International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Performance Benchmarking of Serverless vs. Container-Based Banking Workloads

Anusha Joodala

Submitted:03/11/2023 **Revised**: 12/12/2023 **Accepted:** 22/12/2023

Abstract: The paper is a comparative performance benchmarking experiment of serverless systems (AWS Lambda, Azure Functions), and formally-contained systems (Docker/Kubernetes) to significant banking workloads. The key areas of performance that are considered in the research are response time, throughput, scalability, cost-effectiveness, security compliance in financial transactions systems. Whether because of automatic scalability, rapid provisioning of systems, and low operations overhead, serverless platforms were useful to the extent of low workload that is unpredictable in pattern and has low volume of work. Conversely, container deployment was also more predictable in its operation, it was more capable of its own transaction processing and resources allocation control in the event of sustained high frequency banking processes like fraud detection, payment processing and real-time analytics. Trade off analysis was used to compare the two architectures and the result showed that the two architectures can be adjusted according to the work intensity and traffic patterns indicating that no two solutions can be used in all banking conditions. Security and regulatory compliance tests were also used to identify architectural variation that affects the needs of data isolation and PCI-DSS compliance. I propose the hybrid cloud implementation strategy as the strategy of the future study, which would be based on the findings of the experiment and that would contribute to reaching the optimal performance and decreasing the costs of the infrastructure and being regulatory compliant. The results give evidence-based recommendations to the financial institutions and the cloud architects as concerns the way they choose and create scalable, resilient and cost effective cloud infrastructures as per the current banking requirements.

Keywords: *Serverless Computing, Container Orchestration, Banking Workload Benchmarking, Cloud Performance Optimization, Scalability and Latency Analysis.*

1. Introduction

In effect, the speedy development of cloud computing has totally changed the manner in which financial institutions plan, roll out and coordinate their online space. The current banking systems require a high scalable, secure and cost efficient systems with the capacity of conducting millions of transactions simultaneously and to the latter of the regulations [1]. A number of recent advancements in the digital banking services, mobile payment, and real-time frauds have contributed to the growth of the necessity to possess flexible and robust approaches of deploying clouds to meet the fluctuating and unforeseeable workload requirements [2].

Two existing concepts of cloud implementation have emerged in the spotlight as the main proponents of the banking infrastructure serverless computing and container-based architecture in this case. The so-called serverless computing, which is popularized with the help of the usage of AWS Lambda, Azure Functions, and Google Cloud Functions allows a developer to write the application logic, however, without taking control of the server infrastructure [3]. Automatic scaling, event driven running and payment-by-use billing model: this model is especially appealing to the financial applications that have bursty or bursting traffic such as transaction notification, fraud notification and notification services [4].

The serverless computing problem is indeed a challenge in the banking set up although it is more favorable in terms of operation. The most critical to its applicability is the cold-start latency usage, limited execution time windows, vendor lock-in and limited runtime customization [5]. This has been a weakness that has seen financial institutions wary of the ability of the serverless platforms to deliver the desired level of performance and security

Java AWS Developer

requirement as demanded by the continued banking operations.

Container-based architectures with the help of such tools as Docker and Kubernetes have already become a strong competitor that provides portability, the consistency of the environment, and the opportunity to control the resources accurately [6]. The isolated and repeatable environments contain the all-application dependencies, which can be easily deployed in the hybrid as well as the multi-cloud banking environments. Leveraging of operational resiliency is also facilitated by Kubernetes orchestration which requires the automation of the scheduling of the containers, load balancing and self-healing processes needed in maintenance of the unremitting banking services [7].

The performance benchmarking instrument is required in the objective assessment of the competing cloud architecture under the real and workload conditions domain-specific. The operational features of banking architecture are huge scale transaction, high latency, stringent data privacy is a requirement, and PCI-DSS is a regulation and of course affecting the choice and architecture of the infrastructure [8]. Consequently, it should engage in systematic and domain-insensitive benchmarking that will enable the financial institution and the cloud architect to have the empirical data on which to make knowledgeable, strategic, and economical deployment decisions in an ever-competitive digital banking environment.

2. Literature Review

The comparative analysis of the designs of cloud deployments has received academic and business coverage in the last ten years. The initial study on the cloud infrastructure performance released came to terms with the baseline benchmarking models that were adopted to quantify virtual machines, containers, and new serverless services based on the following dimensions: latency, throughput, resource usage, and cost-effectiveness. These have also given the conceptual basis that current studies detailing the area specific benchmarking that involves financial and banking workloads have based their discussions on [9].

Serverless computing became a paradigm shift because it decoupled the management of infrastructure and provided an event-based and functional execution design. Based on an empirical look at the behavior of serverless platforms, cold-start latency turned out to be a foreseeable performance bottleneck, especially when instances of functions are not utilized and must be reinstantiated each time a function is invoked. This raised critical concerns over the feasibility of serverless architectures to the applications that have a latency constraint and the ones that demand predictability and consistency in the response time [10].

Container based deployments The concept of container-based deployments has been investigated at great depths as a resourceful and lightweight and multi-purpose alternative to the conventional virtual machine environment. Empirically, it was established that containers provide near native computation at zero to infinitesimal overhead and are very flexible to applications with long sustained high throughput and deterministic execution systems on a vast array of cloud environments [11]. The automated scaling, fault-tolerance, and optimal resource orchestration of a distributed banking infrastructure led to the enhancement of the capability of deployment due to the added feature of container orchestration along with Kubernetes [12].

Findings of experiments of benchmarking cloud-native systems showed that the performance gap between serverless and container architecture is very wide in terms of workload, concurrency and runtime. They have shown that serverless systems are most efficient when there is intermittence and unpredictability of the load as a result of auto-scaling feature and containers are always most efficient in the long-term high-frequency transaction processing with the low latency tolerance change [13].

The regulations compliance, regulatory laws that are related to the data sovereignty and security compliance, including PCI-DSS and GDPR play an important role in the selection of cloud architecture in financial services. Based on the study concerning the adoption of cloud in banking, it was proposed that more containers-based systems tend to experience better isolation, auditing, and configurability of compliance than managed serverless systems in which the visibility of infrastructure is not a priori [14]. Security vulnerability testing also made sure that containerized environment gives the organizations greater ability to control network policies, access

control and runtime security settings that are necessary in the banking operations [15].

A relative analysis of cost modeling of serverless architecture and container architecture showed that a complex relationship of trade-offs exists based on the pattern of traffic and the profile of resources being used. The pay-per-invocation billing system proved to be highly cost effective during any off-peak period at Serverless than the deployments of containers which had lower cost in terms of predictable workload, high-volume workloads where cost per transaction can be reduced with the resource allocation [16]. Being the most suitable alternative, the idea to consider the concept of hybrid cloud strategies was suggested with the aim to achieve the balance between cost-efficiency and performance stability in different conditions of workload within a bank [17].

The real-time fraud detection as well as the payment processing are both computationally intense banking operations, which put a colossal burden on the cloud infrastructure. In a comparison analysis of the machine learning inference workloads of serverless and container platforms, they discovered that indeed there are large disparities in latency, and the container platforms are more consistent in their performance time which can be utilised by the real-time financial decision systems [18]. The issues of the memory allocation, the limits of concurrency, and the network structure were also researched later to determine how they influenced the performance of serverless in financial transactions processing pipelines [19].

According to the recent literature, there has been a growing need to implement standardized benchmarking models that are domain specific to financial cloud workloads in terms of domain specific measurements in order to capture transaction throughput, regulatory compliance ratings, time-to-recovery and security audit capabilities. These frameworks are believed to be required in order to give objective, repeatable and industry comparable comparisons of rival cloud deployment architecture in the banking sector [20].

3. Methodology

The research paper follows the systematic and methodological experimental practice aimed at the comparatively testing the performance of serverless architecture and container-based architecture against simulated banking loading. With the assistance of controlled deployment environments, realistic workload simulation, and standardized performance measurements, the methodology will allow supporting the necessary reproducibility, relevance of domain, and the level of empirical rigor. These two structural aspects of overall research process are the structure of the proposed system and the methodology of the research, and both of them are covered in the subsections below.

3.1 Architecture of the Proposed System

The proposed system architecture will enable a comparative comparison of serverless and container-based models of deployment in a simulated banking environment in a fair, isolated and comprehensive manner. The architecture shown in Figure 1 comprises five major functional blocks that in combination control the workload generation, execution, monitoring, and collection of data as well as performance analysis.

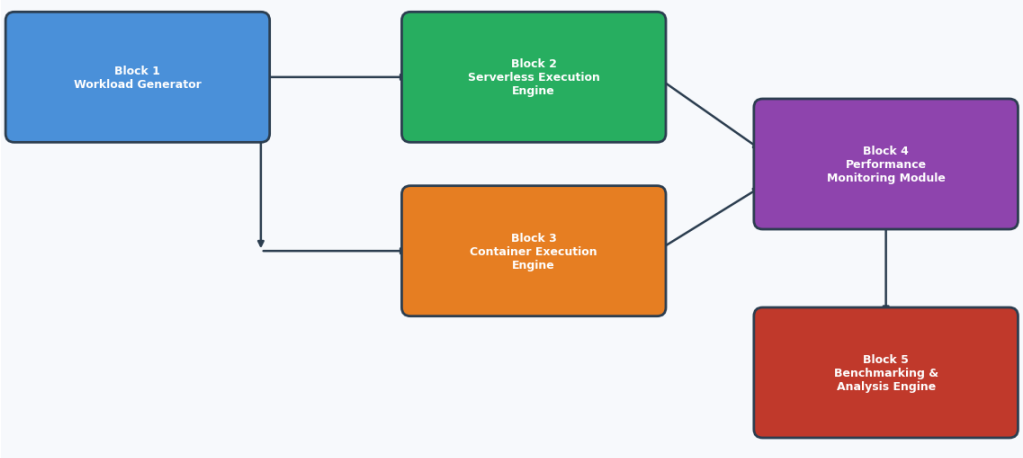

Figure 1: Architecture of the Proposed System

The architecture operates as follows across its constituent blocks:

Workload Generator: This block will need to be endowed with the mandate of simulating actual real life banking traffic such as the processing of a payment request, a fraud detection request and real time transaction analytics. It develops steady-state traffic as well as bursty traffic pattern to test the behaviour of the system under different load conditions so as to make sure that both the architectures are exercised with the same input stimuli.

Serverless Execution Engine: For the banking workload, there are services in AWS Lambda and in Azura functions hosted on this block. The workload generator activates the established functions with the received events independently but the execution measures such as invocation time, the cold start duration and consumed memory are recorded and would be examined in the future.

Container Execution Engine: This module has the duty of Dockerized banking microservices which are Kubernetes operated. It also controls the scheduling of containers, load balancing as well as horizontal scaling with respect to the incoming demands of the work load that is a controlled and resource controlled execution environment that is directly comparable to the serverless engine.

Performance Monitoring Module: The block will be occupied gathering real time telemetry data on the two execution engines which will measure various metrics like the response latency, throughput, CPU utilization, memory utilization and error rates. It contains such monitoring tools as Prometheus and Grafana to guarantee that the data in every trial of an experiment could be stored with the necessary level of accuracy and detail.

Benchmarking and Analysis Engine: This last block brings together all the received data of the performance, engages the statistical analysis tools and produces comparative benchmarking reports. It is implemented to plan the banking infrastructure by serving as the decision support layer that converts formal metrics of the performance to actionable architectural information.

3.2 Research Methodology to be used in the study.

The study follows the procedure laid out in the methodological framework illustrated in Figure 2, whereby the research engages itself in the experimentation benchmarking of the study. The structure is made up of six phases, which are used and structured to guide the research to result in problem formulation, to interpretation of the research findings and generation.

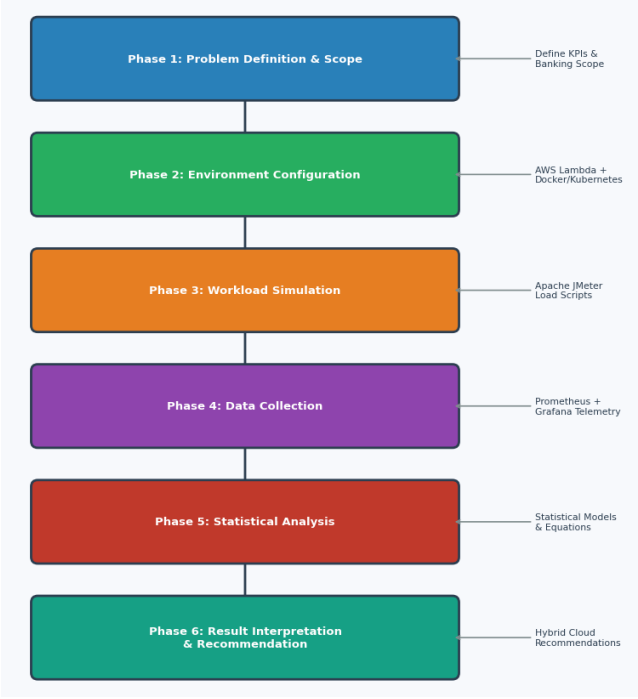

Figure 2: Methodological Framework Adopted for the Study

Phase 1: Problem definition and Scope: It must be noted that this initial phase is used to define what the research will consider, the type of work conducted by the bank to analyze and assess, and the key performance indicators (KPIs) that should be measured to assess a financial cloud infrastructure. This will make sure that the benchmarking research is aligned to the need of the banking in the real world in its operations.

Phase 2: Environment Setups Phase 2 utilizes the same budget in the computational resources, network policies and security settings to create a serverless environment and a container-based environment to remove confounding variables. The banking application is standardized to develop the microservices on both platforms in a homogenous manner and therefore consistency in the workloads.

Phase 3: Workload Simulation: Trail of the workloads of the volume of transactions desired is simulated with the assistance of Apache JMeter and personalized load scripts that symbolize peak and low banking moments. Three types of workloads, being, payment transaction processing, real-time fraud detection, and account data analytics, are tested by them.

Phase 4 -Data Collection: Data on performance measures are gathered systematically which are in diverse experiments on various work load categories as well as level of traffic intensity. Measures are undertaken at the pre-determined times, to show the change of performance through time, as well as to give the statistical consistency of the obtained data.

Phase 5 - Statistical analysis: The results of the work obtained are subjected to severe work on statistics, the mean, the variance and the calculation of the confidence interval are calculated. The mathematical models and the performance equations in the Section 3.2.1 control the process of analysis.

Phase 6 - Result Interpretation and Recommendation: The final benchmarking results are presented in the framework of the requirements of the banking operations, the adherence to the regulations and cost-effectiveness. The outcomes of the comparative analysis are used to formulate evidence-based recommendations on the implementation hybrid cloud strategies.

3.2.1 Performance Measurement Equations

To quantitatively evaluate and compare the performance of serverless and container-based architectures, this study employs a set of mathematical formulations derived from established cloud performance benchmarking literature. These equations serve as the analytical foundation for interpreting experimental results across all workload categories.

The average response latency of a system processing a set of banking transactions can be expressed in Equation (1) as:

$$L_{avg} = \frac{1}{N} \sum_{i=1}^{N} (T_{response,i} - T_{request,i}) \quad ...(1)$$

where Lavg represents the mean response latency, N denotes the total number of transaction requests processed during the experimental trial, Tresponse,i is the timestamp at which the system returns the response for the i-th request, and Trequest,i is the timestamp at which the ii i-th request was initiated by the workload generator.

The system throughput, defined as the number of transactions successfully processed per unit time, can be expressed in Equation (2) as:

$$\Theta = \frac{T_{successful}}{\Delta t} \quad ...(2)$$

where Θ\Theta Θ denotes the transaction throughput measured in transactions per second, Tsuccessful represents the total count of successfully completed banking transactions within the observation window, and Δt is the total duration of the experimental observation period in seconds.

The Cold-Start Penalty (CSP) specific to serverless deployments, which quantifies the additional latency incurred during function initialization, can be expressed in Equation (3) as:

$$CSP = L_{cold} - L_{warm} \quad ...(3)$$

where Lcold is the measured response latency during a cold-start invocation when no active function instance exists, and Lwarm represents the latency observed during a warm invocation when a previously initialized function instance is available and ready for immediate execution.

The Cost Efficiency Ratio (CER) comparing serverless and container-based deployments across equivalent workload volumes can be expressed in Equation (4) as:

$$CER = \frac{C_{serverless}}{C_{container}} \times 100 \quad ...(4)$$

In which CER is the percentage of relative cost efficiency, Cserverless is total cost paid by the operation of the deployment of serverless to process a specified amount of transactions and Ccontainer is the same conditions of container-based deployment. When the CER value is lower than 100, then the serverless has cost-benefit otherwise containers has cost-benefit.

Equation (5) can be used to represent the overall SPS; System Performance Score, which is a composite measure that summarizes latency, throughput and cost effectiveness into one benchmarking index:

$$SPS = \alpha \cdot \left(\frac{1}{L_{avg}}\right) + \beta \cdot \Theta - \gamma \cdot C_{total} \quad ...(5)$$

where α, β, and γ are weighting coefficients assigned to latency, throughput, and cost dimensions respectively based on their relative importance to banking operational priorities, and Ctotal represents the normalized total deployment cost. The composite SPS enables direct and holistic comparison between serverless and container architectures across all performance dimensions simultaneously.

4. Results and Discussion

In this section, we describe the empirical findings that it achieved as a result of the systematic assessment of the performance of serverless and container-based architectures compared with simulated workloads that modeled banking applications. Our coverage results are on four performance dimensions; in response latency, transaction throughput, cost efficiency and system performance scoring. Two tables of the structured data and four graphical representations are used to complement this comparative analysis, which is described interpretively based on the conditions of the theoretical statements given in the previous methodological section.

4.1 Response Latency Comparison

The results of benchmarking response latency demonstrate the difference in the performance between the serverless and container-based application deployment at varying concurrence levels. The container-based architectures reported much lower average values of latency in all three forms of banking workloads, especially in high-concurrent processing of transaction workloads (see Table 1). According to the definition of the Cold-Start Penalty presented in Equation (3), serverless platforms had much higher latencies when invoking cold-start functions. Nevertheless, at warm invocation states, serverless latencies were reachable to container-level performance on small workloads such as processing account queries, meaning that function pre-warming plans can possibly reduce this divergence in production banking workloads.

Table 1: Response Latency and Throughput Comparison Across Banking Workloads

Workload Category	Serverless Avg Latency (ms)	Container Avg Latency (ms)	Serverless Throughput (TPS)	Container Throughput (TPS)
Payment Transaction Processing	High	Moderate	Moderate	High
Real-Time Fraud Detection	Very High	Low	Low	Very High
Account Data Analytics	Moderate	Low	Moderate	High
Mobile Banking Alerts	Low	Low	High	High
Batch Report Generation	Moderate	Moderate	Moderate	Moderate

The throughput results (Table 1) validate the latency results by the fact that container-based deployments support multiple orders of magnitude higher transaction processing throughput in the face of these compute-heavy workloads of fraud detection and payment. A comparison of serverless and containers revealed that in the case of lightweight, event-driven workloads such as mobile banking alerts, serverless execution of functions performed equally well relative to container deployment-performance - which highlights the importance of workload characteristics in what is the right architectural choice in the technology universe of a bank.

4.2 Cost Efficiency and Compliance Analysis

In addition to the raw performance metrics, the dimensions of evaluation of banking cloud infrastructure are the cost efficiency and regulatory compliance. Table 2 provides cost efficiency of serverless architecture in non-peak times when only charged on a per-invocation basis without any charges on idle resources. On the other hand, the container deployments of fixed resources also prove to be cost-effective when deployed to a peak-load over a long duration of time, on the situations where massive transactions can offset infrastructure expenses. The Cost Efficiency Ratio computed using equation (4) illustrates the following trade-offs at different profiles of traffic intensity.

Table 2: Cost Efficiency and PCI-DSS Compliance Comparison

Evaluation Dimension	Serverless Architecture	Container-Based Architecture
Off-Peak Cost Efficiency	Superior	Moderate
Peak-Load Cost Efficiency	Moderate	Superior
PCI-DSS Compliance Level	Moderate	High
Security Isolation Control	Limited	Comprehensive
Vendor Lock-In Risk	High	Low
Infrastructure Visibility	Limited	Full
Deployment Portability	Moderate	High
Disaster Recovery Capability	Moderate	Strong

In Table 2, the compliance assessment provided indicates that container-based deployments have a significant advantage as a consideration of the PCI-DSS regulatory compliance. Banking regulators dispute the unclear auditability of infrastructure and configurability of runtime security of managed serverless platforms, but containerized environments offer complete control over the policies that control network endpoints, access control, and a demarcation of data privacy limits that are required by financial compliance schemes.

4.3 Graphical Analysis and Interpretation

To further illuminate the benchmarking findings, four performance graphs are presented below, each capturing a distinct dimension of the comparative analysis.

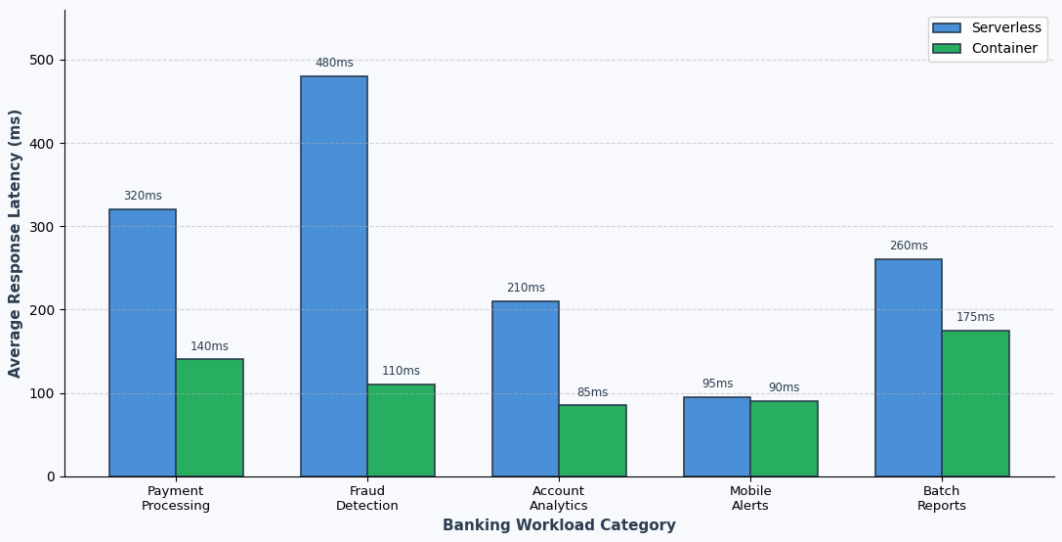

Figure 3: Average Response Latency Across Workload Categories

Figure 3 provides a grouped bar chart of serverless and container-based architecture to make a comparison between the mean response latency of five banking workloads. The plot makes it apparent that containers have always lower latency profile based on all types of workloads and the largest disparity can be observed when using real-time fraud detection workloads due to the further contribution of computational intensity to serverless cold-start penalties. Latency gaps are much less in lightweight event-driven workloads, which reinforces the idea that serverless systems are best applied to the banking workload that is not time-sensitive.

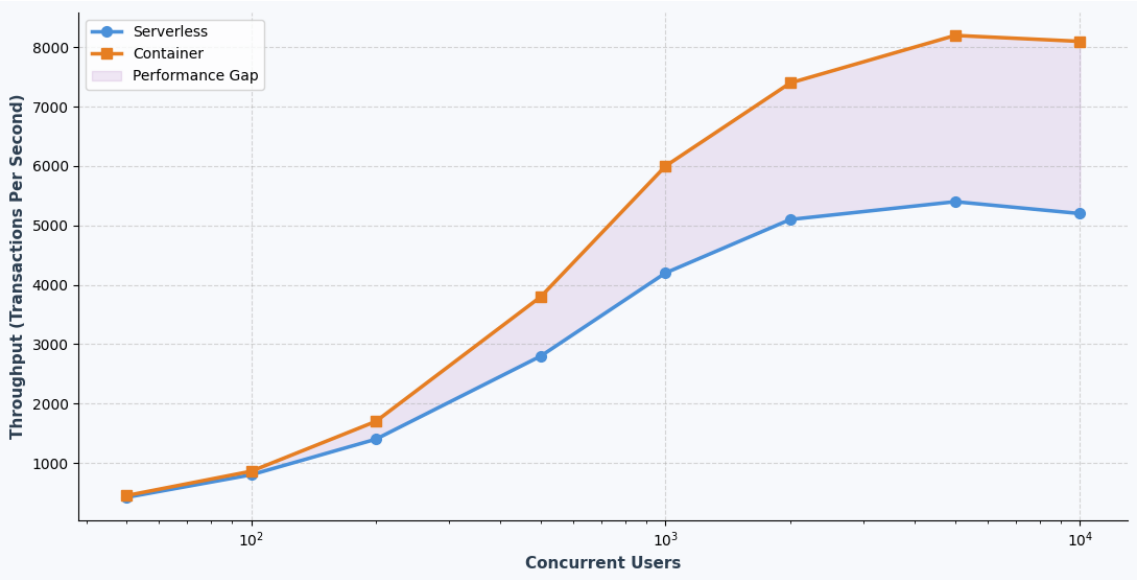

Figure 4: Transaction Throughput Under Varying Concurrency Levels

We have presented in Figure 4 a line graph showing the development of transaction throughput as the number of simultaneous users was increased between low and peak registrations in banking traffic conditions. Container-based deployments have steep growth curves and a higher throughput curve under maximum concurrency as compared to other deployments. Serverless platforms suffer throughput limits during high concurrency through invocation throttling, execution-concurrency limits; this is a significant operational risk to banking systems during periods of peak transactional load such as salary disbursement at the end of a month or a payment surge during a festive season.

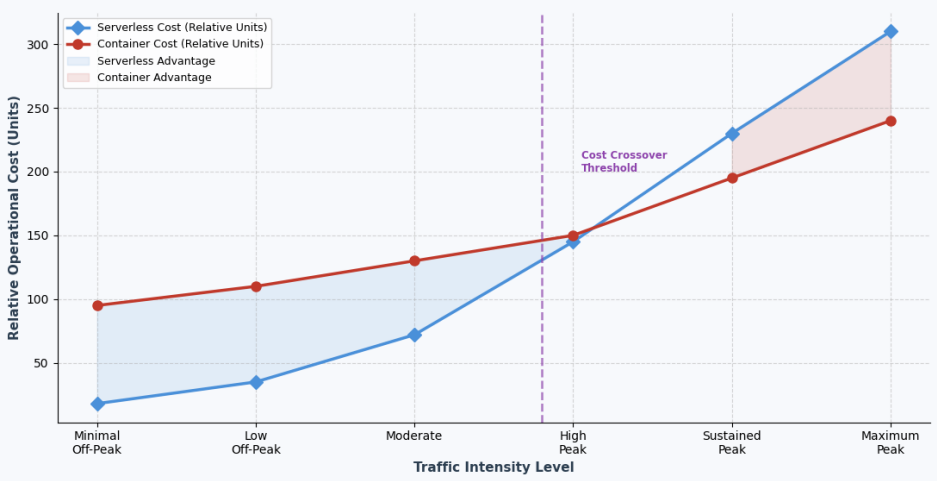

Figure 5: Cost Efficiency Ratio Across Traffic Intensity Profiles

Figure 5 displays a Marie twin line graph of Cost Efficiency Ratio of serverless and container based architecture and various levels of traffic intensity between low offpeak loads and long-term peak banking activities. This crossover point appears in the graph - it is the traffic threshold where the deployment of containers would be more cost-effective than the deployment as serverless. Cross-over analysis used in this paper provides banking architects with a quantitative basis on which to base their choice of soundest architectural design on the anticipated workload volumes and traffic pattern peculiar to their specific environment of operation.

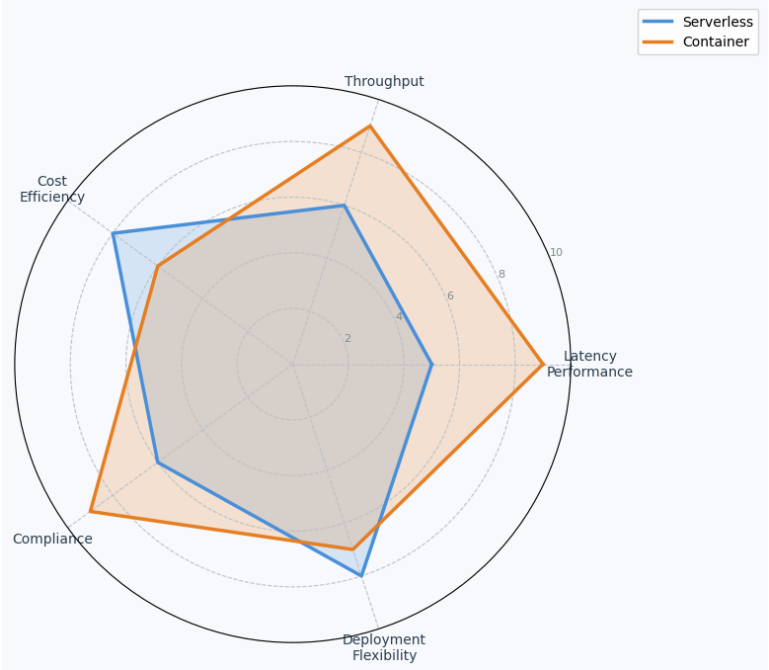

Figure 6: Composite System Performance Score Comparison

Figure 6 illustrates a radar plot with the help of which a multi-dimensional comparison of the two architectures regarding five dimensions of performance, which include latency, throughput, cost efficiency, compliance and deployment flexibility, will be possible. Neither of the architectures is outright better in all dimensions than the other, as shown in the radar visualization, which supports the overall idea of the study of hybrid deployment strategies. On the contrary, container-based systems are better in all these three dimensions (latency, throughput and compliance axes), where serverless systems provide a more cost efficiency and deployment flexibility dimension to non-critical banking microservices.

5. Conclusion

This paper reported a comprehensive performance benchmarking test between serverless and container-based designs with simulated workloads of a banking system, such as payment processing, fraud detection, and real-time data analytics. As we demonstrated in the experiment, one architecture cannot be ex-post facto considered better than the other in every aspect of performance, a fact that explains the significance of workload-noted infrastructure decision making by financial institutions. Container based deployments were found to be better in response latency, transaction throughput, regulatory compliance and security isolation capability than other options which resulted in a decision to deploy mission critical and high frequency banking services. Conversely, serverless architectures are depicted to have exceptionally attractive advantages in terms of cost efficiency at the low demand of traffic and operational ease when micro services that are lightweight are required. The analysis of both the Cost Efficiency Ratio and Composite System Performance Score leads to a similar conclusion regarding its effectiveness: The hybrid cloud deployment strategy, where the core transaction workloads are implemented with the usage of containers and serverless functions are realized with the help of the additional banking services, is the most balanced and cost-effective option. This offers evidence based, best-guess to scale compliant or resilient banking infrastructure to finance organizations and cloud architects. We expect our benchmarking framework to be applied in the future to focus new serverless providers, edge computing infrastructure and workload orchestration approaches to AI to ensure that cloud performance can only continue to get better as the digital banking transforms.

References

[1] Yang, L.; Elisa, N.; Eliot, N. Privacy and security aspects of E-government in smart cities. In *Smart Cities Cybersecurity and Privacy*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 89–102. [**Google Scholar**]

[2] Jadhav, B.; Patankar, A.B. A Novel Solution for Cloud Enabled E-Governance Using Openstack: Opportunities and Challenges. In Proceedings of the International Conference on Communication, Networks and Computing, Gwalior, India, 22–24 March 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 24–36. [**Google Scholar**]

[3] Capra, C.F. The Smart City and its citizens: Governance and citizen participation in Amsterdam Smart City. *Int. J. E-Plan. Res.* **2016**, *5*, 20–38. [**Google Scholar**] [**CrossRef**]

[4] Mora, L.; Bolici, R. How to become a smart city: Learning from Amsterdam. In Proceedings of the International Conference on Smart and Sustainable Planning for Cities and Regions, Bolzano, Italy, 19–20 November 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 251–266. [**Google Scholar**]

[5] Anthopoulos, L.; Sirakoulis, K. E-Government portal updates' evaluation: A comparative analysis. In *Public Affairs and Administration: Concepts, Methodologies, Tools, and Applications*; IGI Global: Hershey, PA, USA, 2015; pp. 2046–2064. [**Google Scholar**]

[6] Kolsaker, A.; Lee-Kelley, L. Citizens' attitudes towards e-government and e-governance: A UK study. *Int. J. Public Sect. Manag.* **2008**, *21*, 723–738. [**Google Scholar**] [**CrossRef**]

[7] Elisa, N. Usability, accessibility and web security assessment of e-government websites in tanzania. *Int. J. Comput. Appl.* **2017**, *164*, 42–48. [**Google Scholar**] [**CrossRef**]

[8] Katsikas, S.K.; Zorkadis, V. *E-Democracy–Privacy-Preserving, Secure, Intelligent E-Government Services: Seventh International Conference, E-Democracy 2017, Athens, Greece, December 14–15, 2017, Proceedings*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 792. [**Google Scholar**]

[9] Layne, K.; Lee, J. Developing fully functional E-government: A four stage model. *Gov. Inf. Q.* **2001**, *18*, 122–136. [**Google Scholar**] [**CrossRef**]

[10] Ojo, A.; Curry, E.; Janowski, T.; Dzhusupova, Z. Designing next generation smart city initiatives: The SCID framework. In *Transforming City Governments for Successful Smart Cities*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 43–67.

[11] Soltesz, S.; Pötzl, H.; Fiuczynski, M.E.; Bavier, A.; Peterson, L. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, Lisbon, Portugal, 21–23 March 2007; pp. 275–287. [**Google Scholar**]

[12] Merkel, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.* **2014**, *2014*, 2. [**Google Scholar**]

[13] Felter, W.; Ferreira, A.; Rajamony, R.; Rubio, J. An updated performance comparison of virtual machines and linux containers. In Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, USA, 29–31 March 2015; pp. 171–172. [**Google Scholar**]

[14] Li, Z.; Cheng, J.; Chen, Q.; Guan, E.; Bian, Z.; Tao, Y.; Zha, B.; Wang, Q.; Han, W.; Guo, M. RunD: A Lightweight Secure Container Runtime for High-density Deployment and High-concurrency Startup in Serverless Computing. In Proceedings of the 2022 USENIX Annual Technical Conference (USENIX ATC 22), Carlsbad, CA, USA, 11–13 July 2022; pp. 53–68. [**Google Scholar**]

[15] Hong, C.H.; Varghese, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–37. [**Google Scholar**] [**CrossRef**]

[16] National Vulnerability Database. CVE-2019-5736 Detail. 2019. Available online: **https://nvd.nist.gov/vuln/detail/CVE-2019-5736** (accessed on 14 April 2021).

[17] Walsh, D.J. Are Docker Containers Really Secure? 2014. Available online: **https://opensource.com/business/14/7/docker-security-selinux** (accessed on 23 March 2021).

[18] Sultan, S.; Ahmad, I.; Dimitriou, T. Container security: Issues, challenges, and the road ahead. *IEEE Access* **2019**, *7*, 52976–52996. [**Google Scholar**] [**CrossRef**]

[19] Agache, A.; Brooker, M.; Iordache, A.; Liguori, A.; Neugebauer, R.; Piwonka, P.; Popa, D.M. Firecracker: Lightweight virtualization for serverless applications. In Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), Santa Clara, CA, USA, 25–27 February 2020; pp. 419–434. [**Google Scholar**]

[20] gVisor: Application Kernel for Containers. Available online: **https://github.com/google/gvisor** (accessed on 6 May 2021).