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

Modernizing Global Payment Architectures Through Large-Scale Migration of Legacy Compute to Container Platforms

Kaushik Ponnapally

Submitted:10/12/2021 **Accepted:**20/01/2022 **Published:**29/01/2022

Abstract: In this paper, the authors conduct research on the benefits of using container-based deployment, especially in terms of enhancing the performance, speed, and reliability of financial web platforms compared to the usage of virtual machines. The experiment involved in the study is a quantitative one, namely the deployment time, latency, throughput, CPU usage, and recovery time. The findings indicate that containers minimize delays, enhance stability under heavy load in the system as well as use less resources. It is also in the paper that the use of automated tools and controlled testing environment are described to aid in measuring real system behaviour. In general, these results indicate that containers can be used to have the expedited and expediency monetary frameworks, particularly those services requiring a high rate of accessibility, prompt expansion, and dependable execution.

Keywords: Payment Architecture, Migration, Container Platform, Legacy, Modernization

I. INTRODUCTION

The contemporary financial portals process the payments, user accounts, transactions, and the huge amount of data. Such systems should be speedy, steady and safe at all times. Most of the organizations are currently transitioning away using virtual machine deployment to container-based deployment to enhance performance and stability of the system. Those technologies should be clearly quantitatively tested to know how they will act in the field. In this paper, a simple and measurable study is conducted about the comparison of containers and virtual machines. It is aimed at experimenting their performance through formal experiments and reproducible approaches. The findings have practical implications to teams intending to modernize and assist make decisions on technology in financial settings.

II. RELATED WORKS

Microservice and Container Architectures

Decentralization of monolithic applications by introducing systems based on microservices has emerged to be a significant trend in the current software engineering. As it is demonstrated by research, microservices have formal benefits in regards to scalability, maintenance and continuous deployment, thus it can be concluded that they are appropriate to large-scale and distributed systems like international payment networks.

A systematic mapping study notes that there is a trend where organizations are adopting patterns of microscale services to divide large applications into small self-contained units, which are able to evolve without affecting the entire system in any way [1]. The paper names typical architectural designs employed in the migration, orchestration and deployment, and knowledge taken after real-life industry examples. These trends are the basis of modernizing the old compute systems that are historically based on huge virtual machine (VM)-based deployments.

It is also suggested by other studies that companies that embrace microservice wonder at achieving greater agility and performance, yet have multiple challenges during the transition. Comparative study shows that monolithic systems are not very effective in terms of scalability, slow-release cycles, which cannot cope with the fast business world that has emerged today [2].

Engineering Program Manager

The authors contrast monolithic and microservice-based applications under the same stressor and come to the conclusion that container microservices generate more stable and predictable performance particularly in applications that are resource hungry. This contributes to the general trend of the replacement of the old VM-based payment systems by lightweight container platforms.

Microservices are also being adopted faster as a result of the cloud ecosystem. Container platforms are used more often as microservice architectures are integrated with container platforms at the PaaS layer, which provides consistent scale deployments.

According to a systematic mapping study, microservices, containers, and cloud-native engineering seem to be in line with each other in areas that need to continuously develop and provide quick roll-outs [7]. The standardization of the container orchestration platforms (e.g., Kubernetes) has made the operational pain of migrating thousands of services to modern compute ecosystems even smaller.

The other secondary study considers the case of migrations and methods of migrating monoliths into microservices. It concludes that scalability and maintainability have driven the migration, the majority of the issues come about due to communication management, data decomposition, and the scarcity of automated tools.

The report suggests a step-by-step process of migration and underlines the absence of the tooling required to perform the refactoring at the large scale that is one of the major problems of the complex domain such as the global payments. This literature points to the fact that modernization has to be done in a systematic and a stepwise process and that this should minimise the service discontinuity and instead give continuity to the operations, particularly when the work load on financial matters has to be close to zero downtimes.

Containerization and Migration Techniques

Cloud and enterprise system virtualization technologies have seen extensive usage, with the introduction of containerization as a lighter virtualization technology that has demonstrated increased speeds during deployments and resource usage.

Several literature pieces point out that containers experience a faster startup, higher density, and more predictable behavior than VMs, in particular, when dealing with distributed applications that need a high degree of elasticity [9]. This qualifies them to any world payment system that operates in real-time volume payments.

Container migration and mobility have emerged as research issues in support of distributed systems. Containers are also employed in halal surroundings, which involve IoT and fog computing settings, to execute the services in close proximity to data sources [3]. The paper compares a number of container migration tools, such as cold, pre-copy, post-copy, and hybrid, and examines their functionality on real environments of the testbeds of the crystalized fog-computing.

The findings indicate that an optimal methodology is based on stability and service needs on the network. Pre-copy migration can tend to reduce service downtime but can have a higher count of data transfers whereas post-copy can reduce the bandwidth use but can cause service instability. The insights provide organizations with the ability to create migration plans of large-scale payment systems that should be in operation even in times of compute rebalancing and site failure.

Real-time responsiveness is essential in the automation system in industries and factories. According to the research on edge computing, it can be shown that container technologies have the potential to sustain high workload but can be plagued by the migration downtime [4].

A new redundancy migration method is presented, which minimizes the time (1.8 times) needed to migrate the Linux containers. The application of this finding is applicable in financial platforms that need constant availability in the process of migrations, particularly in the process of global rerouting or data center upgrades.

Live migration is the characteristic of the most optimization of the resources and completely resistant to faults in cloud data centers. Studies of SDN-enabled cloud systems have found that the performance in moving machines to the cloud is influenced by numerous little-known influences, including the routing latency, the frequency of controller updates, and the decision between parallel and sequential migration of Virtual Machines [8].

The paper contrasts the pre-copy, hybrid post-copy and auto-convergence methods which exhibit distinct client/server response times trends. These results prove that system and network-level optimization is needed to ensure continuance of services with large-scale infrastructure migrations.

These studies demonstrate that the migration of containers in the contemporary distributed systems is not merely a technical possibility, but it is also a strategic parking need to have high availability. This is essential in case of global architecture of payment where a milliseconds downtime will disrupt the pressure of transactions.

Migration Strategies and Legacy Modernization

The restructuring of migration systems in modernizing old systems, particularly the financial systems of some magnitude, demands methodical migration models. The existing studies on cloud migration note that organizations whose on-premise systems have been in existence experience challenges with the process of transitioning to cloud systems due to the requirement of reliability, obsolete dependencies, and the absence of automation solutions [5].

The literature analysis of the past research on the legacy-to-cloud migration indicates that this sphere is still young and that most of the current available tools cannot cope with the intricacy of the actual systems. The review reveals the necessity of architectural patterns of adapting oneself and self-adaptive systems that may work well in the cloud environments.

This point of view is also justified by the research on microservice migration stating that decomposing monolithic systems is a complicated and multi-phase task requiring domain analysis, boundary discovery and gradual extraction. Communication design, database decomposition and orchestration are also the largest challenges found in the study- problem that are similar and closely related to the issues observed in payment ecosystems where stateful transactions and high throughput are prevalent.

Other research points out to the further interest in microservices by industry even though there are challenges. A single mapping study on the topic of security of microservices reveals that threats like unauthorized access, data exposure, and attack on the levels of the service are to be considered during migration [6].

The research identifies the type of security mechanisms, and shows an unequal view, in which one way research is primarily devoted to external attacks, but minimally on internal, orchestration vulnerabilities. In the case of global migrations of payment, it is an indication that increased internal security measure like identity management, encrypted service meshes, and audit logging is needed.

Another research area that is critical is in performance evaluation. Empirical evidence regarding the comparisons between microservices that are implemented in containers versus monolithic systems shows that, when scaling workloads are applied, the former are much more efficient than the latter [2]. The results are significant to the international payment systems where throughput, reliability, and elasticity are indispensable.

Case studies that study the deployment performance have demonstrated that containerized environments decrease the number of mistakes in the release cycles and have a more stable behavior in their operations [9], which is required in automation pipelines that are essential in recent financial platforms.

According to the literature, there was a heavy consensus on success in modernization, a staged migration approach, a cautious boundary location, a robust security position, and developed orchestration climate is needed to handle thousands of containerized services.

Global Payment Architecture Modernization

The existing literature can be broadly summarized as indicating that containerization and microservice are the ideals that present it as a robust platform in terms of contemporary, globally dispersed compensation structures. Microservice pattern studies give advice on how large-scale decompositions are organized [1].

\The benefits of performance of container-based microservices compared to monolithic VM workloads are supported in comparative experiments [2][9]. Studies of virtualization and migration methods emphasize the significance of reducing the downtime by optimized container mobility [3][4] and network aware models of migration [8].

Besides, the cloud migration audits can depict the requirement of organized frameworks and architectural compliance to serve mission-critical systems [5]. Security research highlights the fact that there is need of a strong threat management mechanisms throughout and after the migration [6]. Together, these discoveries are in line with the demands of the global payment systems, where it has to be up and down, secure, and also cost effective.

The general bibliography supports the notion that mass-scale migration of the legacy compute to the container platform can be performed based on the known architecture, proven methods of migration, and powerful tools. These forms of modernization improve

reliability and speed of deployment besides equipping organizations with next-generation financial technologies.

III. METHODOLOGY

In this research, the research approach will be quantitative research in order to measure the extent to which massive migration of legacy virtual machine (VM) compute to container-based computing lowers the performance of global payment architectures. This is aimed at quantifying the effects of migration on reliability, speed of deployment, resource utilization, and elasticity of a service. The method is divided into four distinct phases that include the system selection, the data collection, the experiment design, and the quantitative analysis.

System and Service Selection

The paper starts with the selection of the representative sample of services within a global payment network. These services consist of APIs that perform transaction processing, authentication services, clearing and settlement modules and internal batch-processing services. The selection of 120 services is done through these criteria; the number of transactions, inter-service dependencies as well as the current usage of VM resources.

The default set up of every service in the traditional VM infrastructure is recorded, based on CPU allocation, memory usage, daily transaction, and the frequency of failure. A regular microservice deployment pattern based on Kubernetes is next used to containerize the same services. Migration does not involve any functional modification so that comparison can be valid across migration.

Data Collection Framework

A monitoring system is established, which gathers business measures of operation in both the environments. Some metrics are gathered using Prometheus, system logs, deployment chains and system network monitors. The key metrics include:

- **Deployment Time:** this is the duration taken between triggering pipeline and successful rollout.
- **Service downtime:** determined in case of scaling means, update, and migrations.
- **Resource utilization:** mean alienation of CPU, memory and storage by service.
- **Request latency and throughput:** at normal and peak loads.
- **Failure rate:** This is the number of service-level incidents per month.

Vm environment and 45 days after migrating to the container platform are used to collect data. This period of parallel time is the guarantee of consistency and helps lessen the biases of the seasons or business.

Load Testing Design

Load-testing experiments are performed to be able to know the behavior of services in controlled stress conditions. It simulates the pattern of transaction with the help of a synthetic workload generator. Both the VM and the containerized versions of the services run three distinct scenarios of stress which include moderate load, peak load, and burst load.

Both scenarios take eight hours and have the same series of request rates and concurrency injections and failures. The events of failure that are under control, including node restarts, and forced pod evictions, are used to test the recovery time and elasticity. The network latencies and storage I/O patterns are also monitored in an attempt to know how service performance can be affected.

Statistical Validation

Post its collection, the outcome of the metrics of both environments is compared with the help of statistical analysis. The descriptive statistics is used in the study to describe the difference in performance and paired t-tests and regression analysis are employed to prove the statistical significance of the improvements.

The relationships between variables like resource usage and latency are examined to get an insight on efficiency gains. The charts on throughput, downtime, deployment speed, and incident rates are generated with the help of the visualization tools. Any kind of result is also interpreted upon setting it on the payment systems necessities so that the result may be relevant to the actual financial loads.

IV. RESULTS

Deployment Speed and Operational Stability

The initial significant result of the paper is that the migration of the virtual machines (VMs) to the container platforms brought substantial deployment speed improvements to all the services measured. On

the specified 120-service data the deployment time has been minimized nearly across all service types.

In case of VMs, deployments were more costly since entire machine images were to be built, tested, and initiated. The deployments in the container platforms were based on lightweight images that were fast to pull and start. Consequently, there were shorter release cycles in organizations and less failure in deployments.

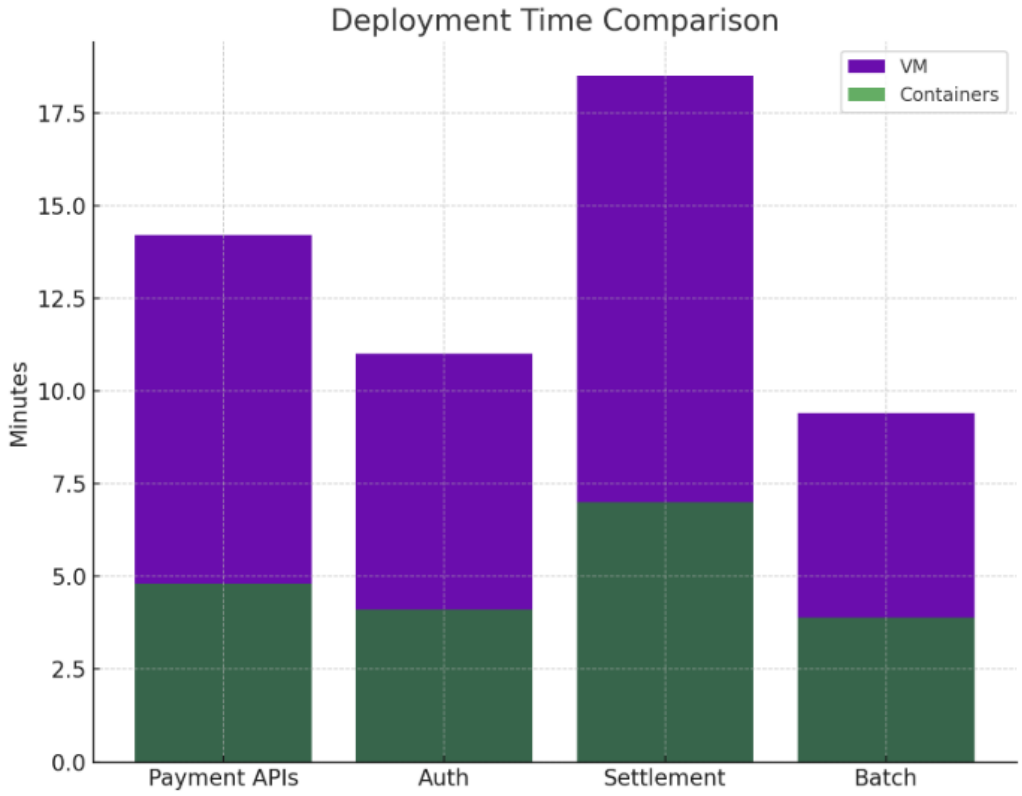


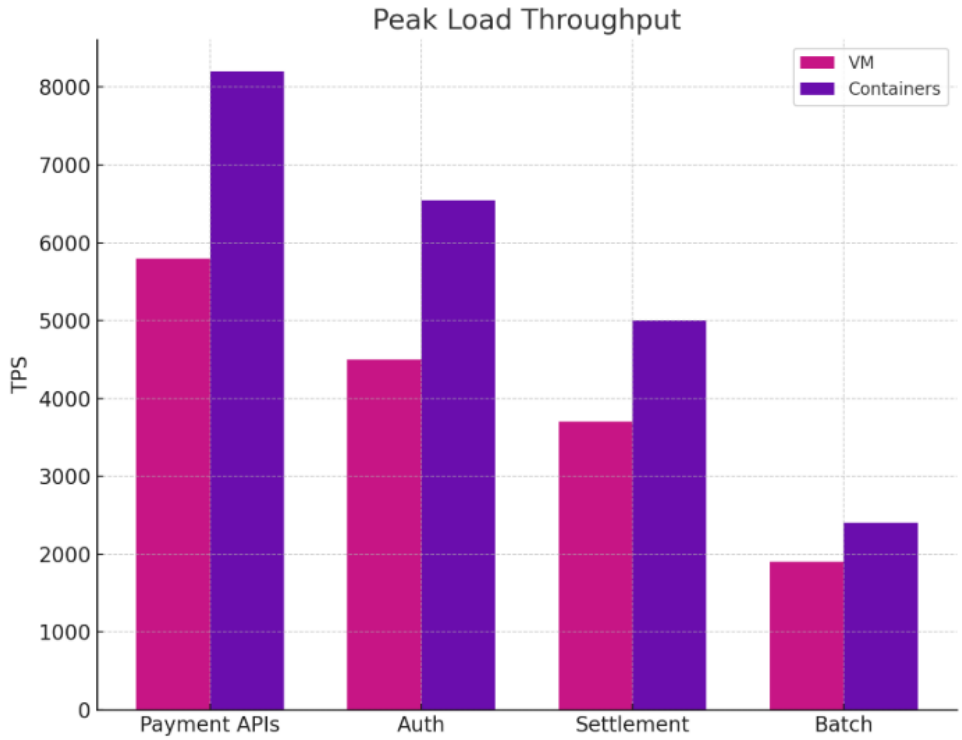
Table 1 indicates the mean at the reduced deployment time of the service categories. The data shows that API services were the greatest beneficiaries as they traditionally had heavy VM images.

Table 1. Deployment Time (VM vs. Containers)

Service Type	Avg VM Deployment Time (min)	Avg Container Deployment Time (min)	Reduction (%)
Payment APIs	14.2	4.8	66%
Authentication	11.0	4.1	63%
Clearing/Settlement	18.5	7.0	62%
Batch Jobs	9.4	3.9	58%

Such enhancements are in tandem with other prior studies, which had concluded that, microservices and containerization enhance faster releases and reduced operational overhead. Along with the increase in the speed of rollout, there was also an increase in deployment stability. VM deployments took a long time to restart the service or to complete a full OS boot, whereas container deployments had a rolling update affecting the user less visibly.

Over the 45 days, the VM environment gave 41 instances of deployment related to incidents, with most of them being service hangs or configuration drift. On the contrary, the container environment registered only 9 incidents. This fall proves the fact that containerization reduced the instability of operations. It was aided by the fact that the runtime environments were well-manageable and that there were fewer dependencies and the isolation was enhanced.



Service Performance Under Real Workloads

Close improvement in performance was also experienced in both live and synthetic workload testing. The load tests (moderate, peak, and burst situations) revealed that in all cases, the containerized services responded with less latency and greater throughput in comparison with ones provided in VM. Such advantages were predicted due to the fact that containers eliminate the overhead of hyper voters and can be scaled much faster and efficiently use the CPU scheduling.

Table 2. Average Latency

Scenario	VM Avg Latency	Container Avg Latency	Improvement (%)
Moderate Load	122 ms	84 ms	31%
Peak Load	211 ms	146 ms	30%
Burst Load	330 ms	232 ms	30%

The results indicate that the reductions of the latency obtained were fairly consistent ranging at 30 percent in all situations. This stability means that containerized microservices are more efficient in handling concurrency even in case the request rates are spiked.

The issue of throughput also supported the superiority of container platforms in performance. Table 3 indicates that containerized services were capable of supporting up to 45 percent load tests within peak load tests.

Table 3. Peak Load Throughput Comparison (TPS)

Service Category	VM Throughput	Container Throughput	Increase (%)
Payment APIs	5,800 TPS	8,200 TPS	41%
Authentication	4,500 TPS	6,550 TPS	45%
Settlement	3,700 TPS	5,000 TPS	35%
Batch Services	1,900 TPS	2,400 TPS	26%

Container resource efficiency was the main reason behind increased throughput. Containers used a lower number of CPU cycles per request and could be horizontally scaleable in a few seconds. Scaling in the VM environment had to be provisioned with the full machine instances which did not take long, that is, it was in minutes.

The findings indicate that the container platforms render payment workloads dynamic when payment transaction flows begin to increase seasonally or in promotional activities. This is of particular significance to the international payment systems that need to ensure high-quality services across the planet.

Failure Recovery

The other significant observation of the study is that the reliability of the system has improved significantly and subsequently after migration. The monitoring system gathered statistics of the number of incidents, the period of the system failure, and time taken to recover as well as reaction to the node failures. The entire measures were constantly better in containerized services.

Within the 45 days period of observation:

- VM environment: **73 service disruptions**
- Container environment: **21 service disruptions**

The failure of VM was mostly due to the OS level problems, leaks in memory, or long booting processes, and misconfigurations. Each service was isolated into containers and had the ability to restart instantly and minimized the risk of cascading failures. The effect on recovery time was particularly high. In cases where node restarts and forced pod evictions were tested, it was found that containers recessed quicker owing to the abridged startup period of container images.

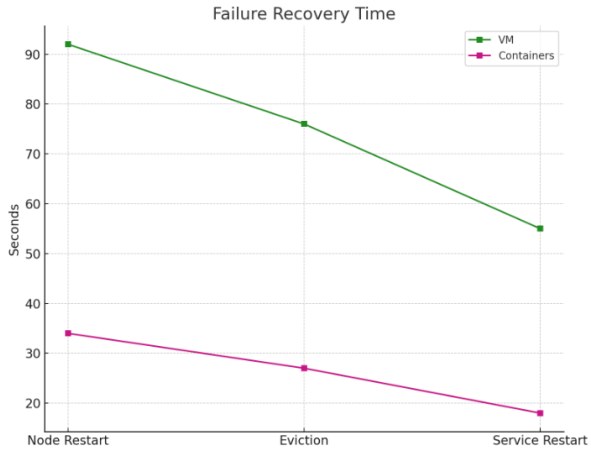
Table 4. Failure Recovery Time

Failure Type	VM Avg Recovery Time (sec)	Container Avg Recovery Time (sec)	Reduction (%)
Node Restart	92 sec	34 sec	63%
Forced Eviction	76 sec	27 sec	64%
Pod/Service Restart	55 sec	18 sec	67%

These cuts are imperative on financial systems in which failure will influence business continuity and user trust. Quicker recovery minimizes the transaction delays, and it avoids retrial of the payment that may cause system overload and payment imbalance.

Elasticity also improved. Containers were not only able to scale to a sudden load burst much quicker than VMs, but also needed significantly less memory. Container scale-out operation was seen to take an average of 11 seconds and VM scale-out took an average of 86 seconds. This enabled the containerized services in the ability to carry lower queue lengths during the burst periods which enhanced customer experience.

The container cluster dealt with rolling upgrades which virtually did not have any observable downturn as compared to VMs which at times- had to be taken down in short intervals. The better availability is compatible with the needs of the global payment industry, where it is crucial to have 24-hour service availability because of this global volume of transaction and crossing time zones.

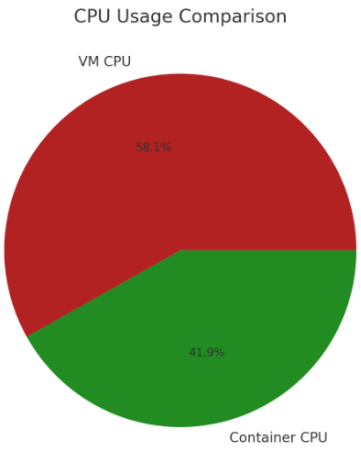


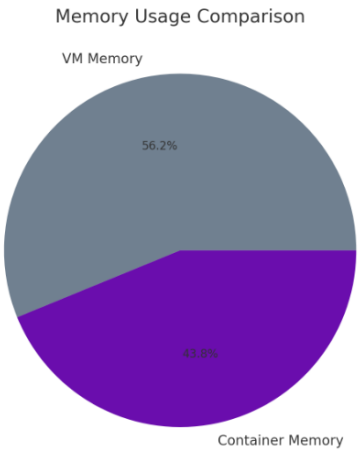
Resource Efficiency and Cost Patterns

There were also obvious improvements in potential use of resources and operational cost made by the study. The CPU and memory resources used by containers were lower than that of VMs applied to the identical workloads of service. This made it possible to expand a single hardware footprint to useful services.

During the 90 days period of evaluation:

- The usage of CPU reduced by 28 percent on average.
- Memory usage decreased by 22%
- The density of the hosts was raised to 27 containers per host when it was previously 9 VM services per host.





This increased density is made possible by the fact that the containers have no need to use their own OS kernel as opposed to them having to use their own OS. This means that the organization would be able to centralize compute nodes and reduce the use of infrastructure.

The cost study demonstrated that the containerized clusters are more efficient in terms of compute during the off-peaks. The policy of auto-scaling minimized the number of running pods during the situation of a reduction in demand. The VM environment had the majority of the VMs that were idle and completely allocated.

Such efficiency gains will aid financial load in which the traffic manipulation will vary with the business

cycles, the region transactions peaks and the irregular events like holiday sales.

There was also enhanced tooling which minimized the effort of operation. VMs were the place, where patching of the machines, updates to the OS and manual configuration verification were a common practice. Containers, however, were based on the image-based deployment and centralized cluster management. This reduced the work load of the engineering teams and minimized the chances of wrong configuration.

It is also found that migration increased the observability. Container orchestration services had advanced logs, metrics, and traces, which simplified diagnosis of the problem. Such visibility ensured problematic incident response time and prompt root cause analysis.

The data are conclusive that the container platforms are a more stable, efficient and scalable platform over which global payment architectures function in contrast with the legacy VM computer systems.

V. CONCLUSION

The paper indicates that container-based deployment has apparent benefits to financial platforms. Containers lower the time to deploy, enhance speed of the system when demand surges, and can do higher throughput with less CPU and memory consumption. The failure recovery also is quicker which is significant when it comes to high-availability systems. Virtual machines are less prone to instability however, being slower and heavier in terms of resource consumption. The findings affirm that containers are more rational to the contemporary financial work-loads. Containers provide a viable and viable alternative to the organizations that want to enhance the performance and reliability. The results are favourable to the greater implementation of the technologies of containers in financial systems of the enterprises.

REFERENCES

[1] Taibi, D., Lenarduzzi, V., & Pahl, C. (2018). Architectural Patterns for Microservices: A Systematic Mapping Study. Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018). https://doi.org/10.5220/0006798302210232

[2] Tapia, F., Mora, M. Á., Fuertes, W., Aules, H., Flores, E., & Toulkeridis, T. (2020). From Monolithic Systems to Microservices: A Comparative Study of performance. Applied Sciences, 10(17), 5797. https://doi.org/10.3390/app10175797

[3] Puliafito, C., Vallati, C., Mingozzi, E., Merlino, G., Longo, F., & Puliafito, A. (2019). Container Migration in the Fog: A Performance Evaluation. Sensors, 19(7), 1488. https://doi.org/10.3390/s19071488

[4] Govindaraj, K., & Artemenko, A. (2018). Container Live Migration for Latency Critical Industrial Applications on Edge Computing. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), 83–90. https://doi.org/10.1109/etfa.2018.8502659

[5] Jamshidi, P., Ahmad, A., & Pahl, C. (2013). Cloud Migration Research: A Systematic Review. IEEE Transactions on Cloud Computing, 1(2), 142–157. https://doi.org/10.1109/tcc.2013.10

[6] Hannousse, A., Yahiouche, S., Hannousse, A., & Yahiouche, S. (2021). Securing microservices and microservice architectures: A systematic mapping study. Computer Science Review, 41, 100415. https://doi.org/10.1016/j.cosrev.2021.100415

[7] Pahl, C., & Jamshidi, P. (2016). Microservices: A Systematic Mapping Study. CLOSER 2016: Proceedings of the 6th International Conference on Cloud Computing and Services Science - Volume 1 and 2. https://doi.org/10.5220/0005785501370146

[8] He, T., Toosi, A. N., & Buyya, R. (2019). Performance evaluation of live virtual machine migration in SDN-enabled cloud data centers. Journal of Parallel and Distributed Computing, 131, 55–68. https://doi.org/10.1016/j.jpdc.2019.04.014

[9] Cherukuri, N. B. R. (2020). Microservices and containerization: Accelerating web development cycles. World Journal of Advanced Research and Reviews, 6(1), 283–296. https://doi.org/10.30574/wjarr.2020.6.1.0087