Performance Evaluation of Mojaloop in Cloud Environments

Test Rounds, Metrics, and Observations

Authors: Mojaloop Foundation and AWS

March 2026

mojaloop foundation

Contents

Executive Summary

The Mojaloop platform has evolved significantly since its initial release, moving beyond its original focus on domestic instant payments to support real-time, cross-border use cases. As Mojaloop becomes a core component of national and regional payment infrastructure, it is increasingly important to validate its performance, scalability, resilience, and cost efficiency. These attributes are foundational to trust, adoption, and sustained usage, particularly in inclusive finance contexts where reliability and affordability are critical.

To address these needs, the Mojaloop Foundation, in partnership with Amazon Web Services (AWS), conducted a comprehensive performance and resilience testing program focused on Mojaloop version 17. This white paper presents the methodology, results, and recommendations from that effort. It is intended to support scheme operators, system integrators, and technical decision-makers by providing evidence-based insight into how Mojaloop performs .

The testing program evaluated Mojaloop's ability to process high transaction volumes while maintaining low latency, high availability, and low error rates. Multiple test rounds were conducted in cloud-based environments, simulating eight Digital Financial Services Providers (DFSPs) with realistic load distributions. Scenarios included sustained transaction volumes of up to 1,000 transactions per second, infrastructure scaling between test rounds, and configurations with and without replication to assess resilience trade-offs. Performance was measured across the full three-phase Mojaloop payment lifecycle: discovery, pre-validation and agreement of terms, and payment execution.

Across all benchmark scenarios, Mojaloop demonstrated stable and predictable performance. **The platform consistently achieved target throughput levels with success rates exceeding 99.97 percent, maintained end-to-end transaction latency within real-time payment expectations, and achieved 100 percent system availability during test runs.** Resource utilization remained modest, indicating significant headroom for growth and confirming that high throughput can be achieved without disproportionately expensive infrastructure for domestic deployment. These results show that Mojaloop can support national-scale instant payment systems while remaining cost-efficient and operationally manageable.

To complement laboratory benchmarking, this white paper also includes real-world performance validation from Mojaloop implementers. **Case studies from Ablipay in the Philippines and the Pay Na-Na Inclusive Instant Payment System in Liberia demonstrate how Mojaloop behaves in production and near-production environments.** These implementations confirm that the performance characteristics observed in controlled testing translate into reliable, real-world operation. They also highlight the importance of deployment architecture, autoscaling behavior, and observability in achieving optimal outcomes.

It is important to note that performance results obtained in laboratory or cloud-based benchmarking environments do not represent guaranteed production outcomes. In real-world scheme environments, a range of external factors can materially influence achievable performance. These include network latency between participants, the performance and configuration of connected Digital Financial Services Providers

(DFSPs), participant-side timeout configurations, infrastructure choices, hardware variability, database tuning, operational maturity, traffic patterns, fraud and compliance integrations, and cross-border routing dependencies. As a result, actual throughput, latency, and availability characteristics in live schemes may differ from benchmark results. The figures presented in this paper should therefore be interpreted as validated reference benchmarks under defined conditions, rather than prescriptive guarantees.

Based on these findings, the Mojaloop Foundation recommends that adopters:

- Follow established international principles for operational resilience
- Conduct performance testing early in the deployment lifecycle
- Prioritize observability and scalable architecture:

Looking ahead, the Foundation plans to expand this work through additional testing, including higher throughput scenarios, fault-injection and resilience testing, on-premises deployments, and evaluation of architectural enhancements such as the introduction of TigerBeetle.

Together, the results presented in this white paper demonstrate that Mojaloop can support reliable, scalable, and affordable instant payment systems for domestic and cross-border use cases, reinforcing its role as a digital public good for inclusive finance.

Introduction and Rationale

The Mojaloop platform has undergone substantial evolution since its initial release, marking a new phase in its journey as a mission-critical infrastructure for inclusive instant payments. Initially designed to support domestic transfers, Mojaloop now enables real-time cross-border payments, a transformation that demands a reassessment of the platform's performance, scalability, and reliability.

These changes come at a time when infrastructure landscapes have also matured. Hardware has become more powerful and affordable, and while data localization requirements may limit widespread adoption, cloud-native deployment models are increasingly being considered as part of national and regional payment strategies.

However, despite these improvements, non-functional requirements—such as infrastructure provisioning, fault tolerance, and system performance are often overlooked during system planning. These elements are not just technical concerns: they are foundational to trust, adoption, and sustained usage. For instance, when a payment fails due to infrastructure constraints, whether from under-provisioned servers, timeouts, or high error rates, the user experience is disrupted. In inclusive finance, where users are often first-time adopters of digital channels, such failures can lead to a loss of confidence and a permanent shift away from the platform.

Moreover, infrastructure costs remain a key barrier for many markets considering the transition to instant payments. The ability to operate efficiently on low-cost infrastructure whether on-premises or in the cloud is vital for enabling financial inclusion at scale.

For these reasons, the Mojaloop Foundation, in partnership with AWS, initiated an intensive performance testing campaign focused on version 17 of the Mojaloop platform. This whitepaper presents the methodology, findings, and recommendations from that testing initiative. It is intended to support decision-makers, implementers, and operators by demonstrating that Mojaloop can achieve high throughput, maintain resilience under stress, and operate affordably in resource-constrained environments.

Environments set up and testing protocol

To evaluate the performance and resilience of the Mojaloop platform at scale, a series of structured tests were conducted under controlled but realistic conditions. The objective was to simulate real-world usage scenarios, including increasing transaction volumes, fault injection, and varying deployment environments on a cloud-based environment (AWS).

This section outlines the key functional components of the Mojaloop system that influence performance, describes the architectural and infrastructure setup used in testing, and provides an overview of the methodology applied to measure system behavior under load and in degraded states.

Key functional features impacting the performance of the systems

Mojaloop is an open-source instant payments platform or payments switch designed to enable interoperability and facilitate real-time payments between Digital Financial Services Providers (DFSPs), or entity such as a bank, mobile money operator, fintech, or microfinance institution that offers digital financial services to end users. DSFPs connect to the switch to send and receive payments on behalf of their customers. The hub is primarily developed in JavaScript and built to handle high transaction volumes of low values.

At the core of Mojaloop's operation is a three-phase payment lifecycle, each of which introduces specific performance considerations:

1. **Discovery Phase**
   In this phase, the system resolves a user-friendly alias, such as a mobile phone number. Mojaloop queries an alias directory service and enriches the alias with the associated beneficiary's details, including the DFSP responsible for servicing the account.
   Performance impact: This phase involves lookup calls and requires low-latency response times to preserve user experience.
2. **Pre-validation / Quotation /Agreement of Terms Phase**
   Once the beneficiary's DFSP is identified, a payment request is sent to them. The beneficiary DFSP validates the request and confirms that it agrees to process the corresponding payment upon receipt. They commit to executing the transaction by adding a cryptographic lock to the response, along with a timeout that defines the validity period of their commitment and, where applicable, includes the fees to process the transaction.
   Performance impact: This step requires reliable inter-DFSP messaging and cryptographic processing and must operate within strict time constraints to maintain end-to-end responsiveness.
3. **Payment Execution Phase**
   The final step is the actual transfer of funds. The initiating DFSP sends a payment message, corresponding to the previous pre-validation request, passing along the cryptographic lock. The beneficiary DFSP validates this message, sends back a confirmation of credit and disburses the funds to the end user.
   Performance impact: Throughput and latency during this phase are critical to achieving instant payment expectations at scale.

Settlement of transactions is handled outside the Mojaloop platform, typically via an external system such as a central bank's RTGS. However, Mojaloop prepares and communicates inter-DFSP obligations, which can be settled in either deferred or real-time modes. The performance of this settlement process was not considered during testing.

Security Architecture

Mojaloop has been architected to operate securely over public internet infrastructure, with a multi-layered security model designed to protect against unauthorized access and data tampering. The platform integrates a combination of transport-level, message-level, and protocol-level security features to ensure the confidentiality, integrity, and authenticity of all transactions and system communications. While essential for ensuring trust and system integrity, security mechanisms introduce additional computational and network overhead. In high-throughput systems like Mojaloop, these impacts must be carefully managed to maintain real-time performance.

Key components include:

1. **Mutual Transport Layer Security (mTLS)**
   Ensures that both the client and server authenticate each other using digital certificates. It provides encrypted communication channels and strong identity verification, mitigating the risk of man-in-the-middle attacks.
   Performance impact: This adds central processing unit (CPU) overhead and latency during connection setup due to certificate-based mutual authentication and encryption. Care must be taken to prefer long-lived mTLS connections between participants and the switch to minimize latency introduced by connection setup.

2. **JSON Web Signature (JWS)**
   Applies digital signatures to structured data (typically in JSON format), ensuring that messages are both tamper-proof and verifiably authentic. This is critical for securing transaction payloads across distributed systems.
   Performance impact: It increases processing time as each message must be cryptographically signed and verified.

3. **OAuth**
   A widely adopted open-standard authorization framework used to control access to resources. In Mojaloop, OAuth enables secure delegation of access rights without exposing sensitive credentials, supporting fine-grained access control between services.
   Performance impact: It introduces cryptographic token validation overhead, especially in high-frequency inter-service communication.

4. **Interledger Protocol (ILP)**
   A protocol designed to enable secure, conditional transfers across different payment networks. In Mojaloop, ILP ensures atomic transaction execution by using cryptographic condition fulfilment, whereby a transaction is completed only if a pre-agreed cryptographic lock is validly signed and fulfilled by the recipient DFSP.
   Performance impact: It requires additional cryptographic computations and message coordination to handle conditional transfers within strict time constraints.

Infrastructure

To assess the platform's performance across various deployment models, Mojaloop was tested in a cloud-based setup on AWS. Each test round used defined infrastructure specifications.

Test Round	Infrastructure Specifications

Round 1: Target 500 TPS	• Switch:
	  ○ Switch Services:
	  x3 m7i.4xlarge (x16 vCPU, 64GB RAM)
	  ○ Switch Kafka:
	  x4 m7i.4xlarge (x16 vCPU, 64GB RAM)
	  ○ Switch MySQL:
	  x1 m7i.4xlarge (x16 vCPU, 64GB RAM)
	• Load Generation
	• FSP Simulators:
	  x8 m7i.2xlarge - 1 per simulated DFSP (x8 vCPU, 32GB RAM)
	• K6 Test Orchestrator:
	  x1 m7i.4xlarge (x16 vCPU, 64GB RAM)

Round 2: Target 1000 TPS	• Switch:
	  ○ Switch Services:
	  x3 m7i.8xlarge (x32 vCPU, 128GB RAM)
	  ○ Switch Kafka:
	  x2 m7i.4xlarge (x16 vCPU, 64GB RAM)
	  ○ Switch MySQL:
	  x1 m7i.4xlarge (x16 vCPU, 64GB RAM)
	• Load Generation
	• FSP Simulators:
	  x8 c7i.4xlarge - 1 per simulated DFSP (x16 vCPU, 32GB RAM)
	• K6 Test Orchestrator:
	  x1 m7i.xlarge (x4 vCPU, 16GB RAM)

Round 3: Target 1000 TPS (with additional data replication enabled)	• Switch:
	  ○ Switch Services:
	  x3 m7i.8xlarge (x32 vCPU, 128GB RAM)
	  ○ Switch Kafka:
	  x2 m7i.4xlarge (x16 vCPU, 64GB RAM)
	  ○ Switch MySQL:
	  x2 m7i.4xlarge (x16 vCPU, 64GB RAM)
	• Load Generation
	• FSP Simulators:
	  x8 c7i.4xlarge - 1 per simulated DFSP (x16 vCPU, 32GB RAM)
	• K6 Test Orchestrator:
	  x1 m7i.xlarge (x4 vCPU, 16GB RAM)

Components Deployed

Each test environment was deployed with a Mojaloop stack and supporting components required to simulate a production-grade implementation. These included:

- **Mojaloop platform version 17:** Services and components required to support full Mojaloop 3 phase transfer flows (discovery, pre-validation/agreement of terms and clearing).
- **Testing Toolkit (TTK):** Mojaloop's testing framework was used to validate system and to verify that the environment was correctly deployed and functioning as expected before performance testing commenced.
- **Keycloak:** an open-source identity and access management tool providing authentication, authorization, and single sign-on.
- **Vault:** a tool for securely managing secrets, credentials, and encryption keys.
- **Cert-manager with Let's Encrypt:** manages TLS certificates for secure communication using Let's Encrypt as a certificate authority.
- **K6:** a modern load testing tool for measuring system performance and scalability under stress.

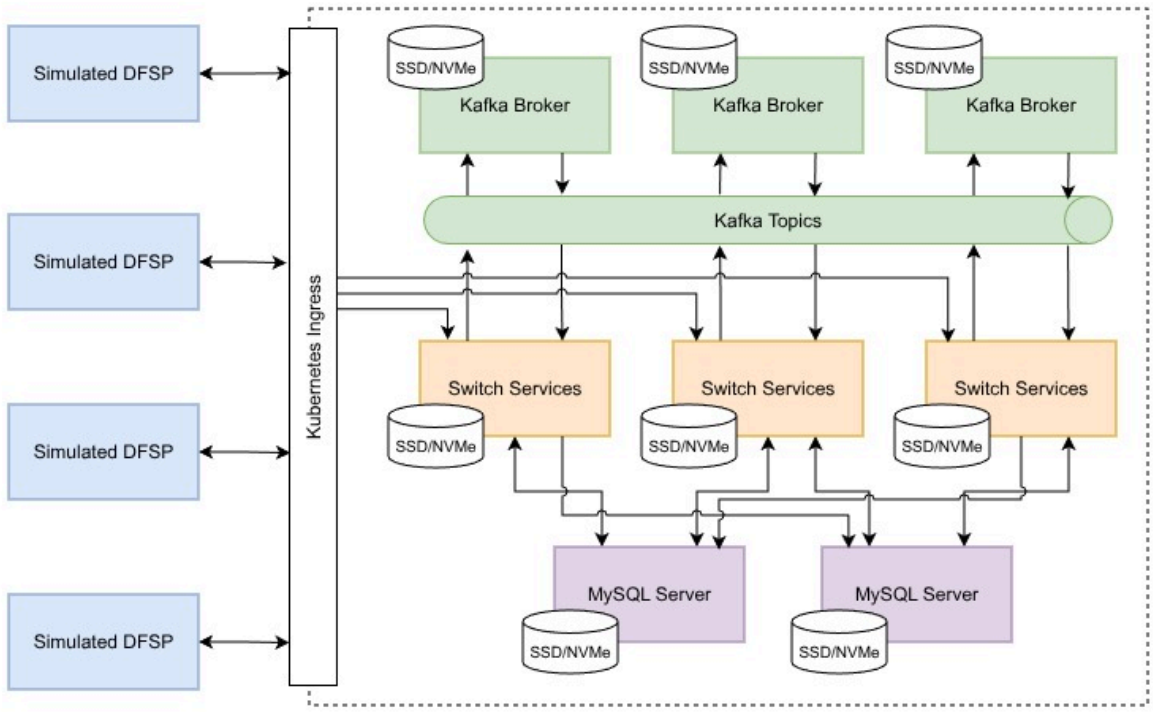


Figure 1: General Performance Testing Deployment Architecture

Test Scenarios and Methodology

The test environment was configured with eight simulated DFSPs, each generating a percentage of the overall load.

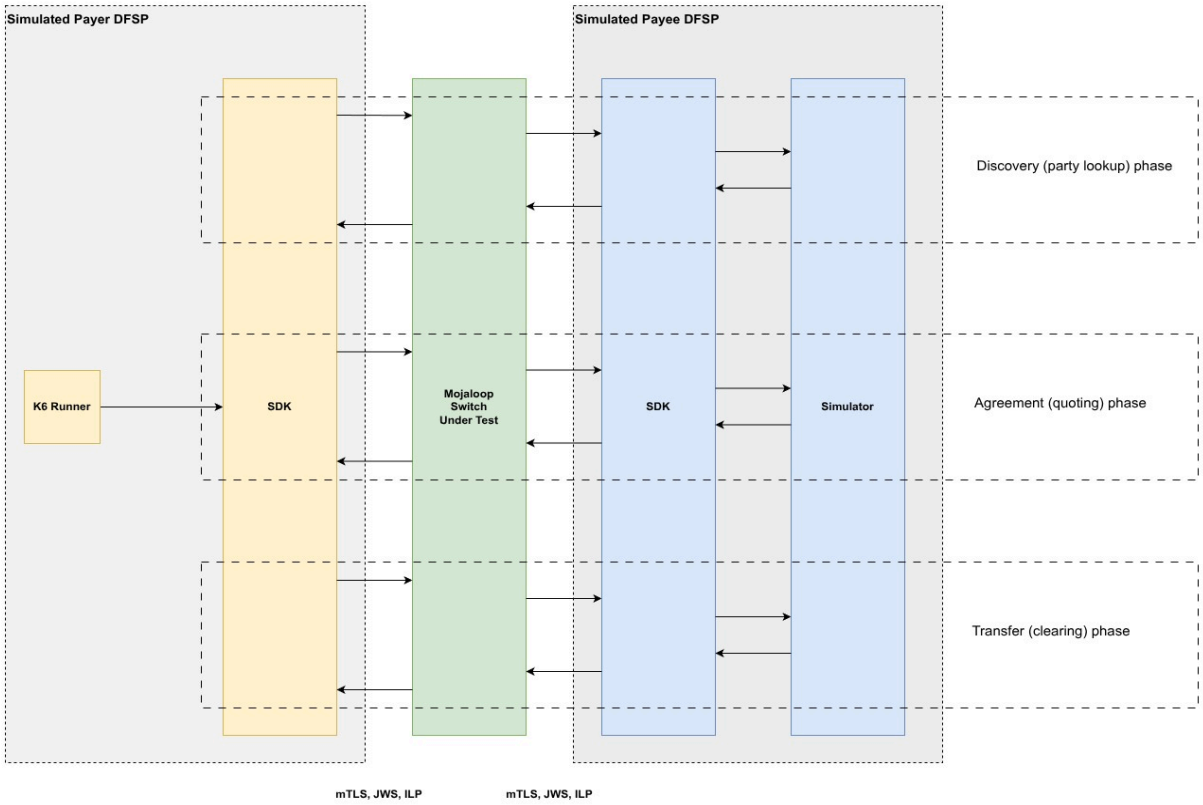


Figure 2: Simulated DFSP arrangement for each DFSP

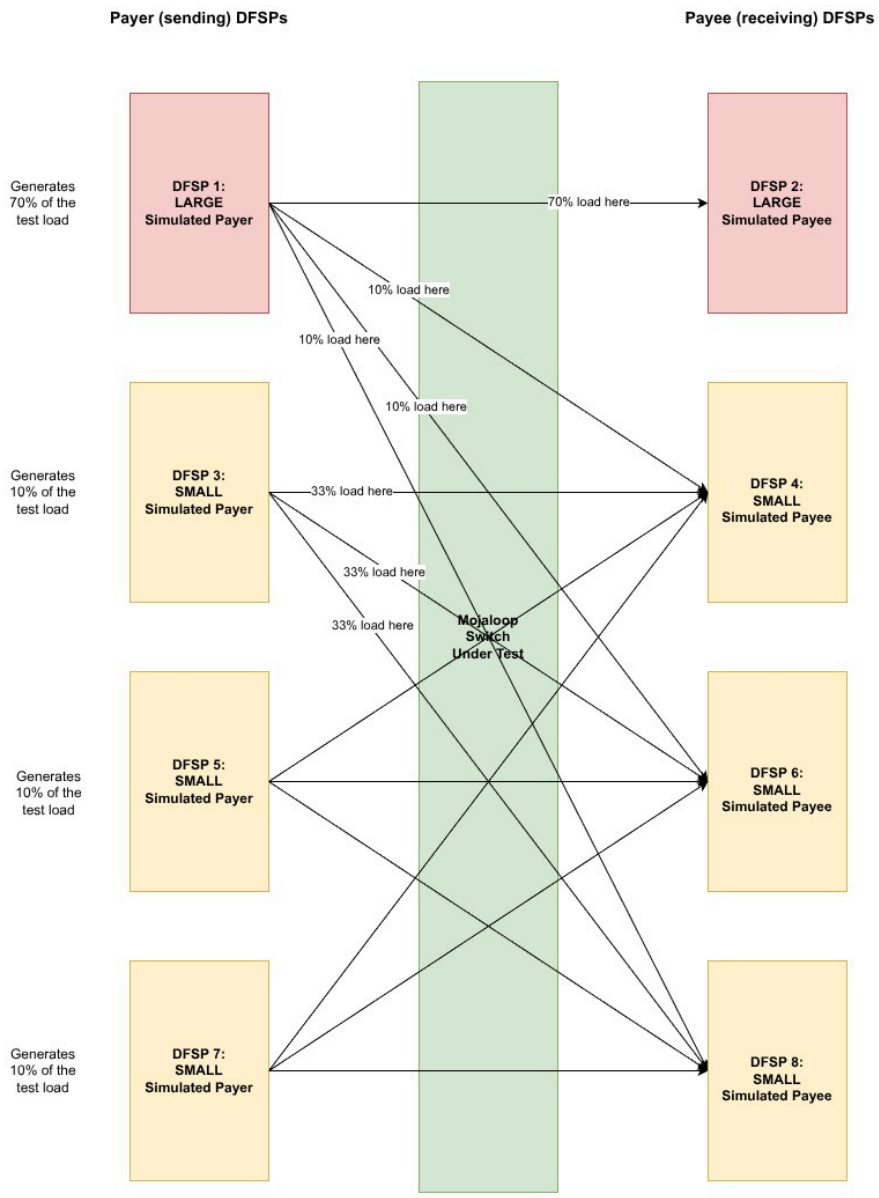


Figure 3: Illustration of load distribution between simulated DFSPs

Relative load percentage for each DFSP was adjusted to simulate a typical scheme with a mix of large, highly active institutions, e.g. mobile money providers, and smaller, less active institutions like MFIs.

The system was set up to perform deferred multilateral net settlement, with a single settlement window configured before each test run. No performance testing of the settlement process itself was performed.

All simulated transactions were peer-to-peer using MSISDN account identifiers. The primary difference between Mojaloop use cases lies in how transactions are initiated and therefore is not relevant when measuring the performance of the switch itself.

Transaction loads were gradually increased to measure the infrastructure's scaling capacity:

- 3,000 transfers at 100 transactions per second (TPS) ending with
- 1,000,000 transfers at 100 transactions per second (TPS)

- 3,000 transfers at 200 transactions per second (TPS) ending with
- 1,000,000 transfers at 200 transactions per second (TPS)

- 3,000 transfers at 500 transactions per second (TPS) ending with
- 1,000,000 transfers at 500 transactions per second (TPS)

- 3,000 transfers at 1,000 transactions per second (TPS)
- 5,000 transfers at 1,000 transactions per second
- 10,000 transfers at 1,000 transactions per second
- 100,000 transfers at 1,000 transactions per second
- 1,000,000 transfers at 1,000 transactions per second

The infrastructure was scaled between test rounds not only to support higher transaction volumes, but also to understand the impact on infrastructure costs when achieving higher throughput levels.

Testing tools

A combination of industry-standard tools was used to measure system performance and to monitor infrastructure behaviour throughout the testing process:

- **K6** was used to inject payment transaction loads at varying rates, simulating both steady-state and peak traffic conditions.
- **Loki and Grafana** were used to collect and visualize real-time logs and system metrics, supporting the identification of bottlenecks and anomalies.
- **Database performance** was monitored using MySQL Query Profiler, providing insight into query execution times and database efficiency under load.
- **System metrics** were collected through AWS CloudWatch and Node Exporter.

This methodology enabled a comprehensive evaluation of the system's throughput, latency and stability under stress.

KPIs and metrics

For each test round, the following indicators were tracked:

- Transaction throughput (overall transaction API flow throughput, from external endpoints perspective)
- Latency/response time to process payments; each phase was measured individually and combined as an end-to-end measurement.
- System availability
- Error rate/system failure
- Resource utilization (percentage of CPU, memory, disk, and network bandwidth)

Results Summary

(The corresponding infrastructure is specified in the table: "*Table*: *Infrastructure details*" in the following section)

Testing Round	Scenario	KPI	Measure
Round 1	8 DFSPs with load distribution as per figure 3. Injecting 1,000,000 3-phase transfers at 500 per second. EBS data replication.	End-to-end transaction throughput; including discovery, agreement and clearing phases.	500 TPS for a total of 1 million transfers achieved with 99.999% success rate
		Latency/response time to process end-to-end payments	e2e_time: avg=690.72ms, min=276ms, med=679ms, max=1.97s, p(90)=867ms, p(95)=929ms
		System availability	100%
		Error rate/system failure	0.001% failed transfers
		Resource utilization (percentage of CPU, memory, disk, and network bandwidth)	Average 10-20% over duration of test.
Round 2	8 DFSPs with load distribution as per figure 3. Injecting 1,000,000 3-phase transfers at 1000 per second. EBS data replication.	End-to-end transaction throughput; including discovery, agreement and clearing phases.	1000 TPS for a total of 1 million transfers achieved with 99.9899% success rate
		Latency/response time to process end-to-end payments	e2e_time: avg=560.11ms, min=173ms, med=523ms, max=4.32s, p(90)=734ms, p(95)=829ms
		System availability	100%

Testing Round	Scenario	KPI	Measure
		Error rate/system failure	0.0101% failed transfers
		Resource utilization (percentage of CPU, memory, disk, and network bandwidth)	Average 20% over duration of test.
Round 3	8 DFSPs with load distribution as per figure 3. Injecting 1,000,000 3-phase transfers at 1000 per second. EBS data replication. MySQL replication (x2 nodes)	End-to-end transaction throughput; including discovery, agreement and clearing phases.	1000 TPS for a total of 1 million transfers achieved with 99.9792% success rate
		Latency/Response time to process end-to-end payments	e2e_time: avg=674.68ms, min=240ms, med=642ms, max=5.55s, p(90)=853ms, p(95)=952ms
		System availability	100%
		Error rate/system failure	0.0208% failed transfers
		Resource utilization (percentage of CPU, memory, disk, and network bandwidth)	Average 10-20% over duration of test.

Please find the comprehensive results in the Mojaloop github organisation here:
https://github.com/mojaloop/ml-perf-whitepaper-ws/tree/main/performance-tests/results

Recommendations

- **Framework**
  The Mojaloop Foundation recommends following CPSS-IOSCO and BCBS Principles for Operational Resilience to ensure systems are adequately resilient to adverse events.
- **Infrastructure recommended**
  The testing performed allowed to establish the following table:

Table 1: Infrastructure details

TPS (Env)	AWS
500 TPS	*Infra for initial run without replication / redundancy* • Mojaloop Switch: m7i.4xlarge (3 nodes) • Kafka Nodes: m7i.4xlarge (1 node) • MySQL Node — m7i.4xlarge (1 node) • FSP Nodes — m7i.2xlarge (8 nodes for 8 FSPs) • k6 node — m7i.xlarge (1 node for injecting load) • Results: 500 (F)TPS for a total of 1 Million transfers achieved with 99.999% success rate with below configuration and details. • Note: This test is a good candidate for further tuning if 500 TPS is the desired outcome. *Detailed results: https://github.com/mojaloop/ml-perf-whitepaper-ws/tree/main/performance-tests/results/500tps*https://github.com/mojaloop/ml-perf-whitepaper-ws/tree/main/phases/11-performance-tests/results/02-scale-500tps
1,000 TPS	*Infra for initial run without replication / redundancy* • Mojaloop Switch: m7i.4xlarge (3 nodes) • Kafka Nodes: m7i.4xlarge (1 node) • MySQL Node — m7i.4xlarge (1 node) • FSP Nodes — m7i.2xlarge (8 nodes for 8 FSPs) • k6 node — m7i.xlarge (1 node for injecting load) *Detailed results: https://github.com/mojaloop/ml-perf-whitepaper-ws/tree/main/performance-tests/results/1000tps*https://github.com/mojaloop/ml-perf-whitepaper-ws/tree/main/phases/11-performance-tests/results/03-target-1000tps *Infra for second run WITH Kafka, mysql replication* • Mojaloop Switch: m7i.8xlarge (3 nodes) • Kafka Nodes: m7i.4xlarge (1 node) • MySQL Node — m7i.4xlarge (1 node) • FSP Nodes — c7i.4xlarge (8 nodes for 8 FSPs) • k6 node — m7i.xlarge (1 node for injecting load) • Note: This setup can support higher loads of about 1200 TPS which was tested and likely can be extended further. *Detailed results: https://github.com/mojaloop/ml-perf-whitepaper-ws/tree/main/performance-tests/results/1000tps-replication*https://github.com/mojaloop/ml-perf-whitepaper-ws/tree/main/phases/11-performance-tests/results/03-replication-1000tps
2,500 TPS	Ongoing – as of 9th Dec 2025

Planned Further Work

As the Mojaloop platform continues to evolve, the Mojaloop Foundation plans to publish an updated edition of this performance whitepaper toward the end of the year. This update will incorporate the results of additional performance and resilience testing aligned with ongoing developments in the Mojaloop product roadmap.

A key evolution currently underway is the introduction of TigerBeetle as the core transaction database. This architectural change is expected to significantly enhance system throughput, consistency, and

fault tolerance. To assess and quantify the impact of this enhancement, a new round of performance testing will be conducted under production-like conditions.

In parallel, the scope of resilience testing will be expanded to further evaluate the platform's behaviour under adverse and degraded operating conditions. These tests will include controlled fault-injection scenarios such as random service failures, simulated power outages, network failures, periods of high network latency, and intermittent or random network disruptions. Particular attention will be given to validating recovery characteristics, including Recovery Time Objective (RTO) and Recovery Point Objective (RPO), during simulated hardware failure scenarios.

While the current testing environment is hosted within AWS laboratory infrastructure, future testing will also be executed in on-premises environments. This will allow the Foundation to better reflect real-world deployment models commonly used by hub operators and to validate Mojaloop's performance and resilience characteristics outside cloud-native environments, including within MLF on-premises laboratory setups.

Through this extended testing programme, the Mojaloop Foundation aims to further demonstrate the platform's ability to operate reliably at scale, maintain transactional integrity under stress, and support a wide range of deployment and operational models across jurisdictions.

Cases Studies from Mojaloop implementers

Background Information

To complement the testing methodology and validate real-world applicability, this whitepaper features insights from two Mojaloop adopters:

- Ablipay, operating a domestic deployment of the Mojaloop hub in the Philippines, focused on enabling inclusive instant payments at the national level.
- The Central Bank of Liberia, running a domestic instant payment system

These use cases demonstrate how the system performs under real-world conditions, particularly specific country or region configuration and offer critical validation of the test scenarios presented. They also provide implementation-specific perspectives on throughput, onboarding, operational readiness, and infrastructure scalability.

Abli Performance Results

Purpose and scope

This report presents the performance results of Abli's Mojaloop-based deployment under production-style operating conditions. It complements the Mojaloop performance benchmarks presented elsewhere in this whitepaper by providing an implementer's perspective on how the platform behaves when deployed as a national payment switch.

The Mojaloop benchmarks establish the intrinsic throughput and latency characteristics of the software under controlled conditions. The results documented here focus instead on end-to-end operational behavior, including infrastructure orchestration, autoscaling dynamics, ingress handling, and resource contention. Together, these perspectives provide a complete view of both software capability and real-world deployability.

Deployment model and operational context

Infrastructure configuration

Abli's deployment reflects a horizontally scaled, cloud-native architecture designed to operate as a national instant payment switch. The environment uses managed services for persistence and messaging, a dedicated ingress tier, and dynamic autoscaling across compute resources.

Table 2: Infrastructure Configuration

Component	Configuration
K8s management nodes	4 t3.medium
Mojaloop Switch	6 c7i-flex.2xlarge. Scalable to 20 nodes
Mojaloop FSP SDK	Same nodegroup as Mojaloop Switch
MySQL - AWS RDS	db.m7g.xlarge
Kafka - AWS MSK	3 broker nodes - m7g.xlarge
Ingress	3 c7i-flex.xlarge. Scalable to 15 nodes
K6 Test Runner	4 m7i.xlarge

This configuration prioritizes elasticity and operational realism over isolated performance measurement. Compute resources are shared across services, and capacity is added dynamically in response to load.

Components deployed and configuration choices

The Mojaloop components deployed were configured to reflect production priorities. Core switch services, including the central ledger, settlement, API adapter, and quoting services, were deployed with minimum pod counts to support availability and were subject to autoscaling policies.

Non-essential testing components were disabled after functional validation. A custom DFSP SDK and ILP generator were used to simulate participant behavior. At the time of testing, these components did not enforce JWS or mTLS, allowing performance behavior of the switch services themselves to be observed without additional cryptographic overhead.

Table 3: Components deployed and configuration details

Component Group	Component Details
Mojaloop Switch	Central-ledger, central-settlement, ml-api-adapter and quoting services have a minimum of 3 pods running. Mojaloop Testing Toolkit was turned off after confirmation that the switch can perform transfers via the testing simulator and test scenario runners.
SDK and ILP Generator	Ablipay wrote a custom golang server to manage DFSPs requesting quoting and transfers to the switch. Ablipay also wrote a ILP generator to be used for confirming transfers. This service doesn't have JWS or mTLS in the test run.
Ingress Nginx Controller	Ingress Nginx Controllers have a minimum of 6 pods running with an HPA installed to scale to 60 pods
K6 Test Runner	The K6 test running uses k8s jobs and parallelizes it for simulating higher TPS numbers. By default 1 job can handle around 750 TPS

Operational Context

Unlike benchmark environments that minimize external variability, this deployment intentionally includes orchestration overhead, pod scheduling latency, ingress queuing, and resource contention. These characteristics are inherent to real-world switch operations and materially influence observed performance, particularly during rapid load changes.

As a result, this environment surfaces behaviors that are critical for national payment systems but are not visible in isolated benchmarks, including transient saturation, partial failure propagation, and recovery dynamics following burst traffic.

Performance scenarios and results

Abli executed two primary classes of performance tests to assess both dynamic behavior under burst conditions and stability under sustained national-scale load. The results presented in this section are reproduced unchanged from the Abli whitepaper.

Round 1 Tests

The first set of tests applied fixed start-to-end TPS ramps over defined durations. These scenarios were designed to observe system behavior during rapid load increases, including autoscaling responsiveness, latency behavior, and early signs of saturation.

Table 4: Round 1: TPS ramp performance results

Base TPS	Max TPS	Success Rate	Failure Rate	Duration	Average	p95	p99
500	500	100%	0%	5m	10.26ms	18.77ms	60.16ms
1000	1000	100%	0%	5m	9.17ms	16.54ms	52.48ms
1000	3000	99.98%	0.02%	5m	9.94ms	44.32ms	160.08ms
1000	5000	99.87%	0.13%	5m	10.75ms	119.65ms	1207.98ms
1000	1000	100%	0%	30m	9.59ms	15.44ms	43.07ms
1000	3000	99.98%	0.02%	30m	10.32ms	48.57ms	138.39ms
1000	5000	99.97%	0.03%	30m	27.21ms	48.00ms	294.50ms

At moderate ramp levels, the system maintained high success rates and stable latency distributions. As peak TPS increased, latency widened and failure rates increased slightly. These effects were most visible during aggressive ramps, where autoscaling lag resulted in temporary resource saturation before additional capacity became available.

Despite these transient effects, overall completion rates remained high, indicating that observed degradation was driven by scaling dynamics rather than by limitations in the transaction flow itself.

Fixed TPS and fixed transfer volume scenarios

The second set of tests applied a constant TPS with a fixed total number of transfers. These scenarios were designed to evaluate system stability, latency consistency, and error rates under prolonged load.

Table 5: Round 2: Fixed TPS and fixed transfer volume results

TPS	Number of Transfers	Success Rate	Failure Rate	Average	p95	p99
1000	1,000,000	99.98%	0.02%	13.57ms	45.15ms	77.88ms

Under sustained operation at 1000 TPS, the system processed one million transfers with a success rate of 99.98 percent. Latency remained stable throughout the test duration, demonstrating that the deployment can support prolonged high-volume operation once scaling has converged.

Summary of performance behavior

Across both scenarios, performance degradation did not increase linearly with load. Meaningful changes in behavior appeared primarily during rapid transitions rather than during steady-state operation. This highlights the importance of scaling responsiveness and resource headroom in synchronous services when operating at national scale.

Observed bottlenecks and failure behavior

Across test scenarios, saturation first appeared in synchronous components, particularly the quoting service and the API adapter, followed by ingress controllers under higher burst conditions. CPU and memory utilization were the dominant limiting resources, rather than network or disk I/O.

An increase in failed quoting requests consistently preceded broader transaction failures. Once quoting failures occurred, downstream transfers failed deterministically, and recovery was slow even after load subsided. This underscores the critical role of quoting services in overall system resilience.

Autoscaling behavior proved to be a decisive factor. At higher TPS levels, default scale-up patterns were insufficiently fast to absorb sudden spikes, resulting in temporary saturation before additional capacity was provisioned.

Metrics that matter for national payment systems

These results indicate that transactions per second is an incomplete measure of readiness for national payment systems. While peak throughput establishes upper bounds, it does not adequately capture user experience or operational resilience.

Successful transfers per TPS provides a more meaningful indicator, particularly under burst conditions. A system that sustains slightly lower peak throughput but maintains high completion rates and rapid recovery is operationally preferable to one that achieves higher TPS at the cost of instability.

Operational implications and forward testing roadmap

The combined evidence from Mojaloop benchmarks and Abli's operational tests demonstrates that Mojaloop is capable of supporting national-scale instant payment systems under realistic operating conditions. Production readiness, however, depends as much on deployment architecture and operational design as on software capability.

Operational results show that autoscaling speed, sufficient CPU and memory headroom in synchronous services, and early detection of saturation in quoting and ingress layers have a greater impact on real-world performance and recovery behavior than further tuning of messaging or persistence components alone. These factors become particularly critical during rapid load transitions rather than steady-state operation.

The tests documented in this report establish a stable operational envelope under the throughput levels exercised, while also identifying the system dynamics that must be addressed as transaction volumes increase. Importantly, these results are not intended to represent a final scale limit. Abli's testing strategy is explicitly designed to incrementally validate Mojaloop's behavior at progressively higher throughput levels, with a long-term objective of supporting sustained and burst traffic in the order of 10,000 transactions per second.

Future test phases will focus on extending throughput beyond current levels while preserving stability and recovery characteristics. Planned areas of focus include higher sustained and burst throughput testing, controlled failure and recovery scenarios, settlement-scale behavior as the number of participating DFSPs increases, and performance testing with security controls such as JWS and mTLS

enabled. Additional resilience testing will evaluate behavior under adverse but realistic conditions, including network latency, partial service degradation, and node-level failures.

Together, these efforts are intended to validate not only peak throughput, but also operational predictability, recovery behavior, and resilience at scale.

Liberia IIPS Use Case: Performance and Operational Validation

Executive Summary

Liberia's "Pay Na-Na" Inclusive Instant Payment System (IIPS) launched on December 16, 2025, as the first dual-currency real-time payment platform in the West African Monetary Zone (WAMZ). Built on Mojaloop, the system enables seamless transfers between mobile money providers in both Liberian Dollars (LRD) and US Dollars (USD).

In its first 40 days of operation, Pay Na-Na processed 676,783 transactions valued at USD 8.73 million equivalent, achieving 100% system uptime and 99.93% transaction success rate. These production results validate Mojaloop's capability to deliver reliable, real-time payment services in a national deployment.

This use case documents the performance validation and capacity testing conducted to confirm operational readiness. Testing validated stable performance at up to 20 transactions per second (TPS) on the production infrastructure, with confirmed headroom for growth as additional Digital Financial Services Providers (DFSPs) join the platform.

Note: This use case focuses exclusively on system performance and operational validation. Business adoption forecasts, regulatory approval processes, and DFSP integration timelines are managed through separate workstreams and are not covered in this document.

Project Background

"Pay Na-Na", a national, real-time, inclusive instant payment system (IIPS) in Liberia was launched on December 16, 2025, to allow users to send and receive money seamlessly across mobile money networks, such as Orange Money and Lonestar Cell MTN. Liberia's IIPS is implemented using the Mojaloop platform to support domestic real-time digital payments between participating Digital Financial Services Providers (DFSPs) in two currencies: Liberian Dollar (LRD) and US Dollar (USD). As the only country in West African Monetary Zone (WAMZ) with dual legal tender, Liberia's Inclusive Instant Payment System (IIPS) was launched with two currencies since Day 1.

Deployment Overview

The Pay Na-Na platform is deployed as a production-grade environment on Amazon Web Services (AWS), aligned with Mojaloop reference architecture and national payment system requirements. The deployment follows a highly available, horizontally scalable design with separate Staging and Production environments to support controlled testing and validation prior to live rollout.

DFPs initiate payments through an Open Source component called Payment Manager (PM4ML). The architecture separates a centrally hosted Mojaloop switching platform from DFSP-specific PM4ML deployments, with each DFSP operating independently in its own AWS VPC and connecting securely to the central platform via mTLS, JWS, and controlled network boundaries.

Components deployed

Deployed components included:

- **Mojaloop Core Services (v17):** Services required to support full three-phase transfer flows, including discovery, pre-validation/terms agreement, and clearing.
- **Mojaloop Stateful Resources:** Databases, Kafka, and other stateful dependencies required by Mojaloop services.
- **Mojaloop Connection Manager (MCM):** Central service responsible for establishing and managing secure connectivity between Mojaloop and Payment Managers, including:
    - mTLS certificate lifecycle and trust management
    - JWS key management and signing configuration
    - Secure channel setup for inter-participant communication
- **Security & Identity:**
    - Keycloak and Ory for authentication and authorization
    - Vault for secure secret and cryptographic key management
    - Vault PKI setup for certificate issuance and lifecycle management (where applicable)
- **Networking & Traffic Management:**
    - Istio for traffic management and secure service-to-service communication
    - External DNS for DNS automation
    - Cert-Manager for certificate automation
    - IP allowlisting at the ingress, restricting inbound access to approved PM4ML endpoints
- **Storage and Persistence:** Longhorn is used for general persistent volumes, while AWS EBS via the EBS CSI driver is used for IOPS-intensive workloads, such as databases and Kafka, to ensure predictable performance and durability.
- **Monitoring:** Platform-level metrics and observability, including Prometheus for metrics collection, Alertmanager for alerting, Loki for log aggregation, and Grafana for visualization.
- **PM4ML Application Services:** DFSP-specific Payment Manager services supporting payer and payee roles for two DFSP participants within the same PM4ML deployment.

Production System Performance

Since launching on December 16, 2025, Pay Na-Na has demonstrated strong operational performance and stability:

Transaction Volumes (First 40 Days)

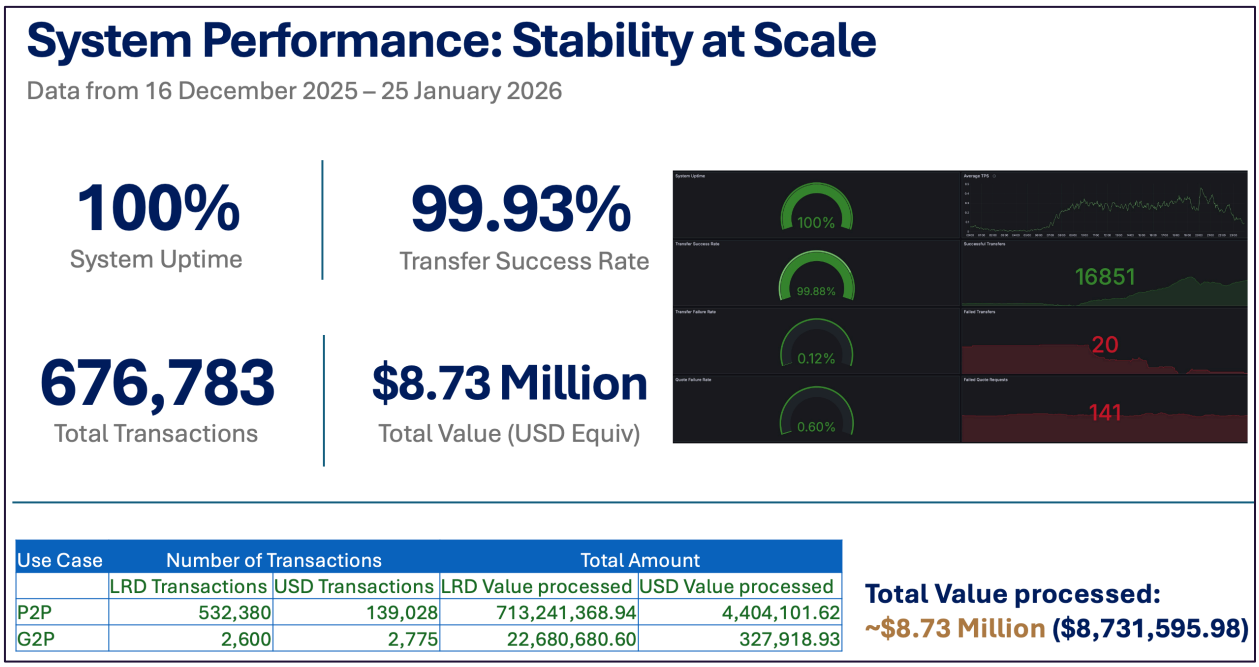
- Total transactions: 676,783 (LRD and USD combined)
- Total value: LRD 735,922,049.54 and USD 4,732,020.55 (USD 8.73M equivalent)
- Daily average: ~16,900 transactions

System Reliability

- System uptime: 100%
- Transaction success rate: 99.93%
- No service interruptions or downtime events

Infrastructure Validation

Current production volumes operate well within validated system capacity. Performance testing confirms the platform can comfortably support 20 transactions per second (TPS), providing substantial headroom for growth as adoption increases and additional DFSPs join the network.



Use Case	Number of Transactions		Total Amount	
	LRD Transactions	USD Transactions	LRD Value processed	USD Value processed
P2P	532,380	139,028	713,241,368.94	4,404,101.62
G2P	2,600	2,775	22,680,680.60	327,918.93

Total Value processed:
~$8.73 Million ($8,731,595.98)

Transaction Throughput

- Validated capacity: Up to 20 transactions per second (TPS)
- Infrastructure performance: Stable and consistent across test duration
- Production headroom: Current volumes (~0.2 TPS average) represent <1% of validated capacity

Response Time and Latency

- End-to-end transaction processing: Within real-time payment service level expectations
- Performance stability: No latency degradation observed under validated load conditions
- Note: Actual response times vary based on DFSP backend processing and network conditions

System Availability

- Architecture: High-availability design with no single point of failure
- Test stability: Zero service interruptions during performance validation exercises
- Production uptime: 100% since launch (40 days as of reporting date)

Error Rates and Reliability

- Test error rates: Within acceptable operational thresholds (<1%)
- Production success rate: 99.93% over first 40 days of operation
- Systemic failures: None detected during testing or production operations

Resource Utilization

- CPU utilization: Peak <60% during maximum validated load (20 TPS)
- Memory utilization: Within planned thresholds with headroom for growth
- Storage I/O: Stable after migration to EBS for IOPS-intensive workloads
- Network: No saturation or bottlenecks observed

Key Finding: Current production transaction volumes operate significantly below validated system capacity, confirming sufficient infrastructure headroom to support growth and additional DFSP onboarding without immediate scaling requirements.

Testing Protocol

Infrastructure Specifications	Test Results
Phase 1: Production-Range Testing (2-20 TPS)	
Switch (Mojaloop Hub): • Switch services: x3 c5a.4xlarge (16 vCPU, 32 GB RAM) • Switch load balancers: x2 t3.small (2 vCPU, 2 GB RAM) • Switch NAT: x1 t3.small (2 vCPU, 2 GB RAM) **PM4ML (Payer and Payee - 2 DFSPs):** • PM4ML services: x3 c5a.4xlarge (16 vCPU, 32 GB RAM) • PM4ML load balancers: x2 t3.small (2 vCPU, 2 GB RAM) • PM4ML NAT: x1 t3.small (2 vCPU, 2 GB RAM)	**Test Round 1:** **Objective**: Establish baseline system behavior under light load **Infrastructure**: Common configuration (3x c5a.4xlarge per component) **Results**: • Stable performance throughout test duration • No resource constraints observed • Provided baseline metrics for comparison **Observations**: System operated well within capacity. All components (compute, memory,

Infrastructure Specifications	Test Results
Load Generation: • JMeter test orchestrator: x1 t3.medium (2 vCPU, 4 GB RAM)	storage, messaging) showed low utilization with substantial headroom. **Test Round 2: Medium Load (10 TPS)** **Objective**: Validate performance at 5x baseline load **Infrastructure**: Common configuration with increased pod replicas **Configuration Changes**: Scaled Mojaloop and PM4ML pods to improve concurrency handling **Results**: • Stable performance maintained • Minor increase in CPU utilization but well within thresholds • No degradation in response times **Observations**: Infrastructure scaling through configuration changes (pod replication) successfully supported increased load without hardware changes.
Phase 2: Extended Capacity Testing (82 TPS)	
The following changes were applied to Phase 1 infrastructure: **Switch (Mojaloop Hub):** • Switch services: x3 c5a.8xlarge (32 vCPU, 64 GB RAM) • Switch load balancers: x2 t3.small (2 vCPU, 2 GB RAM) • Switch NAT: x1 t3.small (2 vCPU, 2 GB RAM) **PM4ML (Payer and Payee - 2 DFSPs):** • PM4ML services: x3 c5a.8xlarge (32 vCPU, 64 GB RAM) • PM4ML load balancers: x2 t3.small (2 vCPU, 2 GB RAM) • PM4ML NAT: x1 t3.small (2 vCPU, 2 GB RAM) **Storage and Messaging:** • IOPS-intensive workloads (databases and Kafka) were migrated to AWS EBS via the EBS CSI driver • Kafka topic count and partitioning were increased to support higher concurrency **Load Generation:** • JMeter test orchestrator: x1 t3.medium (2 vCPU, 4 GB RAM)	**Test Round 4: Extended Capacity Testing (82 TPS)** **Objective**: Explore architectural scaling limits and validate growth patterns **Infrastructure**: Scaled configuration (3x c5a.8xlarge, EBS storage, enhanced Kafka) **Initial Results**: Performance did not scale linearly with infrastructure improvements. Increasing concurrency beyond certain thresholds resulted in lower effective TPS. **Bottleneck Discovery**: Through systematic analysis using Prometheus, Grafana, and database query analysis, the team identified **MySQL row-level locking in the Central Ledger** as the primary constraint. **Root Cause**: In test scenarios with a **single DFSP pair**, all transactions contend for the same balance rows in the database. This creates an unavoidable serialization point where transactions must wait for lock release before proceeding. Higher concurrency increased lock contention, wait times, and retry operations rather than improving throughput.

Infrastructure Specifications	Test Results
	Why This Matters: • This behavior is consistent with Mojaloop's design principles • Mojaloop scales primarily by **participant count**, not by raw concurrency on a single DFSP pair • Real-world deployments with multiple DFSPs distribute balance updates across more rows, reducing lock contention • Testing with a single DFSP pair represents a constrained scenario not reflective of production environments with multiple participants **Optimization Efforts**: Despite the architectural constraint, several optimizations improved overall system performance: • Migrated IOPS-intensive workloads (databases, Kafka) to AWS EBS for predictable storage performance • Tuned Percona XtraDB configurations and Redis settings to improve database and cache efficiency • Increased OS-level networking parameters (nf_conntrack_max) to reduce packet drops • Refined JMeter load profiles to simulate realistic transaction patterns **Outcome**: While 82 TPS was achieved under optimized conditions, the testing validated that: • Production capacity (20 TPS) is well-supported by current infrastructure • Scaling beyond 20 TPS will be most effective through **adding DFSPs** rather than increasing infrastructure for existing participants • Current architecture provides clear growth path aligned with Mojaloop design

Performance testing was conducted across multiple throughput targets to validate production capacity and understand scaling characteristics. Testing was structured in two phases:

Summary of Observations

Through systematic performance testing and optimization, the following key findings emerged:

Infrastructure Performance

- The platform was **not primarily limited** by CPU, memory, Kafka throughput, Kubernetes scaling capabilities, or storage IOPS after infrastructure and configuration optimizations were applied
- These components demonstrated adequate headroom and scaling potential for production needs

Architectural Constraint Identified

- Performance was **ultimately constrained** by **MySQL row-level locking** within the Central Ledger database schema
- With a **single DFSP pair** (testing scenario), all transactions contend for the same balance rows, creating an unavoidable serialization point
- Increasing concurrency beyond optimal thresholds resulted in higher lock contention, increased retries, and lower effective TPS - even when additional compute and messaging capacity was available

Operational Implications

- This finding does **not** indicate a misconfiguration or database failure
- Rather, it represents an **expected outcome of a lock-bound workload** under a single-DFSP testing model
- Production environments with multiple DFSPs naturally distribute balance updates across more rows, reducing contention
- Current production capacity (20 TPS) is well-supported and provides significant headroom for growth

Conclusion

The performance testing results validate Mojaloop's design principles and operational readiness for the Liberia deployment.

Design Validation

- Mojaloop scales primarily by **participant count**, not by increasing concurrency for a single DFSP pair
- High-throughput scenarios require **multiple DFSPs**, which distribute balance updates across more database rows and reduce lock contention
- The observed MySQL locking behavior under single-DFSP testing does **not** indicate a system limitation but rather an **expected outcome** of testing under constrained conditions

Production Readiness Confirmed

- Current infrastructure comfortably supports validated capacity of 20 TPS
- Production transaction volumes (~0.2 TPS average) represent <1% of tested capacity
- Substantial headroom exists for organic growth in transaction volumes

Scaling Pathway

- As additional DFSPs join the network, the platform will naturally scale beyond single-DFSP testing constraints
- Infrastructure can be scaled horizontally when needed, with clear growth patterns established through testing
- Current architecture provides cost-effective operations while maintaining flexibility for future expansion

Stakeholder Confidence

These findings support operational readiness for current and near-term transaction volumes while establishing clear expectations for future growth scenarios.

Key Learnings and Recommendations

- **Observability is Critical:** Implement comprehensive monitoring from the beginning of any deployment. Real-time visibility into system performance enables rapid issue identification and resolution.
- **Performance Testing Drives Infrastructure Decisions:** Conduct performance validation early in the deployment lifecycle to right-size infrastructure before production launch. Use realistic transaction patterns and volumes based on projected adoption scenarios.
- **Cloud Deployment Enables Agility:** Cloud infrastructure is well-suited for national payment systems when properly architected for security, compliance, and operational resilience. The ability to scale resources dynamically provides cost and operational advantages.
- **Horizontal Scaling Provides Growth Path:** Design operations and monitoring to support horizontal scaling from the outset. This architectural approach aligns with Mojaloop's multi-participant model and supports sustainable growth.
- **Define Scaling triggers:** Infrastructure scaling will be considered when any of the following conditions are met:
- Sustained transaction volumes exceed 15 TPS (75% of validated capacity)
- Additional DFSPs onboard and drive cumulative volume growth
- New transaction types or use cases (e.g., bill payments, merchant acquiring) introduce different load patterns
- Operational metrics indicate resource saturation approaching planned thresholds

Conclusion

The performance and resilience exhibited during both steady-state and peak load conditions confirm that Mojaloop can support the demands of real-time payment systems at national and regional scale for a variety of use-cases and scenarios supported by Mojaloop.

The testing protocols, results and subsequent analyses demonstrate that the Mojaloop platform is capable of consistently achieving a throughput as required by adopters (in excess of 1,000 financial transactions per second (FTPS) where needed, with a minimal infrastructure setup of seven (virtual) nodes, (Note that the tests conducted by MLF in AWS infrastructure documented herein incurred a cost of approximately USD 3,500 per month)). These results underscore the platform's ability to deliver reliable, high-volume processing without requiring prohibitively expensive infrastructure investments and with negligible error rate (less than 1%) when configured correctly.

Importantly, the results observed in Mojaloop Labs were reinforced by independent testing carried out by at least two adopters - Ablipay, with a domestic deployment in the Philippines, and Thitsaworks implementing an IIPS solution for Liberia, further validating the platform's readiness for production-scale use.

As a result, the Mojaloop Foundation is confident that the platform is well-suited for deployment in countries with large populations seeking to modernize their domestic instant payment systems or offer inclusive instant payments for the first time, as well as in regional economic blocs aiming to facilitate cost-effective cross-border payments.

Glossary

DFSP (Digital Financial Services Provider)

An entity such as a bank, mobile money operator, fintech or microfinance institution that offers digital services to end users. In the Mojaloop ecosystem, DFSPs connect to the switch to send and receive payments on behalf of their customers.

Performance

Refers to how effectively the Mojaloop platform operates in terms of speed, responsiveness, stability, and efficiency when processing payments or executing payments -related taks e.g settlement. For the test, the following aspects were monitored:

- Transaction throughput
- Latency/Response time to process end-to-end payments
- System availability
- Error rate/system failure
- Resource utilization (percentage of CPU, memory, disk, and network bandwidth)

Resilience

System's capacity to maintain service continuity or recover quickly after failures, attacks, or unexpected load spikes. This includes features like redundancy, failover, and fault tolerance. For this testing protocol, the following KPIs were tracked: Recovery time and recovery point

FTPS

Financial Transfers Per Second: This metric measures the time taken for all the messaging involved in executing the clearing of a financial transfer, for a use case such as a Peer-to-Peer payment. This involves all three (typical) phases of a Mojaloop transfer: account lookup or discovery, quotation or agreement of terms and clearing or transfer (which includes preparing and fulfilling a transfer)

Acknowledgements

This work was sponsored by Amazon Web Services (AWS).

The Mojaloop Foundation acknowledges the support of AWS in providing the cloud infrastructure and technical capabilities used to conduct the performance and resilience testing described in this whitepaper. This support enabled the execution of large-scale, production-like test activities and contributed significantly to the analysis presented.

The Foundation also acknowledges the contributions of Ablipay and ThitsaWorks (alphabetical order), who provided real-world use cases and participated in testing using a deployment designed to closely reflect production conditions. Their involvement provided additional real-world operational context to the testing results.

The Mojaloop Foundation further thanks the system integrators, hub operators, and community contributors who provided technical feedback and supported the review of the testing approach and findings.

The test plan, requirements, and overall approach were developed collaboratively through the *Mojaloop Performance Whitepaper* workstream, with valuable contributions from the workstream committee members. The Mojaloop core team is also acknowledged for executing the performance testing, including deployment, configuration, tuning, performance profiling on AWS, and the capture and analysis of results.

About Mojaloop Foundation

The Mojaloop Foundation's mission is to increase financial inclusion by empowering organizations to create interoperable payment systems that enable digital financial services for all. As a 501(c)(3) nonprofit, the Foundation maintains Mojaloop—its open-source software and community—as public goods to serve global financial inclusion goals. Governments, banks, mobile money providers and other stakeholders can use Mojaloop to build or enhance inclusive real-time payment platforms. For more information, visit: https://mojaloop.io.

About AWS

Since launching in 2006, Amazon Web Services has been providing industry-leading cloud capabilities and expertise that have helped customers transform industries, communities, and lives for the better.

As part of Amazon, we strive to be Earth's most customer-centric company. We work backwards from our customers' problems to provide them with the broadest and deepest set of cloud and AI capabilities so they can build almost anything they can imagine.

Our customers—from startups and enterprises to non-profits and governments—trust AWS to help modernize operations, drive innovation, and secure their data.

mojaloop.io