SoK: Virtualization Challenges and Techniques in Serverless Computing

Vasudha Devarakonda
vasudha.devarakonda@tamu.edu
Texas A&M University
College Station, Texas, USA

Aleksandr Earnest
arearnest7@tamu.edu
Texas A&M University
College Station, Texas, USA

Chia-Che Tsai
chiache@tamu.edu
Texas A&M University
College Station, Texas, USA

Abstract

This systematization of knowledge (SoK) paper summarizes the discussion of virtualization challenges and the corresponding techniques specific to serverless computing. We examine virtualization solutions, including paravirtualization, containers, lightweight hypervisors and kernels, and unikernels, and their applicability to serverless. Then, we discuss several challenges, including cold-start optimization, resource co-location, benchmarking and the research-production gap, hoping to inspire future research.

1 Introduction and Background

Serverless computing (denoted as "serverless" below) is one of the newest cloud computing concepts which shifted the relationship between cloud users and providers [13, 26]. This has led to wide adoption of serverless backends [39, 48] to many businesses as well as popularity among small-scale developers to host their infrastructure. However, serverless also introduces new challenges, especially since the model put more responsibility on the providers to virtualize and optimize the clusters. This paper aims to highlight the perspective of the research community on serverless, how virtualization is applied to serverless, and the remaining challenges not yet addressed by the community.

What is serverless? Serverless is a distributed computing paradigm, which can be best described with three primary properties[26, 53]:

- *Pay-per-invocation:* Serverless workloads incur no charge when there is no invocation and the development model is entirely event-driven.
- *Provider-managed stack:* Serverless eliminates the "micro-operations" of managing and monitoring the software stack and shifts the responsibility of deploying and scaling system runtimes to the providers.
- *Efficient scaling:* Some literature [26] mentions that serverless excels at scaling speed, at reportedly tens of thousands of invocations per second.

It was worth noting that serverless does not necessarily equate Function-as-a-Service (FaaS)[18, 35]. Serverless can be viewed as opposing to the traditional "serverful" paradigm, which requires reservation of instances and management by the users.

Serverless Applications and Workflows: Although single functions tend to serve the needs of most users, chained functions, i.e., workflows, are commonly used to build larger cloud applications comprised of smaller, self-contained functions [29, 37, 38, 53, 60]. Currently, workflow chaining either relies on calling the gateway for the downstream function or utilizes a side-car service such as Step Functions[5]. While uncommon, cyclical behavior[4] and fan-in-fan-out[7, 34, 54] communication does exist among workflows. Applications for workflows include image processing[37, 38, 61], machine learning[29, 38] and scientific workloads[54], but can also consist of simple backends with branching logic paths[21, 29, 43] expressed as function calls.

Contribution of this paper: This paper focuses on demystifying the use of virtualization, including containerization—which is a muddy domain of its own–in the context of serverless. Further, we summarize several observations regarding the key challenges in adopting virtualization for serverless and the potential solutions. Our hope is to draw attention to the system challenges in virtualizing a serverless framework and provide hints for future research.

2 Virtualization in Serverless

The textbook often describes virtualization as the process of creating virtual machines (VMs), but in reality, this topic can be *virtualization is a spectrum*. From virtual machines and containers to more specialized solutions such as MicroVM [1, 17, 33] and Unikernel [28, 45, 51], a variety of design choices and trade-offs have been explored in the literature, from the perspective of efficiency, security, compatibility, or other properties. Mainly, we examine virtualization techniques based on two aspects, on which many solutions have experimented with various degrees: (1) How much the guest kernel is modified to adapt to the hypervisor, or be merged as part of the host kernel; and (2) Whether the hypervisor multiplexes (emulates) IOs or lets them pass through. Next, we will discuss some major categories of solutions and their applicability to serverless. We illustrate the two spectra with Figure 1 and highlight the "sweet spots" for serverless.

The 2nd Workshop on Hot Topics in System Infrastructure (HotInfra'24), co-located with SOSP'24, November 3, 2024, Austin, TX, USA

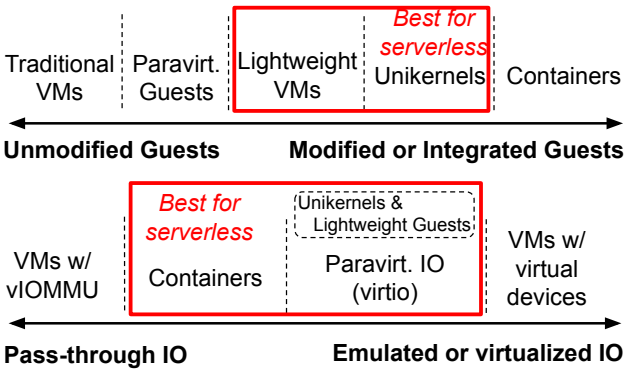

Figure 1. Showing virtualization techniques on two spectra, one on how much the guest is modified or integrated into the host, and the other on whether the IO requests from guests have been pass-through or emulated.

Virtualization and Paravirtualization: Traditionally, "serverful" applications (opposite of serverless) make use of virtual machines, with either type 1 (*bare-metal*) hypervisors [8–10] or type 2 (*hosted*) hypervisors [12, 46, 50], to facilitate multi-tenancy as a core property of cloud computing. The formal definition of virtualization by Popek & Goldberg [49] requires identicality between virtual and physical machines and thus can incur non-trivial overheads from trapping and emulating privileged operations in hypervisors. One of the more efficient choices is paravirtualization [8], which involves modifying the guests with hypercalls and cooperative IO paths (e.g., ring buffers) to streamline hypervisor emulation. For serverless applications, identicality is typically not a concern because the entire underlying infrastructure and OS is completely managed by the provider, so paravirtualized guests are commonly used. Furthermore, evictable VMs or VMs allocated on spared resources, such as spot instances [2] and Harvest VMs [20, 62], are suitable choices for hosting serverless applications given their ephemeral nature.

Containerization: Containers [19, 24] are lightweight sandboxes providing isolated OS and filesystem views for user applications. Container frameworks fuse two functionally distinctive layers: a packaging and deployment toolchain [24] and an OS layer for isolation and virtualization [30]. Containers became popular for serverless mainly due to its convenience and the expensive startup time and resource overheads of "traditional" VM. However, containers fundamentally offer only process level isolation and cannot provide the same level of security isolation as VMs because the shared host kernels[40, 56]. Solutions that suggest implementation of more namespaces also risk making the container implementation more complex and heavy as a regular VM[59]. The lack of security guarantees from containers leads to cloud providers stacking containers with virtual machines, causing even more resource wastes and startup delays [40].

"Lightweight" hypervisors and "Lightweight" guest kernels: Modern serverless platforms aim to provide isolation and cater multiple users on a single physical server. In addition, to maintain isolation between functions, they host one function or container per lightweight virtual machine [33]. Firecracker [1] and Cloud Hypervisors [32] designed an alternative to heavy-weight emulators like QEMU. Built on top of KVM to create virtual machines, they simplify the QEMU code base by eliminating unnecessary drivers and using virtio for I/O virtualization, making them compatible for "simple" serverless applications. LightVM [42] is another strategy that aims to replace the heavy hypervisors like Xen. It optimizes the creation of VMs by replacing XenStore with a shared memory that reduces the communication overhead between the guest VM and Dm0, which controls the creation of the virtual machines. Further optimization is carried out by splitting the creation of VM into two phases. The common stack between the VMs are pre created and kept in a pool and the configuration file is extracted and built on these existing shells [42]. RunD [33], adopted by Alibaba, identified the overheads due to rootfs in microVMs, virtio-fs and virtio-blk with respect to Kata Containers [17] which host containers on virtual machines. It divides the rootfs into read and write and mounts them into the microVM by ovelayfs to optimize the process of creation of containers when a function is invoked. Another alternative approach to avoid the overhead of creating a virtual machine is to use secure containers. gVisor [23], for instance, limits the number of system calls to the host OS to avoid information leaks by traditional containers [22]. It provides a lightweight "linux-like" interface on Linux kernels. Unlike virtual machines, gVisor does not provide hardware virtualization. Instead, it provides a layer that intercepts access to host OS. Gvisor sandbox consists of two main components; Sentry and Gofer. Sentry runs the kernel for the containers hosted in the sandbox. It narrows down the syscalls that can be made to the host by intercepting them to make the calls on behalf of the applications hosted on the container. Gofer handles the file system access which is associated with each container in the sandbox[23]. Though gVisor introduces security checks to containers it compromises the performance benefits[45].

While the above optimized hypervisors guarantee isolation and improve the creation of sandbox to deploy functions, the traditional Linux kernels are redundant and add high memory footprint per sandbox setup [33, 45]. Linux kernels have above 400 syscalls and multiple overlapping functions which increases the need for more security and isolation for multi tenant environments [42]. RunD [33] proposes a lightweight Linux kernel by removing some features that are redundant for serverless. Features like loop device, acpi, ftrace, graphics-related items, i2c, and ceph are disabled. This process reduced the kernel image footprint by 16 MB. Further, it identified that creation of cgroups cannot be parallelized and hence it maintains a pool of cgroups that are renamed

during sandbox creation. [42] proposed TinyX which is a "minimalistic" OS which aims to provide lower boot and creation time. It starts with a base overlayFS on Debian systems and installs necessary libraries for applications thus making it lightweight. Amazon Lambda also uses a lightweight Linux because of its extensive adoption and debugging [1, 11].

Unikernels: The above strategies focus on modification of the Linux kernel for serverless. However, there has been research towards replacing Linux kernels with Unikernels for serverless [28, 45, 51]. Unikernels have small resource footprint and provide single address space for the application and kernel while providing the isolation that traditional VMs offer [16, 44, 47]. The small address space also reduces the attack surface [40]. Unikernel Linux (UKL) [51, 52] proposes a Linux-based unikernel to retain its properties as legacy software. UKL diverts syscalls to its custom library since application is invoked in the kernel space. The work focuses on running general Linux compatible applications in unikernels , improving performance by running applications in kernel address space. USETL [16] designed a unikernel based serverless technique for ETL-style functions. The design simplifies the network and storage virtualization based on ephemeral and minimalistic nature of serverless. Urunc [40] is a unikernel container runtime that proposes the idea of creating unikernels for each function invocation using Knative stack with unikernel OCI images containing unikernel binaries. Evaluation is carried out with simple HTTP reply function and compared against Kata containers (with different hypervisors), generic containers and gVisor. The performance of urunc is similar to generic containers and is better than other mechanisms proving that unikernels offer similar performance as containers with better isolation in multi tenant environment. UniFaaS [44] suggests use of MirageOS, an application-based unikernel as an alternative to containers. The evaluations using IoT applications show that the startup time of unikernel based architectures are 3x better than the "warm" start of Openwhisk containers with minimal memory and CPU footprints.

Although Unikernels offer security with reduced trusted computing base (TCB) and isolation requirements of multi-tenant serverless applications, they are in a naive stage compared to Linux kernels for complete adoption. [44, 45, 51]. Moreover, since OS is bound to the application, we need a clear definition and requirements to replace traditional Linux kernel [28]. Further, single process nature makes debugging in unikernels complex with the requirement of a debugger attached to the hypervisor[45]. The current results favoring Unikernels over Linux-based kernels are promising. However, the testing is conducted over basic or no-op functions. There is still room for evaluations with complex benchmarks for serverless to motivate migration from Linux to Unikernels.

3 Open Challenges and Current Solutions

Serverless is yet to be optimized to the extent of other distributed frameworks. We summarized several notable research directions, and the attempts made by existing work to resolve the issues. However, all existing virtualization techniques have pros and cos in regards to solving these challenges, which we summarize in Table 1.

Cold Start Optimization: Given the virtualization paradigm of serverless, construction of functions requires cold starts when a replica is first deployed, thus is a latency cost worth optimizing. There are three primary forms of research into improving cold start, that being reducing the number of deployments via aggregation[29], reducing the cost of deployment via reuse of deployments or previous memory of invocations (i.e. caching or snapshot)[6, 55, 57], and improving resource scheduling via co-locating invocations[37]. While these solutions do provide better E2E latency, this comes at the cost of either further stressing the auto-scheduler under aggregation (i.e. extended packing problem) or causing co-location of invocations that could cause unintended security vulnerabilities given isolation assumptions while giving resource overhead for provider to keep the containers in a warm state[37, 53].

Co-Location of Downstream Resources: With the compounding of cold start and end-to-end latency of workflows, researchers have been utilizing techniques to aggregate, usually into monolithic designs[29, 37], or to co-locate downstream functions [25] with the attempt to reduce cold start and E2E latency. While these approaches are different, fundamentally the concept is to compile workflows into singular containers[29] or construct micro-services that handle the maximal volume of a request[37], removing inter-function latency. While the intentions are good, the naive approach of full aggregation does not account for the unpredictable resource load on nodes under bursty conditions, which can lead to resource contention[53]. Furthermore these strategies tend to rely on developers to perform the aggregation [29, 37], which is ill-suited due to the developer not knowing of resource load and placement on cluster and assumes near infinite resources[26, 53].

Debugging and Testing: Debugging and testing is essential for any developer for creating reliable, secure and efficient applications [27]. However, unlike monolithic and micro-service (serverful) applications serverless applications cannot be tested locally as it is challenging to replicate serverless environment [15, 58]. Strategies used for debugging in distributed systems are not suitable for serverless applications because of their ephemeral nature, limited control to infrastructure and lower response time expectations[27]. With no standardisation in logging and debugging tools, it becomes very cumbersome to track the root cause of the error in serverless applciations [58]. [15, 31] conducted interviews to list

Challenges for serverless	VMs (w or w/o paravirt.)	Lightweight Guests & Unikernels	Containers
Cold Start Optimization	+: Easier to checkpoint & restore -: Complexity and large VMs	+: Streamlined hypervisor & guest -: Limited compatibility	+: No booting of guest kernels -: Hard to further optimize the host kernel
Resource co-location	+: Mature live migration support -: Large base footprint (100s of MBs–GBs)	+: Small base footprint -: Specialized guest images may not be reusable or aggregateable	+: Shared kernel & user memory -: Poor migration support
Debugging & testing	+: Rich OS support in guests -: Too expensive for short-lived functions	+: Easy for reproduction -: Specialized guests may lack debugging support	+: Easiest for reproduction; DevOps integration -: No access to host kernel states
Research & benchmark	+: Well studied; many solutions & benchmarks -: Outdated for serverless	+: Cutting-edge research -: Proprietary/close-source; hand-crafted apps for motivation	+: Fully open-source ecosystem (Kubernetes & Docker) -: Weak security guarantee

Table 1. Comparison of pros (+, in blue) and cons (-, in red) of various virtualization techniques in regards to challenges specific to serverless.

the challenges for testing serverless applications. Cold starts variance with different runtime environments, simulation of network delays and unpredictable invocation patterns makes it particularly difficult for testing locally. Testing in the cloud is also challenging because it requires approval from cloud providers [15]. Therefore, a more standardised and mature debugging and testing mechanism for these short lived and distributed applications is necessary.

Benchmarking: Benchmarking is essential to research, especially computer science, with the primary reason is standardization of results between implementations and designs, however, serverless presents a unique problem of vendor lock-in[14] and very little standardized benchmarks existing[21, 41, 61]. This leads to researchers picking a selection of workloads familiar to them and adapting to work within the desired framework. Cherry picking in itself is undesirable, but the reason it occurs is because of the lack of knowledge on "in the wild" workloads to sample from due to trace data being the closest representation[36, 53], thus synthesized benchmarks are created and workloads are selected on need for arguments. Nuance of data movement, workflow patterns, cold start or even invocation rate is then left to the side for more "standardized" benchmarks which analyze performance of providers on strengths like small scale function concurrency, data transfer rates and cost, as trace data and workloads remain separate[36, 53]. The recommendation of this paper is for the creation of a framework agnostic benchmark suite that evaluates serverless functions and workflows from multiple disciplines such that proper analysis of workload limitations could be performed.

Research-Production Gap: As alluded to in previous sections, there is a gap in knowledge on what is used in production environments such as AWS, both in the sense of cluster information and in terms of "in the wild" functions and workflows[36, 53]. The reason why this information is obscured or withheld is rather innocent, having more to do with compliance with privacy laws[3]. This however does not excuse cluster providers from gathering willing participants to provide source code and payloads as samples, which can be done by surveys of developers on the cluster or through an opt-in toggle for each individual function. The means by which this information could be collected are numerous, but providing such information could improve understanding of the workloads in the wild and how researchers can improve architecture, including the provider architecture. Without this knowledge the community is reliant on itself to synthesize workloads based on what little data can be scraped from trace data regarding patterns, which is stripped of context of internal design and limitations.

4 Conclusion

From studying the literature, we conclude that traditional virtualization no longer serves the need for rapid deployment and high density of serverless. Using containers also leads to stacked virtualization layers due to security concerns. Specialized, lightweight hypervisors and guest kernels suit serverless better and may further resolve serverless challenges in cold-start optimization and resource co-location. More importantly, the research community needs better benchmarks and production frameworks to conduct meaningful experiments.

References

[1] Alexandru Agache, Marc Brooker, Andreea Florescu, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. 2020. Firecracker: lightweight virtualization for serverless applications. In *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation* (Santa Clara, CA, USA) *(NSDI'20)*. USENIX Association, USA, 419–434.

[2] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Assaf Schuster, and Dan Tsafrir. 2013. Deconstructing Amazon EC2 Spot Instance Pricing. *ACM Trans. Econ. Comput.* 1, 3, Article 16 (sep 2013), 20 pages. https://doi.org/10.1145/2509413.2509416

[3] Amazon. 2024. AWS Privacy. https://aws.amazon.com/privacy/.

[4] Amazon. 2024. Recursive patterns that cause run-away Lambda functions - AWS Lambda. https://docs.aws.amazon.com/lambda/latest/operatorguide/recursive-runaway.html.

[5] Amazon. 2024. Workflow Orchestration - AWS Step Functions - AWS. https://aws.amazon.com/step-functions/.

[6] Lixiang Ao, George Porter, and Geoffrey M. Voelker. 2022. FaaSnap: FaaS made fast using snapshot-based VMs. In *Proceedings of the Seventeenth European Conference on Computer Systems* (Rennes, France) *(EuroSys '22)*. Association for Computing Machinery, New York, NY, USA, 730–746. https://doi.org/10.1145/3492321.3524270

[7] awslabs. 2019. GitHub - awslabs/lambda-refarch-mapreduce. https://github.com/awslabs/lambda-refarch-mapreduce.

[8] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (Bolton Landing, NY, USA) *(SOSP '03)*. Association for Computing Machinery, New York, NY, USA, 164–177. https://doi.org/10.1145/945445.945462

[9] Broadcom. 2024. VMware ESXI. https://www.vmware.com/products/cloud-infrastructure/esxi-and-esx.

[10] Broadcom. 2024. VMware Vsphere. https://www.vmware.com/products/cloud-infrastructure/vsphere.

[11] Marc Brooker, Mike Danilov, Chris Greenwood, and Phil Piwonka. 2023. On-demand Container Loading in {AWS} Lambda. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. 315–328.

[12] Edouard Bugnion, Scott Devine, Mendel Rosenblum, Jeremy Sugerman, and Edward Y. Wang. 2012. Bringing Virtualization to the x86 Architecture with the Original VMware Workstation. *ACM Trans. Comput. Syst.* 30, 4, Article 12 (nov 2012), 51 pages. https://doi.org/10.1145/2382553.2382554

[13] Alibaba Cloud. 2024. What Is Serverless Computing? What are the Features of Serverless? https://www.alibabacloud.com/en/knowledge/what-is-serverless.

[14] Cloudflare. 2024. What is vendor lock-in? | Vendor lock-in and cloud computing. https://www.cloudflare.com/learning/cloud/what-is-vendor-lock-in/.

[15] Dilshan De Silva and Lakindu Hewawasam. 2024. The Impact of Software Testing on Serverless Applications. *IEEE Access* 12 (2024), 51086–51099. https://doi.org/10.1109/ACCESS.2024.3384459

[16] Henrique Fingler, Amogh Akshintala, and Christopher J. Rossbach. 2019. USETL: Unikernels for Serverless Extract Transform and Load Why should you settle for less?. In *Proceedings of the 10th ACM SIGOPS Asia-Pacific Workshop on Systems* (Hangzhou, China) *(APSys '19)*. Association for Computing Machinery, New York, NY, USA, 23–30. https://doi.org/10.1145/3343737.3343750

[17] Open Infrastructure Foundation. 2024. Kata Containers. https://katacontainers.io/.

[18] The Apache Software Foundation. 2024. OpenWhisk. https://openwhisk.apache.org/.

[19] The Linux Foundation. 2024. Kubernetes. https://kubernetes.io/.

[20] Alexander Fuerst, Stanko Novaković, Íñigo Goiri, Gohar Irfan Chaudhry, Prateek Sharma, Kapil Arya, Kevin Broas, Eugene Bak, Mehmet Iyigun, and Ricardo Bianchini. 2022. Memory-harvesting VMs in cloud platforms. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '22)*. Association for Computing Machinery, New York, NY, USA, 583–594. https://doi.org/10.1145/3503222.3507725

[21] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. 2019. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 3–18. https://doi.org/10.1145/3297858.3304013

[22] Xing Gao, Zhongshu Gu, Mehmet Kayaalp, Dimitrios Pendarakis, and Haining Wang. 2017. ContainerLeaks: Emerging Security Threats of Information Leakages in Container Clouds. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 237–248. https://doi.org/10.1109/DSN.2017.49

[23] gVisor. 2008. gVisor. https://gvisor.dev/docs/

[24] Docker Inc. 2024. Docker. https://www.docker.com/.

[25] Zhipeng Jia and Emmett Witchel. 2021. Nightcore: efficient and scalable serverless computing for latency-sensitive, interactive microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Virtual, USA) *(ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 152–166. https://doi.org/10.1145/3445814.3446701

[26] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, and David A. Patterson. 2019. Cloud Programming Simplified: A Berkeley View on Serverless Computing. arXiv:1902.03383 [cs.OS] https://arxiv.org/abs/1902.03383

[27] Shreyas Kharbanda and Pedro Fonseca. 2023. Always-On Recording Framework for Serverless Computations: Opportunities and Challenges. In *Proceedings of the 1st Workshop on SErverless Systems, Applications and MEthodologies* (Rome, Italy) *(SESAME '23)*. Association for Computing Machinery, New York, NY, USA, 41–49. https://doi.org/10.1145/3592533.3592810

[28] Ricardo Koller and Dan Williams. 2017. Will Serverless End the Dominance of Linux in the Cloud?. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems* (Whistler, BC, Canada) *(HotOS '17)*. Association for Computing Machinery, New York, NY, USA, 169–173. https://doi.org/10.1145/3102980.3103008

[29] Swaroop Kotni, Ajay Nayak, Vinod Ganapathy, and Arkaprava Basu. 2021. Faastlane: Accelerating Function-as-a-Service Workflows. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 805–820. https://www.usenix.org/conference/atc21/presentation/kotni

[30] Petros Koutoupis. 2018. Everything You Need to Know about Linux Containers, Part II: Working with Linux Containers (LXC). https://www.linuxjournal.com/content/everything-you-need-know-about-linux-containers-part-ii-working-linux-containers-lxc.

[31] Valentina Lenarduzzi and Annibale Panichella. 2021. Serverless Testing: Tool Vendors' and Experts' Points of View. *IEEE Software* 38, 1 (2021), 54–60. https://doi.org/10.1109/MS.2020.3030803

[32] LLC LF Projects. 2021. Cloud Hypervisor. https://www.cloudhypervisor.org/.

[33] Zijun Li, Jiagan Cheng, Quan Chen, Eryu Guan, Zizheng Bian, Yi Tao, Bin Zha, Qiang Wang, Weidong Han, and Minyi Guo. 2022. {RunD}: a lightweight secure container runtime for high-density deployment

and high-concurrency startup in serverless computing. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. 53–68.

[34] David H. Liu, Amit Levy, Shadi Noghabi, and Sebastian Burckhardt. 2023. Doing More with Less: Orchestrating Serverless Applications without an Orchestrator. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 1505–1519. https://www.usenix.org/conference/nsdi23/presentation/liu-david

[35] OpenFaaS Ltd. 2024. OpenFaas. https://www.openfaas.com/.

[36] Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. 2021. Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis. In *Proceedings of the ACM Symposium on Cloud Computing*. 412–426.

[37] Ashraf Mahgoub, Edgardo Barsallo Yi, Karthick Shankar, Sameh Elnikety, Somali Chaterji, and Saurabh Bagchi. 2022. ORION and the Three Rights: Sizing, Bundling, and Prewarming for Serverless DAGs. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. USENIX Association, Carlsbad, CA, 303–320. https://www.usenix.org/conference/osdi22/presentation/mahgoub

[38] Ashraf Mahgoub, Edgardo Barsallo Yi, Karthick Shankar, Eshaan Minocha, Sameh Elnikety, Saurabh Bagchi, and Somali Chaterji. 2022. WISEFUSE: Workload Characterization and DAG Transformation for Serverless Workflows. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 2, Article 26 (jun 2022), 28 pages. https://doi.org/10.1145/3530892

[39] Jesse Maida. 2019. Serverless Architecture Market Size Worth $9.17 Billion by 2023 - Technavio. https://www.businesswire.com/news/home/20190709005418/en/Serverless-Architecture-Market-Size-Worth-9.17-Billion-by-2023---Technavio.

[40] Charalampos Mainas, Ioannis Plakas, Georgios Ntoutsos, and Anastassios Nanos. 2024. Sandboxing Functions for Efficient and Secure Multi-tenant Serverless Deployments. In *Proceedings of the 2nd Workshop on SErverless Systems, Applications and MEthodologies* (Athens, Greece) *(SESAME '24)*. Association for Computing Machinery, New York, NY, USA, 25–31. https://doi.org/10.1145/3642977.3652096

[41] Pascal Maissen, Pascal Felber, Peter Kropf, and Valerio Schiavoni. 2020. FaaSdom: a benchmark suite for serverless computing. In *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems* (Montreal, Quebec, Canada) *(DEBS '20)*. Association for Computing Machinery, New York, NY, USA, 73–84. https://doi.org/10.1145/3401025.3401738

[42] Filipe Manco, Costin Lupu, Florian Schmidt, Jose Mendes, Simon Kuenzer, Sumit Sati, Kenichi Yasukata, Costin Raiciu, and Felipe Huici. 2017. My VM is Lighter (and Safer) than your Container. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) *(SOSP '17)*. Association for Computing Machinery, New York, NY, USA, 218–233. https://doi.org/10.1145/3132747.3132763

[43] Jason Mihalopoulos. 2019. Serverless Data Processing with AWS Step Functions — An Example. https://servian.dev/serverless-data-processing-with-aws-step-functions-an-example-6876e9bea4c0.

[44] Chetankumar Mistry, Bogdan Stelea, Vijay Kumar, and Thomas Pasquier. 2020. Demonstrating the Practicality of Unikernels to Build a Serverless Platform at the Edge. In *2020 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 25–32. https://doi.org/10.1109/CloudCom49646.2020.00001

[45] Felix Moebius, Tobias Pfandzelter, and David Bermbach. 2024. Are Unikernels Ready for Serverless on the Edge? *arXiv preprint arXiv:2403.00515* (2024).

[46] Oracle. 2024. Oracle VM VirtualBox. https://www.virtualbox.org/.

[47] Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe. 2015. Arrakis: The Operating System Is the Control Plane. *ACM Trans. Comput. Syst.* 33, 4, Article 11 (nov 2015), 30 pages. https://doi.org/10.1145/2812806

[48] Informa PLC. 2024. Omdia: serverless computing, valued at $19bn is the fastest-growing cloud service. https://omdia.tech.informa.com/pr/2024/jun/omdia-serverless-computing-valued-at-19-billion-dollars-is-the-fastest-growing-cloud-service.

[49] Gerald J. Popek and Robert P. Goldberg. 1974. Formal requirements for virtualizable third generation architectures. *Commun. ACM* 17, 7 (jul 1974), 412–421. https://doi.org/10.1145/361011.361073

[50] Avi Qumranet, Yaniv Qumranet, Dor Qumranet, Uri Qumranet, and Anthony Liguori. 2007. KVM: The Linux virtual machine monitor. *Proceedings Linux Symposium* 15 (01 2007).

[51] Ali Raza, Parul Sohal, James Cadden, Jonathan Appavoo, Ulrich Drepper, Richard Jones, Orran Krieger, Renato Mancuso, and Larry Woodman. 2019. Unikernels: The next stage of linux's dominance. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. 7–13.

[52] Ali Raza, Thomas Unger, Matthew Boyd, Eric B Munson, Parul Sohal, Ulrich Drepper, Richard Jones, Daniel Bristot De Oliveira, Larry Woodman, Renato Mancuso, Jonathan Appavoo, and Orran Krieger. 2023. Unikernel Linux (UKL). In *Proceedings of the Eighteenth European Conference on Computer Systems* (Rome, Italy) *(EuroSys '23)*. Association for Computing Machinery, New York, NY, USA, 590–605. https://doi.org/10.1145/3552326.3587458

[53] Mohammad Shahrad, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, Boston, Massachusetts, USA, 205–218. https://www.usenix.org/conference/atc20/presentation/shahrad

[54] Vaishaal Shankar, Karl Krauth, Kailas Vodrahalli, Qifan Pu, Benjamin Recht, Ion Stoica, Jonathan Ragan-Kelley, Eric Jonas, and Shivaram Venkataraman. 2020. Serverless linear algebra. In *Proceedings of the 11th ACM Symposium on Cloud Computing* (Virtual Event, USA) *(SoCC '20)*. Association for Computing Machinery, New York, NY, USA, 281–295. https://doi.org/10.1145/3419111.3421287

[55] Wonseok Shin, Wook-Hee Kim, and Changwoo Min. 2022. Fireworks: a fast, efficient, and safe serverless framework using VM-level post-JIT snapshot. In *Proceedings of the Seventeenth European Conference on Computer Systems* (Rennes, France) *(EuroSys '22)*. Association for Computing Machinery, New York, NY, USA, 663–677. https://doi.org/10.1145/3492321.3519581

[56] Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. 2019. Container Security: Issues, Challenges, and the Road Ahead. *IEEE Access* 7 (2019), 52976–52996. https://doi.org/10.1109/ACCESS.2019.2911732

[57] Dmitrii Ustiugov, Plamen Petrov, Marios Kogias, Edouard Bugnion, and Boris Grot. 2021. Benchmarking, analysis, and optimization of serverless function snapshots. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Virtual, USA) *(ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 559–572. https://doi.org/10.1145/3445814.3446714

[58] Jinfeng Wen, Zhenpeng Chen, and Xuanzhe Liu. 2022. Software engineering for serverless computing. *arXiv preprint arXiv:2207.13263* (2022).

[59] Dongjin Yu, Yike Jin, Yuqun Zhang, and Xi Zheng. 2019. A survey on security issues in services communication of Microservices-enabled fog applications. *Concurrency and Computation: Practice and Experience* 31, 22 (2019), e4436.

[60] Minchen Yu, Tingjia Cao, Wei Wang, and Ruichuan Chen. 2023. Following the Data, Not the Function: Rethinking Function Orchestration in Serverless Computing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 1489–1504. https://www.usenix.org/conference/nsdi23/presentation/yu

[61] Tianyi Yu, Qingyuan Liu, Dong Du, Yubin Xia, Binyu Zang, Ziqian Lu, Pingchao Yang, Chenggang Qin, and Haibo Chen. 2020. Characterizing serverless platforms with serverlessbench. In *Proceedings of the 11th*

ACM Symposium on Cloud Computing (Virtual Event, USA) *(SoCC '20)*. Association for Computing Machinery, New York, NY, USA, 30–44. https://doi.org/10.1145/3419111.3421280

[62] Yanqi Zhang, Íñigo Goiri, Gohar Irfan Chaudhry, Rodrigo Fonseca, Sameh Elnikety, Christina Delimitrou, and Ricardo Bianchini. 2021. Faster and Cheaper Serverless Computing on Harvested Resources. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles* (Virtual Event, Germany) *(SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 724–739. https://doi.org/10.1145/3477132.3483580