The Case for a New Cloud-Native Programming Model with Pure Functions

Ana Klimovic
ETH Zurich
aklimovic@ethz.ch

Cloud evolution: Over the past two decades, the cloud has become the dominant platform for running all kinds of applications, from data analytics to web services. In the process, cloud platforms have evolved from renting virtual machines (VMs) on-demand to offering elastic compute and storage services. While the ability to support legacy applications was critical in the early days of cloud to ease migration from on-premise, today's users commonly develop *cloud-native* applications by composing cloud storage services (e.g., S3), compute services (e.g., AWS Lambda), data analytics services (e.g., BigQuery), machine learning services (Azure ML), and elastic databases (e.g., Snowflake [4]). With this approach, users no longer need to explicitly provision CPU/memory/storage for their applications, as the elastic services automatically scale-out based on load and bill users for the resources consumed [7].

Opportunity and obstacle: By abstracting resource management from users, elastic cloud services have the potential to optimize resource allocation, task scheduling, and data movement under the hood to improve overall performance and energy-efficiency. Multi-tenant cloud services like AWS S3 and Lambda can optimize resource allocation with a global view across users [8].

However, a major optimization obstacle is that today's cloud programming model captures very little about the resource requirements and data access patterns of individual applications, leaving cloud services with little information to apply optimizations. Despite new cloud-native models like Functions as a Service (FaaS), today's cloud is still built around the principle of executing *opaque*[1] user applications inside VMs. For example, FaaS platforms execute a user function as an opaque unit in a MicroVM [1]. Each serverless function arbitrarily combines custom computation logic and calls to external cloud services for data passing. The platform is not aware of inter-function nor inter-service dependencies, making it difficult to optimize task scheduling and data prefetching. As a result, serverless functions often spend a large fraction of their execution time blocked on I/O [5]. To avoid idling CPU cores while functions block, the platform can multiplex many VMs per core. However, context switching securely between VMs adds latency [2] and comes with a high memory footprint, as the platform must allocate the total memory needed for all in-flight VMs.

Rethink the programming model: A promising way to enable cloud platforms to improve performance and resource efficiency is to rethink the cloud-native programming model, such that users develop applications in ways that provide the cloud platform with key information to guide task scheduling and data prefetching optimizations.

We propose a programming model that strictly separates compute tasks (custom user logic) and I/O tasks (interactions between cloud services). In this new paradigm, users express applications by composing two types of functions: 1) *pure compute functions*, i.e., untrusted user code snippets that compute exclusively on declared inputs and produce declared outputs and 2) *I/O functions*, i.e., trusted code implemented by the platform and exposed to users as a library, enabling interaction with other services, like storage.

Separating compute and I/O has several benefits. First, it makes application dataflow explicit to the platform, enabling data prefetching and task scheduling optimizations [3, 11]. For example, the platform can colocate functions that need to exchange data and allocate CPU cores and memory to functions only when their inputs are ready. Second, separating I/O tasks (which require interaction with the operating system and hence have a large attack surface) from other user code enables executing user code with more lightweight isolation mechanisms than canonical VMs [9, 10] to improve performance. Finally, separating computation and I/O in the programming model simplifies offloading each type of task to hardware accelerators, as accelerators are typically specialized for either fast computation or fast I/O. We are currently exploring these ideas in *Dandelion* [6], a new serverless platform.

[1] *Opaque* execution refers to execution with no awareness of application characteristics, such as data dependencies.

References

[1] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. "Firecracker: Lightweight Virtualization for Serverless Applications". In: *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 2020.

[2] Amazon Web Services. *The Security Design of the AWS Nitro System*. https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/security-design-of-aws-nitro-system.html. 2022.

[3] Ankit Bhardwaj, Meghana Gupta, and Ryan Stutsman. "On the Impact of Isolation Costs on Locality-aware Cloud Scheduling". In: *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*. 2020.

[4] Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Motivala, Abdul Q. Munir, Steven Pelley, Peter Povinec, Greg Rahn, Spyridon Triantafyllis, and Philipp Unterbrunner. "The Snowflake Elastic Data Warehouse". In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 215–226. ISBN: 9781450335317. DOI: 10.1145/2882903.2903741. URL: https://doi.org/10.1145/2882903.2903741.

[5] Yuhan Deng, Angela Montemayor, Amit Levy, and Keith Winstein. "Computation-Centric Networking". In: *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. HotNets '22. 2022.

[6] Tom Kuchler, Michael Giardino, Timothy Roscoe, and Ana Klimovic. "Function as a Function". In: *Proceedings of the 2023 ACM Symposium on Cloud Computing*. SoCC '23. 2023.

[7] Ingo Müller, Renato Marroquín, and Gustavo Alonso. "Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD '20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 115–130. ISBN: 9781450367356. DOI: 10.1145/3318464.3389758. URL: https://doi.org/10.1145/3318464.3389758.

[8] Vivek Narasayya and Surajit Chaudhuri. "Multi-Tenant Cloud Data Services: State-of-the-Art, Challenges and Opportunities". In: *Proceedings of the 2022 International Conference on Management of Data*. SIGMOD '22. Philadelphia, PA, USA: Association for Computing Machinery, 2022, pp. 2465–2473. ISBN: 9781450392495. DOI: 10.1145/3514221.3522566. URL: https://doi.org/10.1145/3514221.3522566.

[9] Vasily A Sartakov, Lluís Vilanova, David Eyers, Takahiro Shinagawa, and Peter Pietzuch. "CAP-VMs: Capability-Based Isolation and Sharing for Microservices". In: *Proceedings of Operating Systems Design and Implementation*. OSDI '22. 2022.

[10] Simon Shillaker and Peter Pietzuch. "Faasm: Lightweight Isolation for Efficient Stateful Serverless Computing". In: *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. 2020.

[11] Minchen Yu, Tingjia Cao, Wei Wang, and Ruichuan Chen. "Following the data, not the function: Rethinking function orchestration in serverless computing". In: *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 2023, pp. 1489–1504.