Towards a Holistic Cloud System with End-to-End Performance Guarantees

Remo Andreoli* and Tommaso Cucinotta†
Sant'Anna School of Advanced Studies
Pisa, Italy
name.surname@santannapisa.it

* PhD Candidate
† Supervisor

***Abstract*—Computing technologies are undergoing a relentless evolution from both the hardware and software sides, incorporating new mechanisms for low-latency networking, virtualization, operating systems, hardware acceleration, smart services orchestration, serverless computing, hybrid private-public Cloud solutions and others. Therefore, Cloud infrastructures are becoming increasingly attractive for deploying a wider and wider range of applications, including those with more and more stringent timing constraints, like the emerging use case of deploying time-critical applications. However, despite the availability of a number of public Cloud offerings, and of products (or open-source suites) for deploying in-house private Cloud infrastructures, still there are no solutions readily available for managing time-critical software components with predictable end-to-end timing requirements in the range of hundreds or even tens of milliseconds. The goal of this discussion is to present the multi-domain challenges associated with orchestrating a holistic Cloud system with end-to-end guarantees, which is the subject of my current PhD investigations.**

***Index Terms*—Cloud Orchestration, Cloud System, End-to-End Performance**

I. Introduction & Background

The emerging use case of deploying time-critical applications in the Cloud is a challenging task due to stringent requirements in terms of system availability and service quality. Contrary to conventional cloud-based use cases, the correctness of a time-critical system depends not only on the computation's results but also on the time at which the individual results are produced. Time-criticality covers a plethora of heterogeneous use cases: from industrial control systems to media streaming platforms [2]. Depending on the "criticality" degree of the use case, a disruption or lateness in service response times may lead to simple customer dissatisfaction (i.e., stuttering in a virtual reality multiplayer environment) as well as catastrophic consequences in the physical world (i.e., a cloud-controlled autonomous vehicle that detects a pedestrian too late). As of today, no cloud provider stipulates a Service-Level Agreement (SLA) for individual request guarantees, but rather for a number of requests over a wide time period (i.e., months, years). For instance, AWS DynamoDB [10] and Google BigTable [9] are two fully-managed cloud service that claim to guarantee single-digit millisecond per-request latency; nevertheless, their SLAs are defined in terms of the "usual" monthly service availability[1,2].

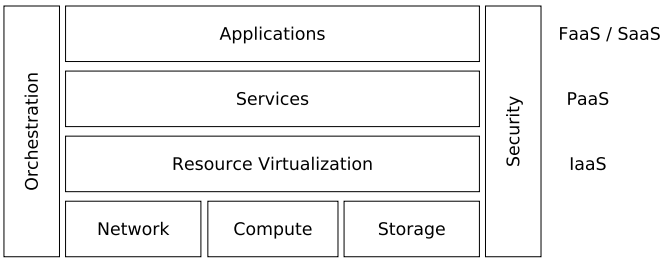

Fig. 1. Cloud computing architecture stack.

The adoption of cloud technologies for time-criticality is hampered by two problems: A) performance uncertainties are inherent of today's conventional cloud and networking infrastructures; B) the goal of typical cloud orchestration operations (e.g., placement, provisioning, run-time scaling, and monitoring) is to maximize throughput and utilization of shared resources. Such a best-effort Quality-of-Service (QoS) approach does not apply to time-critical applications, which require more granular, per-request QoS guarantees. The two problems are highly correlated: proper orchestration is unfeasible without a cloud infrastructure with time-critical capabilities; improved predictability is worthless without adequate resource orchestration. The construction of a comprehensive cloud system that incorporates time-critical guarantees across the $entire$ cloud stack (Figure 1), requires addressing the two problems to the same extent.

Many academic studies focus on improving the predictability of cloud services and infrastructures (problem A) and provide integrations with production-grade cloud technologies. Conversely, very few papers investigate proper orchestration for the time-critical use case (problem B), nor do they provide practical implementations on real-world cloud orchestrators. For instance, every orchestration tool deploys one-by-one the components of a complex application, without taking

[1]https://aws.amazon.com/dynamodb/sla/
[2]https://cloud.google.com/bigtable/sla

into account structural requirements (or just supporting the specification of simple affinity/anti-affinity constraints); this may lead to a suboptimal initial placement which is hardly able to guarantee a specific response time. On a similar note, there is a lack of proper monitoring tools for correcting a time-critical deployment at run-time. My current research direction is focused on filling these gaps on the orchestration side. The rest of this section briefly presents the current state-of-the-art for problem A (Section I-A) and problem B (Section I-B), including my contributions to both categories.

A. Performance predictability

There has been plenty of academic research investigating the predictability of system performance at different levels of the cloud stack. At the infrastructure level, the recent developments in the 5G community have been instrumental to the emergence of deterministic networking technologies for end-to-end latencies, such as Time-sensitive Networking (TSN) [11]. A lot of academic works integrate predictable real-time CPU scheduling and resource isolation for popular hypervisors and container orchestrators, such as rt-Xen [19] and rt-Kubernetes [12]. On the storage area, the zoned storage model [7] for flash-based SSDs promises an optimized I/O subsystem for isolation, predictable latency, intelligent data placement, and I/O scheduling.

At the service layer, there has been a similar focus on integrating time-criticality within the services' software. For instance, I worked on rt-MongoDB [3]–[5], a modified version of MongoDB which incorporates differentiated performance capabilities. If such feature get integrated in mainline, a cloud provider offering a cloud-hosted database service can use it to serve higher-priority requests with shorter response times and less variance, with respect to lower-priority requests. Naturally, a time-sensitive workload will be labeled as high-priority, whereas a batch-processing workload will be labeled as low-priority.

B. Time-Critical orchestration

There have been some works focusing on different steps of the orchestration procedure for time-critical applications in cloud environments [8], [15], [17]. Cloud-native applications are often represented as directed acyclic graphs (DAG), whose composing activities are often represented as microservices. However, most of these works provide overly complex user-defined parameters in terms of task-to-task communication bandwidth, hardware/software requirements, and so on. In reality, a cloud user overestimates, or is simply unaware, of the precise resource requirements of his/her application. This leads to resource and money waste, as well as misunderstandings with the cloud provider. The cloud user should only specify the topology that constitutes its cloud-native application, along with the required end-to-end response time for fulfilling a single user request. The cloud provider should be entrusted with providing the optimal deployment decisions that minimize the violation of such requirements at design-time and run-time. Unsurprisingly, Serverless computing seems to be the right paradigm to deal with time-criticality, as it relieves the customers from the burden of provisioning virtual resources. There are some foundational works [16], [18] on the topic, but none of them consider complex, distributed application topologies. Moreover, fault-tolerance and the related repercussions on service availability are often neglected in these works, despite their relevance in real-world cloud platforms (and for the proper functioning of time-critical applications). In this regard, I contributed to a novel performance model for time-critical, fault-tolerant cloud architectures [1], [6] in collaboration with Ericsson Research. Our work focuses on estimating at design-time the worst-case response time for a set of microservice-based time-critical applications by taking into account the worst-case workload patterns, as well as the occurrence of faults.

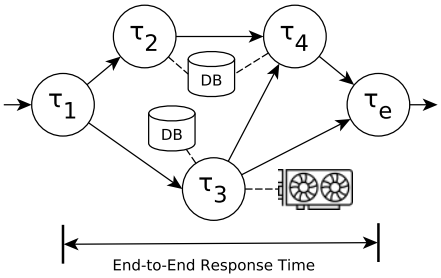

Fig. 2. Example of distributed cloud-native application.

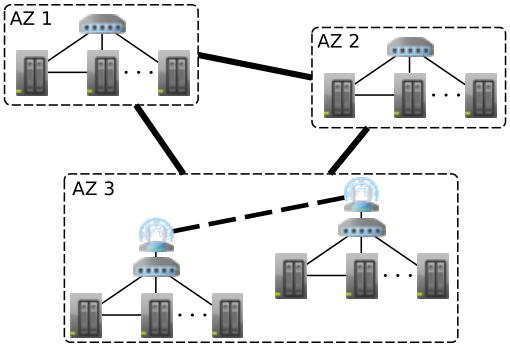

Fig. 3. Example of a distributed cloud infrastructure divided into independent availability zones.

There is also a lack of recent work on proper monitoring and actuation methods to guarantee at run-time high availability and scalability for time-critical workloads. Conventional threshold-based approaches are increasingly becoming too cumbersome due to the complexity and sheer size of modern distributed cloud infrastructures. An example is Khaleq et al. [13], which propose a Machine Learning (ML) method to predict scaling actions and improve response times at run-time.

II. Future Works

This section describes the next steps of my research direction. My current goal is to incorporate novel deployment strategies for distributed applications with temporal constraints on a widely adopted, open-source cloud orchestrator like Kubernetes. As with previous works, a cloud-native application is represented as a DAG of cloud-based activities. A single activity may be a simple computational job, but also a query to a database service with time-critical capabilities, such as rt-MongoDB. An activity may also require specialized hardware, such as an accelerator (Figure 2). The number of requirements specified by the user is minimal and mainly in terms of individual response times and service availability guarantees. Therefore, following the Serverless paradigm, the whole orchestration burden is left to the cloud provider. A (non-exhaustive) list of duties that the cloud provider must perform: i) estimate the worst-case response for each activity, based on the application topology; ii) plan in advance the provisioning of the virtual resources and their placement on the underlying infrastructure (such as the one in Figure 3), so that the user-defined request response times are guaranteed; iii) infer if such performance requirements are currently feasible; iv) monitor and timely react to fault conditions; v) monitor and dynamically adapt the deployment in order to withstand traffic bursts with minimal disservice to the time-critical applications. My previous work, briefly described in Section I-B, is a step towards facilitating some of these duties, mainly i), ii) and iii). However, there is still much work to be done to properly characterize the underlying infrastructure, especially network-wise, by introducing geo-distributed infrastructures, such as edge and fog nodes. I tackled the deployment problem using approaches based on Mixed-Integer Linear Programming (MILP) or Mixed-Integer Quadratic Program (MIQCP). However, the solving time exponentially increases with the size of the underlying infrastructure, and it becomes even more cumbersome with probabilistic formulations. Therefore, I'm planning to steer away from traditional solvers providing exact solutions, in favor of heuristics and ML-based approaches. Furthermore, there is also a growing interest in sustainable orchestration techniques for energy efficiency; however, I have not yet investigated in this direction.

Regarding the monitoring part of orchestration, I collaborated in the creation of an ML-based detection system for cloud operations in OpenStack, extending the work in [14]. The idea is to overcome the limitations of traditional, threshold-based detection systems by automatically applying corrective actions in response to observed anomalies. However, such work does not address time-criticality.

References

[1] Luca Abeni, Remo Andreoli, Harald Gustafsson, Raquel Mini, and Tommaso Cucinotta. Fault tolerance in real-time cloud computing. In *2023 IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC)*, 2023.

[2] Fredrik Alriksson, Lisa Boström, Joachim Sachs, Y-P Eric Wang, and Ali Zaidi. Critical IoT connectivity Ideal for Time-Critical Communications. *Ericsson technology review*, 2020(6):2–13, 2020.

[3] Remo Andreoli and Tommaso Cucinotta. Differentiated performance in nosql database access for hybrid cloud-hpc workloads. In *High Performance Computing: ISC High Performance Digital 2021 International Workshops, Frankfurt am Main, Germany, June 24–July 2, 2021, Revised Selected Papers 36*, pages 439–449. Springer, 2021.

[4] Remo Andreoli, Tommaso Cucinotta, and Daniel Bristot De Oliveira. Priority-driven differentiated performance for nosql database-as-a-service. *IEEE Transactions on Cloud Computing (accepted and to be published)*.

[5] Remo Andreoli, Tommaso Cucinotta, and Dino Pedreschi. Rt-mongodb: A nosql database with differentiated performance. In *CLOSER*, pages 77–86, 2021.

[6] Remo Andreoli, Harald Gustafsson, Luca Abeni, Raquel Mini, and Tommaso Cucinotta. Design-time analysis of time-critical and fault-tolerance constraints in cloud services. In *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*, 2023.

[7] Matias Bjørling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R Ganger, and George Amvrosiadis. Zns: Avoiding the block interface tax for flash-based ssds. In *USENIX Annual Technical Conference*, pages 689–703, 2021.

[8] Antonio Brogi, Stefano Forti, and Ahmad Ibrahim. Optimising qos-assurance, resource usage and cost of fog application deployments. In *Cloud Computing and Services Science: 8th International Conference, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018, Revised Selected Papers 8*, pages 168–189. Springer, 2019.

[9] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2), jun 2008.

[10] Mostafa Elhemali, Niall Gallagher, Bin Tang, Nick Gordon, Hao Huang, Haibo Chen, Joseph Idziorek, Mengtian Wang, Richard Krog, Zongpeng Zhu, et al. Amazon DynamoDB: A Scalable, Predictably Performant, and Fully Managed NoSQL Database Service. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 1037–1048, 2022.

[11] Norman Finn. Introduction to time-sensitive networking. *IEEE Communications Standards Magazine*, 2(2):22–28, 2018.

[12] Stefano Fiori, Luca Abeni, and Tommaso Cucinotta. Rt-kubernetes: containerized real-time cloud computing. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 36–39, 2022.

[13] Abeer Abdel Khaleq and Ilkyeun Ra. Intelligent autoscaling of microservices in the cloud for real-time applications. *IEEE Access*, 9:35464–35476, 2021.

[14] Giacomo Lanciano, Filippo Galli, Tommaso Cucinotta, Davide Bacciu, and Andrea Passarella. Extending OpenStack Monasca for Predictive Elasticity Control. *Big Data Mining and Analytics*, 2023.

[15] Li Liu, Qi Fan, and Rajkumar Buyya. A deadline-constrained multi-objective task scheduling algorithm in mobile cloud environments. *IEEE Access*, 6:52982–52996, 2018.

[16] Hai Duc Nguyen, Chaojie Zhang, Zhujun Xiao, and Andrew A. Chien. Real-Time Serverless: Enabling Application Performance Guarantees. In *Proceedings of the 5th International Workshop on Serverless Computing*, WOSC '19, page 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

[17] Polona Štefanič, Matej Cigale, Andrew C Jones, Louise Knight, Ian Taylor, Cristiana Istrate, George Suciu, Alexandre Ulisses, Vlado Stankovski, Salman Taherizadeh, et al. Switch workbench: A novel approach for the development and deployment of time-critical microservice-based cloud-native applications. *Future Generation Computer Systems*, 99:197–212, 2019.

[18] Mark Szalay, Peter Matray, and Laszlo Toka. Real-time faas: Towards a latency bounded serverless cloud. *IEEE Transactions on Cloud Computing*, 2022.

[19] Sisu Xi, Justin Wilson, Chenyang Lu, and Christopher Gill. Rt-xen: Towards real-time hypervisor scheduling in xen. In *Proceedings of the ninth ACM international conference on Embedded software*, pages 39–48, 2011.