakes, E., L. Yang, K. Houck, T. Harter, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. 2017. "Pipsqueak: Lean Lambdas with Large Libraries". In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW),* 395–400. doi:10.1109/ICDCSW.2017.32.

Pozdniakova, Olesia, and Dalius Mazeika. 2017. "Systematic Literature Review of the Cloud-ready Software Architecture". *Baltic Journal of Modern Computing* 5 (1): 124.

Roberts, Mike. 2016. "Serverless Architectures". Visited on February 1, 2018. https://martinfowler.com/articles/serverless.html.

Sbarski, Peter, and S Kroonenburg. 2017. *Serverless Architectures on AWS: With examples using AWS Lambda.* Manning Publications, Shelter Island.

Spillner, Josef. 2017a. "Exploiting the Cloud Control Plane for Fun and Profit". *CoRR* abs/1701.05945. arXiv: 1701.05945. http://arxiv.org/abs/1701.05945.

———. 2017b. "Snafu: Function-as-a-Service (FaaS) Runtime Design and Implementation". *CoRR* abs/1703.07562. arXiv: 1703.07562. http://arxiv.org/abs/1703.07562.

———. 2017c. "Transformation of Python Applications into Function-as-a-Service Deployments". *CoRR* abs/1705.08169. arXiv: 1705.08169. http://arxiv.org/abs/1705.08169.

Spillner, Josef, Cristian Mateos, and David A. Monge. 2018. "FaaSter, Better, Cheaper: The Prospect of Serverless Scientific Computing and HPC". In *High Performance Computing,* edited by Esteban Mocskos and Sergio Nesmachnow, 154–168. Cham: Springer International Publishing. ISBN: 978-3-319-73353-1.

Varghese, Blesson, and Rajkumar Buyya. 2018. “Next generation cloud computing: New trends and research directions”. *Future Generation Computer Systems* 79:849–861. ISSN: 0167-739X. doi:https://doi.org/10.1016/j.future.2017.09.020. http://www.sciencedirect.com/science/article/pii/S0167739X17302224.

Villamizar, Mario, Oscar Garces, Lina Ochoa, Harold Castro, Lorena Salamanca, Mauricio Verano, Rubby Casallas, Santiago Gil, Carlos Valencia, Angee Zambrano, et al. 2016. “Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures”. In *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on,* 179–182. IEEE.

Walker, Mike J. 2017. “Hype Cycle for Emerging Technologies, 2017”. Visited on February 7, 2018. https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/.

Yan, Mengting, Paul Castro, Perry Cheng, and Vatche Ishakian. 2016. “Building a Chatbot with Serverless Computing”. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs,* 5:1–5:4. MOTA ’16. Trento, Italy: ACM. ISBN: 978-1-4503-4669-6. doi:10.1145/3007203.3007217. http://doi.acm.org/10.1145/3007203.3007217.

Youseff, L., M. Butrico, and D. Da Silva. 2008. “Toward a Unified Ontology of Cloud Computing”. In *2008 Grid Computing Environments Workshop,* 1–10. doi:10.1109/GCE.2008.4738443.