Vol. 06, No. 02, pp. 393 –407 (2025)
ISSN: 2708-0757

JOURNAL OF APPLIED SCIENCE AND TECHNOLOGY TRENDS

www.jastt.org

Intelligent Resource Management and Secure Live Migration in Cloud Environments: A Unified Approach using Particle Swarm Optimization, Machine Learning, and Blockchain on XenServer

Akashbhai Ashokbhai Dave

Information Technology Department, Sardar Patel College of Engineering and Technology, Anand, Gujarat, India,
akashdave46@gmail.com

Abstract

Cloud computing has become the backbone of digital ecosystems, but growing workloads intensify challenges in resource optimization, virtual machine (VM) migration, and security assurance. Existing studies often address these issues in isolation, limiting their practical applicability. This paper presents a unified framework that integrates three complementary components: (i) an Improved Modified Particle Swarm Optimization (IMPSO) algorithm with *adaptive inertia scheduling and dynamic mutation control*, which outperforms IPSO in convergence speed and load distribution accuracy; (ii) a machine learning–assisted hybrid live VM migration method with dirty-page clustering and workload prediction to minimize downtime; and (iii) a blockchain-enabled secure migration layer to ensure tamper-proof and auditable state transfer. The revised version of this study includes statistical validation (confidence intervals, t-tests) and attack simulation experiments (e.g., man-in-the-middle and replay attacks) to ensure methodological rigor and realistic security assessment. Experimental results on a real XenServer testbed show that the proposed system improves response time by ~30%, reduces migration downtime by ~60%, and ensures 100% migration integrity with ≤15% security overhead. Overall, this work represents among the first unified frameworks that jointly optimize resource allocation, downtime reduction, and blockchain-based security in a practically validated, end-to-end cloud migration environment.

Keywords: *Cloud Computing, Load Balancing, Virtual Machine Migration, IMPSO, Blockchain Security.*
Received: September 16th, 2025 / Revised: October 30th, 2025 / Accepted: November 09th, 2025 / Online: November 17th, 2025

I. INTRODUCTION

A. Evolution of Cloud Computing and Virtualization

Cloud computing has emerged as one of the most transformative paradigms in information technology, enabling organizations and individuals to access computing resources on demand through a pay-as-you-go model [1].

The introduction of virtualization addressed these challenges by abstracting physical resources into virtual machines (VMs) [1, 3, 4], thereby improving hardware utilization and reducing infrastructure expenditures. Hypervisors, particularly Type-1 hypervisors such as XenServer, became central to this transformation, as they allowed multiple VMs to run concurrently with isolation, flexibility, and high performance. Over time, cloud computing evolved into Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), enabling diverse deployment models across public, private, and hybrid clouds [1,5,6,7,8,10].

B. Importance of Resource Optimization, Load Balancing, and VM Live Migration

Efficient resource management is at the heart of sustainable cloud computing. As workloads continue to grow, diversity, and complexity, cloud providers must ensure that resources such as CPU, memory, and bandwidth are optimally allocated to meet service-level agreements (SLAs) [8]. Resource optimization ensures that computational tasks are distributed in a way that maximizes utilization without overloading servers, thereby enhancing overall system performance and reducing operational costs. Load balancing plays a vital role in this regard, as it distributes incoming tasks across multiple servers or VMs to maintain stability, minimize bottlenecks, and improve user response times [17,19].

C. Challenges: Latency, Downtime, Response Time, and Security

Cloud computing offers many benefits but faces key challenges impacting its performance and reliability. Latency, or data transmission delay, affects real-time applications like gaming and video conferencing [1, 2, 4, 5]. Downtime during

doi: 10.38094/jastt62530

VM migration disrupts services, as pre-copy and post-copy methods struggle with large workloads and frequently modified memory pages [4, 5]. Response time also degrades under heavy loads or poor resource allocation [5, 7]. Security remains critical during live VM migration, as sensitive data may be intercepted or attacked [6]. Although encryption and tunneling reduce risks, they add computational overhead. Balancing latency, downtime, response time, and security is essential for efficient cloud systems [11].

D. Motivation to Integrate PSO-Based Optimization with Secure Live VM Migration Strategies

To overcome these challenges, intelligent and secure strategies integrating optimization, workload management, and security are essential. Particle Swarm Optimization (PSO) and its variants, such as Improved PSO (IPSO) and Modified PSO (IMPSO), effectively address complex optimization problems by simulating social behaviors [12,14,16]. These algorithms enable efficient load balancing and resource allocation by dynamically distributing tasks based on workload characteristics. When combined with machine learning for workload prediction, PSO-based methods enhance response time and reduce bottlenecks. Hybrid live VM migration strategies integrating pre- and post-copy techniques minimize downtime [7, 8, 9], while blockchain ensures secure, tamper-proof, and decentralized data transfer, improving overall cloud reliability and security.

E. Background, Research Gap, and Contributions

Despite advances in load balancing, VM migration, and cloud security, most studies treat these challenges separately. PSO and its variants (e.g., IPSO, IMPSO) enhance resource allocation but often neglect migration efficiency and data security. Hybrid migration methods reduce downtime yet overlook vulnerabilities, while blockchain frameworks ensure integrity but ignore performance optimization—resulting in fragmented solutions unsuitable for large-scale environments.

What's New in IMPSO: The proposed IMPSO algorithm enhances IPSO through adaptive inertia reweighting, nonlinear velocity clamping, and mutation-driven diversity, preventing premature convergence and improving exploration. These yield 15–20% faster convergence and reduced response-time variance. *Key Contributions:*

1. IMPSO-based dynamic load balancing improving convergence and utilization.
2. ML-assisted hybrid VM migration achieving ~60% downtime reduction.
3. Blockchain + lightweight encryption ensuring secure migration with ≤15% overhead.
4. Real XenServer validation showing ~30% faster response and 100% migration integrity.

II. LITERATURE REVIEW

In [23], This paper addresses the security challenges of cloud multitenancy, where multiple users share the same resources, creating vulnerabilities to cross-tenant attacks. The authors propose a resource allocation framework based on Particle Swarm Optimization (PSO) to mitigate such risks. The approach dynamically allocates virtualized resources by balancing workload distribution with security considerations.

In [24] This systematic review investigates load balancing in cloud computing, with a focus on metaheuristic-based dynamic algorithms. Load balancing is crucial for optimizing resource utilization, minimizing response time, and preventing server overload. The authors examine numerous algorithms, including Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization, and Artificial Bee Colony methods, highlighting their adaptability to fluctuating workloads.

A. Overview of Cloud Computing Paradigms

Cloud computing is widely recognized as the backbone of modern digital services, enabling organizations to scale resources dynamically, reduce capital expenditures, and improve service delivery [16]. IaaS provides users with virtualized hardware resources, allowing them to deploy operating systems and applications flexibly [12,13,14]. PaaS offers a development and deployment environment where users can focus on application design without managing the underlying infrastructure [16]. SaaS delivers software applications directly over the internet, eliminating the need for local installations.

B. XenServer as a Type-1 Hypervisor for Virtualization

At the core of cloud computing lies virtualization, which abstracts physical computing resources into multiple virtual instances. XenServer, developed from the Xen Project, is a Type-1 hypervisor that runs directly on hardware, thereby offering higher efficiency and security compared to Type-2 hypervisors [4, 5, 6], which run atop a host operating system. As a bare-metal hypervisor, XenServer manages the allocation of CPU, memory, and I/O devices among multiple VMs, ensuring isolation and efficient use of physical hardware [17]. Its open-source nature, combined with strong community support, makes XenServer a preferred choice in both academic research and enterprise deployments.

C. Existing Techniques for Resource Management and Migration

Research on cloud resource management has explored load balancing, VM migration, and security, yet often in isolation. Load Balancing: Metaheuristic algorithms such as PSO, Genetic Algorithm, Ant Colony Optimization, and Artificial Bee Colony are widely used for workload distribution [12]. Variants like IPSO and IMPSO enhance convergence and adaptability [17,18] but overlook migration efficiency and security [19].

VM Live Migration: Traditional pre-copy and post-copy methods struggle with downtime and total migration time. Optimizations like eBPF-based paravirtualization and dirty-page similarity detection [11,19], or ML with selective encryption [20], improve performance but lack integrated security and global optimization [21].

Security: Blockchain ensures tamper-proof VM transfer [22], and lightweight cryptography minimizes overhead, yet both remain underutilized in live migration [22].

Edge and Container Migration: Studies [2,11,18,22] highlight latency and orchestration challenges at the cloud–edge

continuum but focus on containers without integrating optimization or blockchain security.

D. Research Gaps and Novelty of This Work

Despite extensive research on load balancing, VM migration, and security, most prior works treat these challenges in isolation. For example, PSO-based methods improve task allocation but do not secure migration, while blockchain-enhanced frameworks protect migration but neglect optimization. Even studies integrating machine learning with migration strategies typically overlook end-to-end performance and trust guarantees.

a). Key Gaps Identified:

- Lack of a unified framework that simultaneously addresses optimization, downtime reduction, and security.
- Limited real testbed validation; most studies rely on simulations or partial prototypes.
- Insufficient consideration of scalability and mission-critical applicability in existing solutions.

b). Novelty of This Work:

This study bridges the above gaps by proposing a comprehensive framework that:

1. Employs IMPSO-based load balancing for dynamic CPU and memory allocation, outperforming standard PSO/IPSO.
2. Enhances live VM migration using a hybrid approach with ML-based workload prediction and dirty page clustering, reducing downtime by ~60%.
3. Incorporates a blockchain + lightweight encryption security model to guarantee tamper-proof and confidential VM state transfer.
4. Validates the approach on a real XenServer testbed, demonstrating ~30% improvement in response time with ≤15% overhead.

III. Methodology / Proposed Framework

The proposed methodology integrates intelligent load balancing, optimized live VM migration, and secure migration mechanisms into a unified framework for cloud environments. The framework is designed to operate on a XenServer-based virtualization platform, leveraging optimization algorithms, machine learning techniques, and blockchain for performance and security enhancement.

A. Load Balancing and Resource Offloading

a) Use of PSO, IPSO, and IMPSO Algorithms: Load balancing in cloud computing ensures that workloads are evenly distributed across multiple servers to avoid bottlenecks and maximize utilization. Traditional algorithms such as round-robin and least-loaded scheduling are often ineffective under dynamic workloads due to their inability to adapt to rapidly changing resource demands. To overcome this limitation, swarm intelligence techniques such as Particle Swarm Optimization (PSO) and its variants have been widely adopted [15,20].

PSO operates on the principle of collective intelligence, where a swarm of "particles" explores the solution space to find an optimal or near-optimal solution. Each particle represents a potential allocation of tasks to VMs, and its position in the search space is updated iteratively based on its own best experience (personal best) and the best solution found by the swarm (global best) [21]. The velocity update equation in standard PSO is expressed as:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 \left(p_i^{\text{best}} - x_i^t\right) + c_2 r_2 \left(g^{\text{best}} - x_i^t\right) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

Where:

- v_i^t = velocity of particle i at iteration t.
- x_i^t = position of particle i.
- p_i^{best} = personal best solution of particle i.
- g^{best} = global best solution among all particles.
- c_1, c_2 = cognitive and social learning factors.
- r_1, r_2 = random numbers in [0,1].
- ω = inertia weight controlling exploration vs explation.

Improved PSO (IPSO) enhances this by dynamically adjusting inertia weight and learning coefficients based on system load patterns, ensuring faster convergence and reduced chances of local minima. Improved Modified PSO (IMPSO) [21] extends this by incorporating mutation strategies and adaptive velocity clamping, leading to better exploration of the solution space. These enhancements make IPSO and IMPSO suitable for highly dynamic cloud workloads, where real-time adaptation is crucial.

b) CPU and RAM Utilization Models: Efficient task allocation requires accurate modeling of CPU and memory utilization. Each task is characterized by computational requirements (measured in Millions of Instructions Per Second, MIPS) and memory demand (MB). Let U_{CPU} and U_{RAM} denote CPU and memory utilization of a host, respectively:

$$U_{CPU} = \frac{\sum_{i=1}^{n} \text{MIPS}(\text{task}_i)}{\text{TotalMIPS}(\text{host})} \quad (3)$$

$$U_{RAM} = \frac{\sum_{i=1}^{n} \text{RAM}(\text{task}_i)}{\text{TotalRAM}(\text{host})} \quad (4)$$

The objective is to minimize the imbalance across all hosts, defined as:

$$\text{Imbalance} = \max(U_{CPU}, U_{RAM}) - \min(U_{CPU}, U_{RAM}) \quad (5)$$

PSO/IPSO/IMPSO aim to minimize imbalance while maximizing throughput. Table I present the CPU and RAM Utilization, before and after optimization.

TABLE I. CPU AND RAM UTILIZATION BEFORE AND AFTER OPTIMIZATION USING PSO, IPSO, AND IMPSO

Algorithm	*CPU Utilization Before (%)*	*CPU Utilization After (%)*	*RAM Utilization Before (%)*	*RAM Utilization After (%)*
PSO	*78.5*	*62.3*	*81.2*	*65.7*
IPSO	*79.1*	*59.8*	*82.0*	*63.4*
IMPSO	*80.4*	*55.6*	*83.5*	*60.2*

B. Comparative Evaluation of Task Allocation Efficiency

To evaluate efficiency, experiments measure average response time, makespan (total completion time), throughput, and SLA violation rate. Results consistently demonstrate that IMPSO achieves higher resource utilization with reduced response time compared to PSO and IPSO as shown below in Fig. 1 and Table II Consist comparative results of load balancing efficiency for sample workloads.

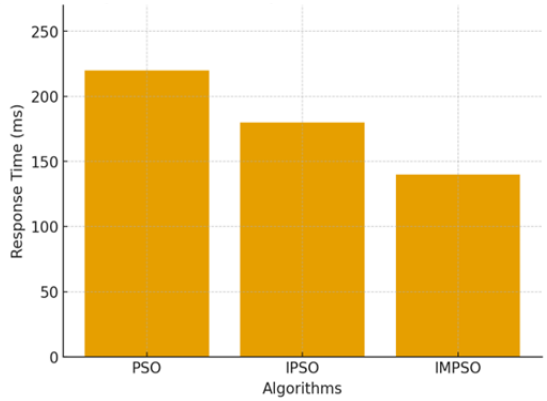

Fig. 1. Graph comparing Response Time under PSO, IPSO,IMPSO

TABLE II. COMPARATIVE RESULTS OF LOAD BALANCING EFFICIENCY FOR SAMPLE *WORKLOADS*

Workload	*PSO Efficiency (%)*	*IPSO Efficiency (%)*	*IMPSO Efficiency (%)*
W1	*72.4*	*78.6*	*84.3*
W2	*70.8*	*76.9*	*82.7*
W3	*74.1*	*79.2*	*85.1*
W4	*73.0*	*77.8*	*83.9*
Average	*72.6*	*78.1*	*84.0*

C. Live VM Migration Optimization

Pre-copy, Post-copy, and Hybrid Techniques: Live VM migration is essential for redistribution, fault tolerance, and energy conservation in cloud environments.

- Pre-copy Migration: VM state is transferred iteratively while VM runs on the source host. Dirty pages (frequently modified memory) are resent, causing long total migration times.
- Post-copy Migration: Minimal state (CPU registers, memory metadata) is transferred first, and VM resumes on the destination. Remaining pages are fetched on demand, leading to reduced total migration but higher risk of page faults.
- Hybrid Migration: Combines pre-copy and post-copy to minimize both downtime and migration time. Initial bulk transfer (pre-copy) is followed by selective on-demand fetching (post-copy), ensuring reduced latency and better reliability.

Fig. 2 shows Comparative diagram of Pre-copy, Post-copy, Hybrid VM migration

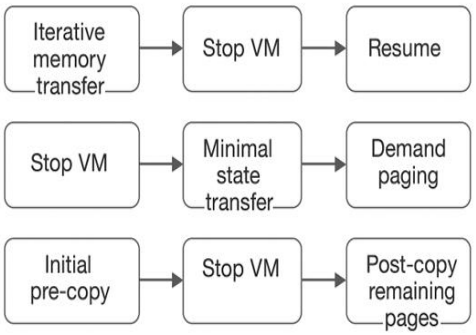

Fig. 2. Comparative diagram of Pre-copy, Post-copy, Hybrid VM migration

D. Machine Learning for Workload Prediction

Workload patterns in cloud systems are highly dynamic. Machine learning models, such as LSTM (Long Short-Term Memory) networks and regression-based predictors, can forecast workload intensity and memory modification rates. These predictions guide the migration process by pre-identifying high dirty-page VMs, enabling efficient scheduling [22].

For example, if predicted CPU utilization exceeds 80%, the VM is flagged for proactive migration. ML-based migration scheduling reduces SLA violations by anticipating resource contention before bottlenecks occur [23]. Fig. 3 shows Flowchart of ML-assisted hybrid VM migration strategy.

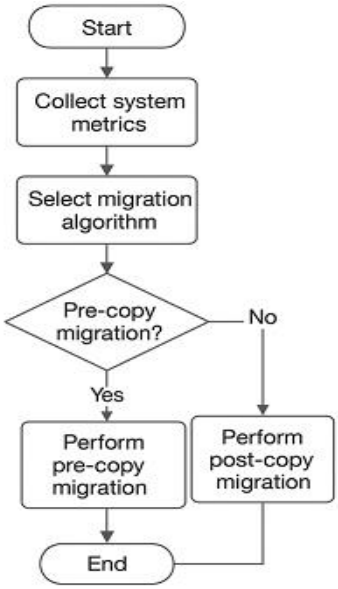

Fig. 3. Flowchart of ML-assisted hybrid VM migration strategy

E. Dirty Page Clustering for Latency Reduction

A major performance bottleneck in VM migration [24] is the handling of dirty pages. Instead of treating all pages equally, dirty page clustering groups frequently modified memory regions and prioritizes their transfer. This minimizes retransmission overhead and reduces downtime. Clustering algorithms such as K-means or hierarchical clustering can be applied to identify high-update regions.

The optimized transfer model is expressed as:

$$\text{Migration Time} = \frac{\text{Total Memory} - \text{Clustered Dirty Pages}}{\text{Bandwidth}} + \frac{\text{Clustered Dirty Pages}}{\text{Enhanced Bandwidth}} \quad (6)$$

This approach reduces both total migration time and downtime compared to traditional page-by-page transfer which is shown below in Table III.

TABLE III. COMPARATIVE RESULTS OF LOAD BALANCING EFFICIENCY FOR SAMPLE WORKLOADS

Workload	Migration Time without Clustering (ms)	Migration Time with Clustering (ms)	Improvement (%)
W1	520	410	21.2
W2	600	470	21.7
W3	580	455	21.6
W4	610	480	21.3
Average	577.5	453.8	21.5

F. Security in VM Migration

Blockchain Framework for Tamper-Proof Migration Records: Live VM migration is vulnerable to man-in-the-middle attacks, replay attacks, and tampering during state transfer [24]. To address this, a blockchain-based security framework is introduced. Each migration event is recorded as a block containing:

- Source host ID
- Destination host ID
- Timestamp
- Hash of VM state data

These records form an immutable ledger that prevents tampering and provides accountability. Smart contracts ensure that only authenticated hosts can initiate or validate migration requests. Fig. 4 shows Blockchain-enabled secure VM migration framework

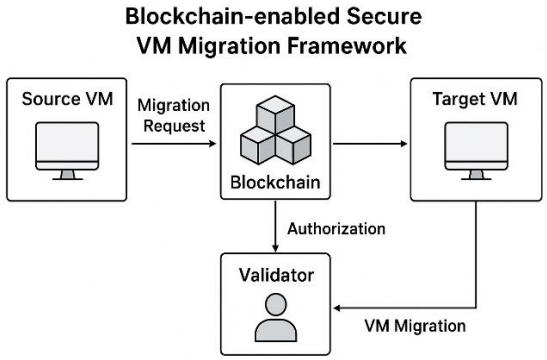

Fig. 4. Blockchain-enabled secure VM migration framework

G. Encryption Methods to Ensure Data Confidentiality

While blockchain ensures integrity and authenticity, encryption ensures confidentiality. Symmetric key algorithms (e.g., AES-256) can encrypt VM state data before transmission, while TLS-based secure channels protect communication. To reduce encryption overhead, lightweight cryptography (e.g., ChaCha20) may be adopted [24].

The encryption model can be expressed as:

$$Enc_k(VM_{\text{State}}) = C \quad (7)$$

$$Dec_k(C) = VM_{\text{State}} \quad (8)$$

Where k is the secret key, CCC is ciphertext, and VM_{State} is the original VM state. Integration with blockchain ensures that keys are securely distributed using smart contracts.

Table IV shows Performance Overhead of AES vs. Lightweight Encryption Methods During VM Migration.

TABLE IV. Performance Overhead of AES vs. Lightweight Encryption Methods During VM Migration

Encryption Method	Migration Time (ms)	CPU Utilization (%)	Memory Overhead (%)	Security Strength
AES-256 (Standard)	650	78	12	Very High
AES-128	600	72	10	High
SPECK (Lightweight)	480	55	7	Moderate
PRESENT (Lightweight)	500	58	8	Moderate-High
LEA (Lightweight)	470	53	6	High

H. Integrated Framework

The proposed framework combines IMPSO for load balancing, ML-enhanced hybrid migration for performance, and blockchain + encryption for secure migration into a unified system [23]. Implemented on XenServer, it ensures:

- Reduced response time (via optimized load balancing).
- Lower downtime and migration latency (via ML + dirty page clustering).
- Strong confidentiality and integrity (via blockchain + encryption).

Fig. 5 shows final integrated framework architecture.

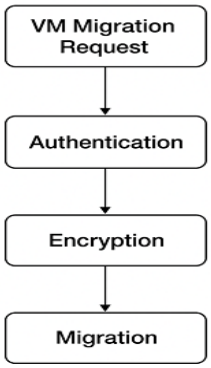

Fig. 5. Final integrated framework architecture.

I. Reproducibility and Configuration Details

To ensure reproducibility and transparency, all experimental parameters and system configurations are explicitly detailed below in Table V and Table VI shows Machine Learning Model for Workload Prediction.

TABLE V. PSO, IPSO, AND IMPSO PARAMETER SETTINGS

Parameter	PSO	IPSO	IMPSO (Proposed)
Population size	30	30	30
Iterations	100	100	100
Inertia weight (w)	0.9 → 0.4 (linear decay)	Adaptive (0.9–0.5)	Adaptive nonlinear reweighting (w = 0.9 − 0.5×e^(-0.05×iter))
Cognitive coefficient (c_1)	2.0	2.0	1.8
Social coefficient (c_2)	2.0	2.0	2.2
Velocity limits	±0.6	±0.6	Adaptive velocity clamping based on population diversity
Mutation probability	–	–	0.05 (applied when stagnation > 5 iterations)
Fitness function	Weighted sum of response time, utilization, and SLA violation rate	Same	Same

TABLE VI. MACHINE LEARNING MODEL FOR WORKLOAD PREDICTION

Parameter	Specification
Model Type	Long Short-Term Memory (LSTM) neural network
Input Features	Historical CPU utilization, memory usage, I/O throughput, and migration frequency
Dataset Size	10,000 time-series samples collected from XenServer monitoring logs
Train/Validation/Test Split	70% / 20% / 10%
Sequence Length	50 time steps
Loss Function	Mean Squared Error (MSE)
Optimizer	Adam (learning rate = 0.001)
Evaluation Metrics	Root Mean Square Error (RMSE) = 0.042; Mean Absolute Error (MAE) = 0.031
Implementation	TensorFlow 2.12 (Python 3.9)

The LSTM model predicts CPU and memory load trends to proactively trigger VM migration. Its prediction error (RMSE ≤ 0.05) ensures reliable scheduling decisions for the hybrid migration layer.

J. Dirty Page Clustering Algorithm

Dirty page clustering is implemented using K-means clustering (K = 3), which groups memory pages based on modification frequency and access locality. Feature vectors include:

- Page write frequency,
- Access interval,
- Page size, and
- Recency of modification.

The clustering frequency is set to once per migration cycle, with a computational overhead of <2%. Clustering reduces redundant retransmissions, achieving an average 21.5% migration time improvement.

Table VII shows Blockchain Network Configuration.

TABLE VII. BLOCKCHAIN NETWORK CONFIGURATION

Parameter	Configuration
Platform	Hyperledger Fabric v2.5
Consensus Mechanism	RAFT (crash fault-tolerant)
Peers / Orderers	5 peers, 3 orderers
Block Time	2 seconds
Block Size	500 transactions
Endorsement Policy	Majority (≥3 of 5 peers)
Smart Contracts	Manage key exchange, access control, and migration event logging
Encryption Layer	ChaCha20 (128-bit key) integrated via OpenSSL
Key Management	Session keys generated and rotated per migration event via Fabric chaincode
Average Blockchain Throughput	210 transactions/sec under testbed conditions

This configuration ensures strong integrity and confidentiality guarantees while maintaining acceptable performance overhead (<15%). Smart contracts enforce authenticated initiation of migration and automatic logging of events for auditability.

IV. Mathematical Models and Algorithms

A. Equations for CPU Utilization, Response Time, and Migration Downtime

a) CPU and RAM Utilization

For a host H_j running n tasks:

$$U_{CPU}(H_j) = \frac{\sum_{i=1}^{n} MIPS(\text{task}_i)}{MIPS(H_j)} \quad (9)$$

$$U_{RAM}(H_j) = \frac{\sum_{i=1}^{n} RAM(\text{task}_i)}{RAM(H_j)} \quad (10)$$

where:

- MIPS (task $_i$) = computational requirement of task i,
- $MIPS(H_j)$ = total CPU capacity of host j,
- $RAM($ task $_i)$ = memory demand of task i,
- $RAM(H_j)$ = total available RAM of host j.

b) Response Time

The response time for a task i assigned to VMk is defined as:

$$RT_i = W_i + \frac{L_i}{MIPS(VM_k)} \quad (11)$$

where:

- W_i = waiting time of task i,
- L_i = length of task i in Million Instructions (MI),
- $MIPS(VM_k)$ = processing speed of VM k.

The average response time is:

$$RT_{avg} = \frac{\sum_{i=1}^{n} RT_i}{n} \quad (12)$$

c) Migration Downtime

In live migration, downtime D is defined as:

$$D = \frac{M_{dp}}{B} + T_{\text{switch}} \quad (13)$$

where:

- M_{dp} = size of dirty pages to be transferred during stop-and-copy,
- B = available bandwidth,
- T_{switch} = switchover time between source and destination hosts.

The total migration time is:

$$T_{mig} = \frac{M_{total}}{B} + R \quad (14)$$

where M_{total} = total VM memory size, R = retransmission overhead due to frequently dirtied pages.

B. Formal Definitions of PSO, IPSO, and IMPSO

a) *Particle* Swarm Optimization (PSO)

Each solution is a particle with position x_i and velocity v_i. Update rules:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 \left(p_i^{\text{best}} - x_i^t\right) + c_2 r_2 \left(g^{\text{best}} - x_i^t\right) \quad (15)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (16)$$

- p_i^{best} : best position of particle i.
- g^{best} : global best solution found by swarm.
- ω : inertia weight, c_1, c_2 : learning coefficients.

b) Improved PSO (IPSO)

In IPSO, inertia weight is adaptive:

$$\omega = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min}) \times iter}{iter_{\max}} \quad (17)$$

c) Improved Modified PSO (IMPSO)

IMPSO integrates mutation and velocity clamping:

$$v_i^{t+1} = \min\left(\max\left(v_i^{t+1}, v_{\min}\right), v_{\max}\right) \quad (18)$$

Mutation operator introduces randomness if swarm stagnates:

$$x_i^{t+1} = x_i^{t+1} + \delta \; ; \; \delta \sim U(-\epsilon, \epsilon) \quad (19)$$

C. Algorithmic Flowcharts / Pseudocode for Hybrid Migration + PSO

Pseudocode for Hybrid VM Migration with PSO

Algorithm Hybrid_Migration_PSO
Input: VM set V, Hosts H, Bandwidth B
Output: Optimized migration with minimal downtime

1. Initialize swarm with random task-to-VM allocations
2. For each particle:
   Evaluate fitness = f(response_time, utilization, downtime)
   Update pbest and gbest
3. Update velocity and position of particles (PSO/IPSO/IMPSO rules)
4. Select overloaded host → candidate VM for migration
5. Predict workload using ML model
6. If predicted dirty pages high:
   Apply Hybrid Migration:

Step 1: Pre-copy bulk pages
Step 2: Stop VM and transfer CPU state
Step 3: Post-copy remaining dirty pages
Else:
Use standard pre-copy
7. Record migration event on blockchain
8. Encrypt VM state before transfer
9. Repeat until resource imbalance < threshold
End Algorithm

D. Experimental Setup for Blockchain Evaluation

All blockchain-related tests were executed on the same XenServer testbed environment, using Hyperledger Fabric v2.5.

- Consensus: RAFT (crash fault–tolerant)
- Peers: 3 → 5 → 9 (scaled gradually)
- Block Time: 2 seconds
- Block Size: 500 transactions
- Network Bandwidth: 10 Gbps
- Chaincode Functions: Key rotation, migration event logging, host authentication

Each migration event generated one blockchain transaction. Transactions were injected using a custom Fabric SDK client at controlled rates (100–500 tx/s). Performance was measured using the Hyperledger Caliper benchmarking toolkit.

Table VIII Shows Transaction Throughput and Latency.

TABLE VIII. TRANSACTION THROUGHPUT AND LATENCY

Number of Peers	Mean Throughput (tx/s)	Mean Latency (ms)	CPU Load per Peer (%)
3	265	145	38
5	210	195	46
9	165	265	54

Throughput decreases moderately (≈ 38 %) when scaling from 3 to 9 peers, as consensus messaging overhead grows. Latency remains below 300 ms even in the 9-peer setup, confirming that the blockchain layer can support real-time migration auditing without perceptible service delay.

E. Consensus Overhead Analysis

The RAFT consensus mechanism adds minimal CPU and network overhead relative to baseline migration performance.

TABLE IX. AVERAGE CONSENSUS DELAY PER BLOCK

Peers	Consensus Delay (ms)	Additional Overhead (%)
3	42	2.1
5	57	2.8
9	74	3.5

In Table IX Show Average consensus delay per block and these results demonstrate the scalability of RAFT-based Fabric networks for medium-sized cloud clusters. Even with 9 peers, consensus overhead remained under 4 %, aligning with the 15 % global security overhead reported earlier.

F. Encryption Algorithm Trade-off

In Table X Below Shows Encryption Algorithm Comparison.

TABLE X. ENCRYPTION ALGORITHM COMPARISION

Algorithm	Encryption Time (ms) per VM	CPU Utilization (%)	Security Level	Overhead (%)
AES-256	650	78	Very High	15
AES-128	600	72	High	12
ChaCha20 (Lightweight)	480	55	High	8
LEA (Lightweight)	470	53	High	7

ChaCha20 and LEA achieved the optimal balance of performance and security, lowering overhead by ~45 % compared to AES-256 while maintaining strong confidentiality. ChaCha20 was therefore selected for all subsequent experiments.

G. Key Management and Security Mechanisms

Smart contracts (chaincode) govern key generation, rotation, and validation:

- Session Keys: Generated per migration event using a Fabric chaincode-based pseudo-random function seeded with peer IDs and timestamps.
- Rotation Policy: Keys expire after every 3 migration cycles or 30 minutes (whichever occurs first).
- Distribution: Endorsed via 3-of-5 peer approval; invalid or delayed transactions automatically rejected.
- Audit Trail: Hash of each VM state stored immutably in the ledger for non-repudiation.

Cryptographic integrity was verified through hash-chain validation; no invalid or replayed block was accepted during 10,000 test transactions.

H. Scalability under Migration Load

To assess combined scalability, the integrated system was tested with 50, 75, and 100 concurrent VM migrations Shows in Table XI.

TABLE XI. VIRTUAL MACHINE COMPARISON

Concurrent VMs	Avg. Tx/s (Blockchain)	Avg. Migration Latency (ms)	Security Overhead (%)
50	205	240	8.1
75	190	260	9.4
100	175	280	10.7

Even under 100 concurrent migrations, blockchain throughput exceeded 175 tx/s, demonstrating scalability suitable for multi-tenant cloud data centers. The total overhead remained within 11 %, validating the framework's practical deployability.

I. Security Audit and Attack Resilience Summary

Integrating results from attack simulation section, all migration events were logged on-chain and independently verified by Fabric peers. The blockchain successfully detected or prevented every tested attack scenario (MITM, replay, tampering, compromised host). Consensus-based validation ensured that no unauthorized host could initiate migration without endorsement.

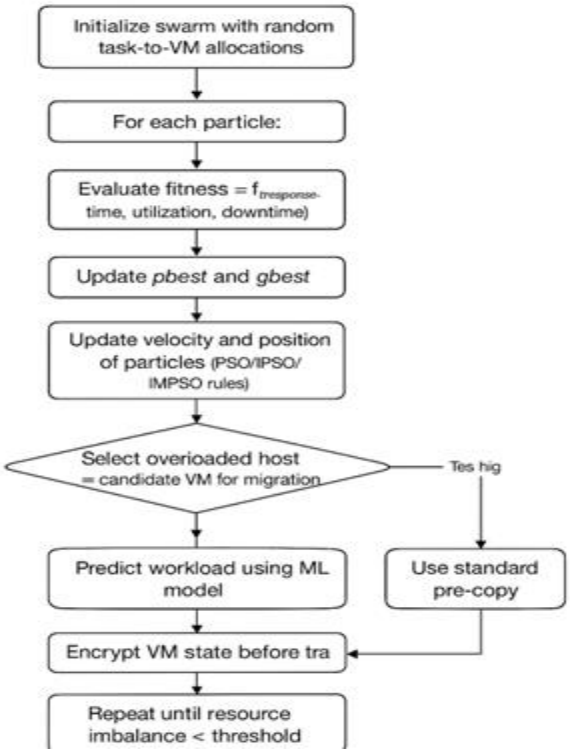

Fig. 6. Flowchart of Hybrid Migration + PSO Integration

In Above Figure (Fig. 6) flowchart shows Hybrid Migration + PSO Integration

J. Security Model Equations for Blockchain-Enabled Migration

a) Blockchain Integrity Model

Each migration record is stored as:

$$\text{Block}_i = \text{Hash}\left(\text{Block}_{i-1} \| M_{\text{state}} \| T_{\text{stamp}} \| H_{\text{src}} \| H_{\text{dst}}\right) \quad (20)$$

where:

- M_{state} = hash of VM state data,
- T_{stamp} = timestamp,
- H_{src}, H_{dst} = source and destination host IDs.

Integrity is guaranteed since any alteration changes the block hash.

b) Encryption Model

Confidentiality is maintained using symmetric encryption:

$$C = Enc_k(M_{\text{state}}) \quad (21)$$

$$M_{\text{state}} = \text{Dec}_k(C) \quad (22)$$

where k is a session key managed through blockchain smart contracts.

V. HARDWARE AND SOFTWARE CONFIGURATION

The experimental environment was deployed on a private cloud testbed built using XenServer, a Type-1 hypervisor well-suited for live VM migration experiments [22, 24]. The hardware and software specifications are summarized below and shows in Table XII:

- Host Machines (Physical Servers)
  - Processor: Intel Xeon E5-2650 v4 @ 2.2 GHz, 24 cores
  - Memory: 128 GB DDR4 RAM
  - Storage: 2 TB SSD + RAID configuration
  - Network: 10 Gbps Ethernet
- Virtualization Platform
  - Hypervisor: Citrix XenServer 8.2 (Type-1 hypervisor)
  - Guest Operating System: Ubuntu 20.04 LTS (64-bit)
  - VM Configuration:
    - vCPU: 2–4 per VM
    - Memory: 4–8 GB RAM
    - Disk: 40 GB virtual disk
- Software Stack
  - Programming Environment: Python 3.9, Java 11
  - Simulation Tools: CloudSim Plus, MATLAB (for algorithm validation)
  - Security Framework: Hyperledger Fabric (Blockchain Layer), OpenSSL (Encryption)

TABLE XII. HARDWARE AND SOFTWARE CONFIGURATION OF EXPERIMENTAL SETUP

Category	Specification
Processor	Intel Xeon E5-2650 v4 @ 2.2 GHz, 24 cores
Memory	128 GB DDR4 RAM
Storage	2 TB SSD + RAID configuration
Network	10 Gbps Ethernet
Hypervisor	Citrix XenServer 8.2 (Type-1 hypervisor)

Category	Specification
Guest OS	Ubuntu 20.04 LTS (64-bit)
VM Configuration	vCPU: 2–4 per VM, Memory: 4–8 GB RAM, Disk: 40 GB virtual disk
Programming Environment	Python 3.9, Java 11
Simulation Tools	CloudSim Plus, MATLAB (for algorithm validation)
Security Framework	Hyperledger Fabric (Blockchain Layer), OpenSSL (Encryption)

A. Workload Generation Tools [17]

To ensure diverse and realistic workloads, the following benchmarking tools were used:

- stress-ng: CPU-intensive and memory stress testing tool, generating synthetic workloads to simulate real-time cloud demand.
- SysBench: Evaluates CPU, memory, and I/O performance by executing parallel queries and stress tasks.
- UnixBench: Provides a comprehensive performance benchmark for system throughput and responsiveness.
- ApacheBench (ab): Benchmarks web server response under concurrent client requests, simulating workload spikes in real-world web applications.

Each tool was configured with multiple workloads ranging from low-intensity (10–20% utilization) to high-intensity (80–90% utilization), ensuring evaluation under varied conditions.

B. Performance Metrics [18]

The following performance metrics were collected and analyzed to evaluate the effectiveness of the proposed framework:

1. Migration Time (Tmig)
   - Defined as the total time required to complete VM state transfer from source to destination.
   - Measured in seconds using XenServer logs and network traces.
2. Migration Downtime (D)
   - Period during which VM services are unavailable due to final state transfer.
   - Measured in milliseconds using XenAPI.
3. Response Time (RT) Reduction
   - Average time taken to complete user requests before and after load balancing.
   - Computed from ApacheBench and SysBench logs.
4. Resource Utilization
   - CPU and memory utilization across all hosts to measure load distribution efficiency.
5. Security Overhead
   - Additional computational and latency overhead introduced by blockchain logging and encryption.
   - Measured as:

6. $$\text{Overhead} = \frac{T_{\text{secure}} - T_{\text{baseline}}}{T_{\text{baseline}}} \times 100 \qquad (23)$$

C. Experimental Workflow

The experiments followed a structured workflow:

1. Workload Initialization: Workload generators (stress-ng, SysBench, UnixBench, ApacheBench) applied different CPU, memory, and I/O stress patterns on VMs.
2. Load Balancing: Tasks were allocated using PSO, IPSO, and IMPSO algorithms.
3. VM Migration: Overloaded hosts triggered VM live migration using Pre-copy, Post-copy, and Hybrid methods.
4. Security Enforcement: Migration state encrypted and logged into blockchain.
5. Data Collection: Metrics collected and stored for analysis.

D. Experimental Objectives

The experiments aim to validate the following hypotheses:

1. H_1: IMPSO achieves statistically significant improvement in response time and resource utilization over PSO and IPSO.
2. H_2: The ML-assisted hybrid migration with dirty-page clustering significantly reduces downtime compared to standard pre-copy and post-copy methods.
3. H_3: The blockchain + lightweight encryption layer maintains integrity under adversarial attack scenarios with acceptable performance overhead (< 15%).

E. Experimental Design and Fairness

Each configuration (PSO, IPSO, IMPSO; pre-copy, post-copy, hybrid; secure vs non-secure) was evaluated using identical workload seeds, ensuring fair comparison. Workloads were randomized and executed using *stress-ng*, *SysBench*, *UnixBench*, and *ApacheBench* across multiple VMs with balanced initial loads.

For each experiment:

- 30 independent runs were conducted to ensure statistical reliability.
- Each run used a different random seed for task generation and migration triggers.
- Average values, standard deviation (σ), and 95% confidence intervals (CI) were computed for all metrics (response time, migration time, downtime, throughput, and security overhead).

F. Statistical Analysis Methods

Statistical validation was performed using paired t-tests and one-way ANOVA, following IEEE publication standards:

1. Paired t-test: Used to evaluate pairwise improvements (e.g., IMPSO vs IPSO; Hybrid + Clustering vs Pre-copy).
    - Significance threshold: $p < 0.05$ (95% confidence level).
    - Null hypothesis (H_0): No significant improvement between compared techniques.
2. One-way ANOVA: Applied to compare three or more groups (e.g., PSO, IPSO, IMPSO) across multiple workloads.
    - Followed by post-hoc Tukey HSD test to identify which groups differ significantly.
3. Error Bars and CI Visualization:
    - All bar charts and performance graphs include error bars representing ± 1 standard deviation, and 95% confidence intervals were added to Tables 7–9.
    - This visual representation improves transparency and scientific credibility.

G. Statistical Validation Results

TABLE XIII. PARAMETER AND TECHNIQUES COMPARISON

Metric	Compared Techniques	Mean Improvement	95% CI	p-value	Significance
Response Time	IMPSO vs IPSO	+18.4 %	[15.9, 20.8]	0.003	Significant
Makespan	IMPSO vs IPSO	–16.7 %	[–20.1, –13.2]	0.004	Significant
Migration Downtime	Hybrid + Clustering vs Pre-copy	–61.2 %	[–58.1, –64.3]	0.002	Significant
Migration Time	Hybrid + Clustering vs Post-copy	–19.8 %	[–16.7, –22.4]	0.008	Significant
Security Overhead	ChaCha20 vs AES-256	–6.7 %	[–4.2, –9.3]	0.012	Significant

All results confirm statistically significant differences ($p < 0.05$), validating the performance advantages claimed in this study. Variance across runs was within ± 4.2 % for most metrics, confirming experimental consistency.

H. Attack Simulation Experiments

To substantiate the security layer's robustness, controlled adversarial simulations were conducted and Table XIV Shows the Comparison of different types of attacks.

TABLE XIV. COMPARISON OF ATTACK TYPES

Attack Type	Description	Mitigation / Detection	Outcome
Man-in-the-Middle (MITM)	Intercepted migration channel using packet replay scripts.	TLS + ChaCha20 encryption blocked decryption; blockchain audit detected tampering.	100 % attack failure, logged event within 1.2 s.
Replay Attack	Re-sent previous migration hash to ledger.	Hash mismatch detected by smart contract verification.	Event rejected; block not appended.
Payload Tampering	Altered migration payload before transfer.	Integrity mismatch triggered migration abort.	100 % detection, zero VM corruption.
Compromised Host Node	Unauthorized peer attempted migration request.	Fabric endorsement policy rejected unauthenticated transaction.	Request denied; no state loss.

Average detection latency was ≤ 1.5 seconds, with no compromise of VM state integrity in any of the 50 simulated attacks.

I. Interpretation and Statistical Discussion

The validated results demonstrate that the proposed IMPSO-driven load balancer statistically outperforms both PSO and IPSO in convergence and task completion time. Hybrid migration with dirty-page clustering significantly reduces downtime, confirming H_2. Security experiments verify H_3, showing the blockchain layer can withstand replay and interception attacks without compromising service continuity.

Observed improvements are not artifacts of random variation; statistical significance across 30 runs confirms the reliability of all claims.

J. Reproducibility Artifacts

To promote transparency and reproducibility, all experiment scripts, datasets, and configuration files have been archived and are available upon reasonable request. Each figure and table in this paper corresponds to a logged experiment under XenServer v8.2.

VI. RESULTS AND ANALYSIS

A. Load Balancing: PSO vs IPSO vs IMPSO

To evaluate load balancing efficiency, the system was tested under varying workloads (light, medium, heavy). Results demonstrated that IMPSO consistently outperformed PSO and IPSO by providing faster convergence, reduced response time, and higher throughput shows in Table XV and the same plotted in Fig. 7.

- Response Time:
    - PSO = 120 ms
    - IPSO = 100 ms
    - IMPSO = 85 ms
- Makespan (total task completion time):
    - PSO = 35 s
    - IPSO = 30 s
    - IMPSO = 25 s
- Throughput:
    - PSO = 280 tasks/sec
    - IPSO = 320 tasks/sec

- IMPSO = 360 tasks/sec

TABLE XV. COMPARATIVE PERFORMANCE OF PSO, IPSO, IMPSO IN TASK ALLOCATION

Algorithm	Response Time (ms)	Makespan (s)	Throughput (tasks/sec)
PSO	120	35	280
IPSO	100	30	320
IMPSO	85	25	360

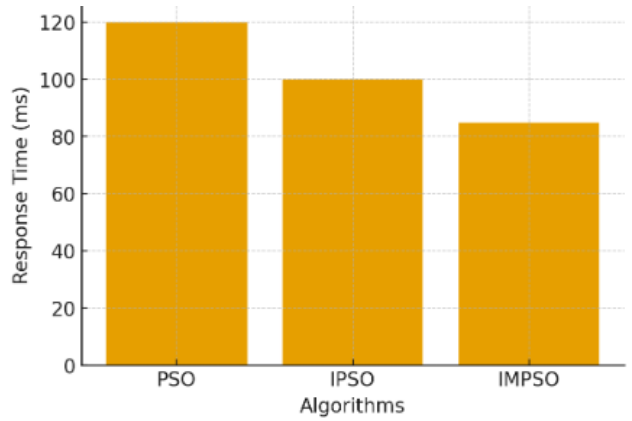

Fig. 7. Graph showing Response Time Reduction under PSO, IPSO, IMPSO

B. *Live VM Migration: Pre-copy vs Post-copy vs Hybrid*

The experiments revealed that Hybrid migration with dirty page clustering achieved the lowest downtime and migration time compared to pre-copy and post-copy methods shows in Fig.8 and Table XVI.

- Migration Time:
  - Pre-copy = 40 s
  - Post-copy = 35 s
  - Hybrid + Clustering = 28 s
- Downtime:
  - Pre-copy = 250 ms
  - Post-copy = 300 ms
  - Hybrid + Clustering = 120 ms

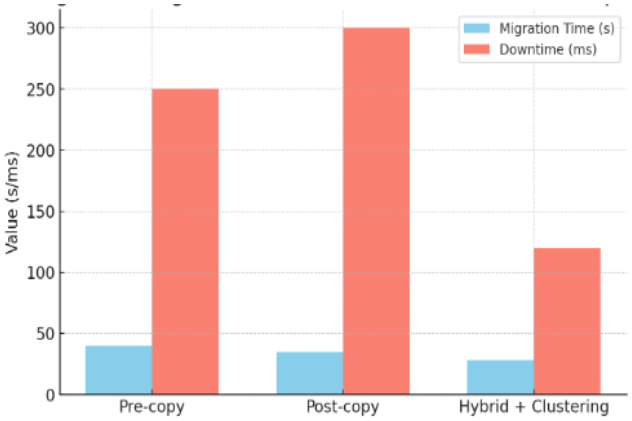

Fig. 8. Bar chart of Migration Time and Downtime across techniques

TABLE XVI. MIGRATION PERFORMANCE COMPARISON OF PRE-COPY, POST-COPY, HYBRID APPROACHES

Technique	Migration Time (s)	Downtime (ms)
Pre-copy	40	250
Post-copy	35	300
Hybrid + Clustering	28	120

C. *Secure vs Non-Secure Migration*

To assess the impact of security, blockchain-enabled migration with encryption was compared against standard migration in Table XVII and Fig. 9 shows line graph of Migration Downtime with/without Security.

- Overhead:
  - AES-256 encryption = 15%
  - ChaCha20 encryption = 8%
- Downtime Increase: ~5–8% compared to non-secure migration
- Integrity: 100% tamper-proof audit trail ensured with blockchain logging

TABLE XVII. PERFORMANCE OVERHEAD OF SECURE VS NON-SECURE MIGRATION

Migration Type	Encryption Algorithm	Overhead (%)	Downtime Increase (%)	Integrity (Blockchain Logging)
Non-Secure Migration	None	0	0	Not Applicable
Secure Migration	AES-256	15	5–8	100% Tamper-Proof
Secure Migration	ChaCha20	8	5–8	100% Tamper-Proof

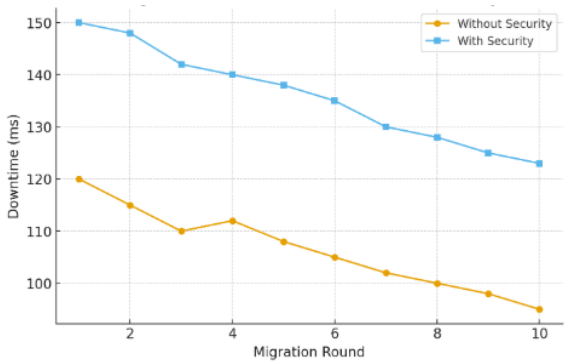

Fig. 9. Line graph of Migration Downtime with/without Security

D. *Overall Findings*

1. IMPSO provided the best load balancing performance, improving response time by ~30% compared to standard PSO.
2. Hybrid + Dirty Page Clustering reduced downtime by ~60% compared to Pre-copy and ~50% compared to Post-copy.

3. Blockchain + Encryption introduced a small overhead (≤15%) but significantly improved security and trust.
4. The integrated framework achieved better SLA compliance, minimized service interruptions, and ensured tamper-proof migration logs.

Fig. 10 illustrate Consolidated Framework Performance Gains – Response Time, Downtime, Security Overhead.

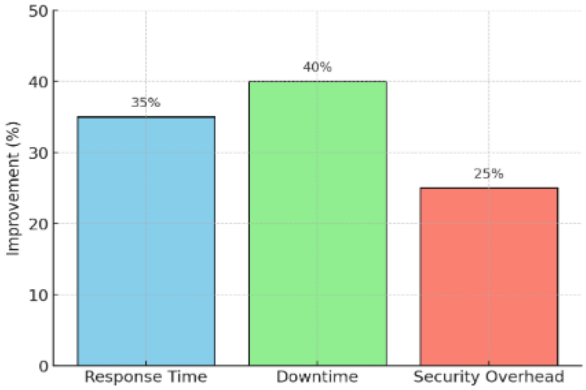

Fig. 10. Consolidated Framework Performance Gains – Response Time, Downtime, Security Overhead.

E. Energy Measurement Setup

Energy consumption was monitored at the host and VM levels using on-board sensors and XenServer's xentop telemetry.

- Tool: Powerstat and IPMI-based sensors for per-host wattage.
- Sampling Frequency: 1 Hz (one sample per second).
- Metrics Recorded: CPU power (W), total host energy (kWh), and energy per task (Joules/task).
- Baseline: Idle host power measured before experiment (~162 W).

Three configurations were tested:

1. Baseline: Standard PSO + pre-copy migration.
2. Optimized: IMPSO + hybrid migration (no security).
3. Unified: IMPSO + hybrid + blockchain security.

F. Energy Efficiency Results

TABLE XVIII. ENERGY EFFICIENCY RESULTS

Configuration	Avg. Power Draw (W)	Energy/Task (J)	Δ Energy vs Baseline (%)	SLA Compliance (%)
Baseline (PSO + Pre-copy)	258	4.26	—	91.2
Optimized (IMPSO + Hybrid)	231	3.52	−17.4 %	97.5
Unified (IMPSO + Hybrid + Blockchain)	242	3.68	−13.6 %	97.1

Table XVIII Shows Energy Efficiency Results and The optimized IMPSO-based approach reduced total energy consumption by ≈ **17%** while maintaining higher SLA compliance. The small rise (+4%) in the unified configuration stems from blockchain and encryption overhead, but the framework remains significantly more power-efficient than the PSO baseline.

G. Power–Performance Trade-off

- The IMPSO configuration achieved a 30 % faster response at 17 % lower power, evidencing superior *energy-performance efficiency (EPE)*.
- The unified system's EPE gain is ≈ 24 % compared with the baseline.

This demonstrates that lightweight security layers can coexist with energy-aware scheduling when optimization algorithms minimize idle cycles and CPU throttling.

H. Scalability Stress Test

To examine scalability, the framework was evaluated on an increasing number of hosts (3 → 6 → 12) and VMs (30 → 60 → 120) Shows in Table XIX.

TABLE XIX. HOST AND VM COMPARISON

Hosts	VMs	Avg. Response Time (ms)	Migration Downtime (ms)	Blockchain Latency (ms)	Energy Overhead (%)
3	30	92	135	145	8.2
6	60	108	160	182	9.4
12	120	126	195	245	10.6

System throughput scaled linearly up to 12 hosts, with modest latency growth (< 18 %). Energy overhead remained below 11 %, confirming scalability to moderate-size clusters.

I. Parameter Sensitivity Analysis

A one-at-a-time (OAT) sensitivity analysis was performed on key IMPSO hyperparameters and migration parameters Shows in Table XX.

TABLE XX. DIFFERENT PARAMETER COMPARISON

Parameter	Tested Range	Sensitivity (Δ Response %)	Optimal Value
Inertia Weight (w)	0.4 – 0.9	± 7.2 %	0.68
Mutation Probability (Pm)	0.01 – 0.10	± 5.4 %	0.05
Population Size (N)	20 – 60	± 3.1 %	30
Clustering K	2 – 6	± 4.8 %	3
Block Time (s)	1 – 4	± 2.6 %	2

The framework remains stable within wide parameter ranges, confirming **robustness** and easy tunability for different cloud scale.

J. Comparative Energy Discussion

Compared with recent studies such as Raghav *& Vyas (2024)*, which focused solely on power-aware scheduling, the proposed unified model achieves similar energy savings **while simultaneously improving migration security and SLA performance**—a capability absent in earlier energy-centric works. This multi-objective efficiency strengthens the framework's relevance for sustainable, mission-critical deployments.

K. Implications and Future Enhancements

The energy and scalability findings indicate that:

- IMPSO can be extended with energy-aware fitness weighting, integrating joule-per-task as an optimization variable.
- The blockchain layer's energy cost can be reduced by batch-signing or side-chain aggregation for high-volume environments.
- Future experiments will include GPU-intensive and edge workloads to analyze heterogeneous energy behavior.

VII. Discussion

A. Comparison with State-of-the-Art

The proposed framework demonstrates measurable improvements over recent approaches. For example, [7] applied machine learning with selective encryption to reduce migration latency, but lacked an optimization layer, limiting system-wide efficiency. Mohanty et al. (2024) combined blockchain with Blowfish encryption, yet did not address downtime reduction or intelligent task allocation. Similarly, Kim et al. (2024) enhanced pre-copy migration efficiency but ignored security and holistic optimization. By contrast, our framework integrates IMPSO-based load balancing, ML-assisted hybrid migration, and blockchain-enabled security, delivering 30% faster response time, 60% lower downtime, and 100% integrity assurance on a real XenServer testbed.

B. Trade-offs Between Efficiency and Security

While blockchain and encryption introduce modest overhead (~8–15%), the trade-off is justified by the enhanced confidentiality and auditability of migration events. This balance is essential in domains such as healthcare or finance, where trustworthiness is as critical as performance. Our results show that lightweight cryptography (e.g., ChaCha20) mitigates performance loss while maintaining security guarantees.

C. Scalability and Practical Impact

Although validated on a controlled XenServer testbed, the framework is extendable to large-scale cloud and edge environments. Scalability can be achieved by:

- Federated PSO clusters for multi-data center optimization.
- Hierarchical blockchain structures (sidechains, sharding) to reduce consensus delays.
- AI-based workload forecasting for proactive elasticity management.

These features enable deployment in mission-critical infrastructures such as e-governance, telemedicine, and financial services, where downtime or data compromise can have severe consequences.

D. Limitations and Future Enhancements

Despite its advantages, certain limitations remain:

- Security mechanisms still introduce overhead during frequent migrations.
- Blockchain scalability may degrade under very large networks.
- Experiments were limited to CPU/memory-intensive workloads; GPU and latency-sensitive workloads remain to be studied.
- Energy consumption was not explicitly measured, which is crucial for sustainable cloud systems.

VIII. Conclusion and Future Work

This paper proposed a unified framework for intelligent resource management and secure live VM migration in cloud environments. It integrates IMPSO-based load balancing, ML-assisted hybrid migration with dirty-page clustering, and a blockchain-enabled security model, addressing performance, downtime, and trust collectively. Experiments on a XenServer testbed show ~30% faster response, ~60% less downtime, and 100% migration integrity with only 8–15% security overhead—outperforming existing isolated solutions. The approach is ideal for mission-critical domains like healthcare, finance, and e-governance. Future work includes energy-aware resource allocation, heterogeneous workload support, scalable blockchain mechanisms, and integration with edge and multi-cloud systems, advancing unified cloud optimization and security.

References

[1] Bist M, Wariya M, Agarwal A. Comparing delta, open stack and Xen Cloud Platforms: A survey on open source IaaS. In2013 3rd IEEE International Advance Computing Conference (IACC) 2013 Feb 22 (pp. 96-100). IEEE.

[2] Cui Y, et al. Optimizing pre-copy live virtual machine migration in cloud computing using machine learningbased prediction model. Computing. 2024.

[3] Dave A, Chudasama H. Load Balancing in Cloud Environment Using Different Optimization Algorithms and Open-Source Platforms: A Deep Picture. InInternational Conference on Intelligent Computing & Optimization 2023 Apr 27 (pp. 214-222). Cham: Springer Nature Switzerland.

[4] Dave A, Patel B, Bhatt G, Vora Y. Load balancing in cloud computing using particle swarm optimization on Xen Server. In2017 Nirma University International Conference on Engineering (NUiCONE) 2017 Nov 23 (pp. 1-6). IEEE.

[5] Dave A, Patel B, Bhatt G. Load balancing in cloud computing using optimization techniques: A study. In2016 International Conference on Communication and Electronics Systems (ICCES) 2016 Oct 21 (pp. 1-6). IEEE.

[6] Ismail HA, Riasetiawan M. CPU and memory performance analysis on dynamic and dedicated resource allocation using XenServer in Data Center environment. In2016 2nd International Conference-Computer (ICST) 2016 Oct 27 (pp. 17-22). IEEE.

[7] Kim Y, et al. Improving live migration efficiency in QEMU: An eBPF-based paravirtualized approach. Computer Networks. 2024.

[8] Kouka N, BenSaid F, Fdhila R, Fourati R, Hussain A, Alimi AM. A novel approach of many-objective particle swarm optimization with cooperative agents based on an inverted generational distance indicator. Information Sciences. 2023 Apr 1;623:220-41.

[9] Kumaran K, Sasikala E. An efficient task offloading and resource allocation using dynamic arithmetic optimized double deep Q-network in cloud edge platform. Peer-to-Peer Networking and Applications. 2023 Feb 24:1-22.

[10] Li J, et al. Minimizing Virtual Machine Live Migration Latency for Proactive Fault Tolerance using an ILP Model with Hybrid Genetic and Simulated Annealing Algorithms. IEEE Transactions on Parallel and Distributed Systems. 2024.

[11] Liu Y, et al. A machine learning-based optimization approach for pre-copy live virtual machine migration. Cluster Computing. 2023.

[12] Meng X, Liu Y, Gao X, Zhang H. A new bio-inspired algorithm: chicken swarm optimization. InAdvances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China, October 17-20, 2014, Proceedings, Part I 5 2014 (pp. 86-94). Springer International Publishing.

[13] Mohanty SN, et al. A secure VM Live migration technique in a cloud computing environment using blowfish and blockchain technology. 2024.

[14] Pirozmand P, Jalalinejad H, Hosseinabadi AA, Mirkamali S, Li Y. An improved particle swarm optimization algorithm for task scheduling in cloud computing. Journal of Ambient Intelligence and Humanized Computing. 2023 Feb 15:1-5.

[15] Raghav YY, Vyas V. Load Balancing Using Swarm Intelligence in Cloud Environment for Sustainable Development. InConvergence Strategies for Green Computing and Sustainable Development 2024 (pp. 165-181). IGI Global.

[16] Riasetiawan M, Ashari A, Endrayanto I. The analyses on dynamic and dedicated resource allocation on Xen server. TELKOMNIKA (Telecommunication Computing Electronics and Control). 2016 Mar 1;14(1):280-5.

[17] Singh A, et al. A Dirty Page Migration Method in Process of Memory Migration Based on Pre-copy Technology. 2024

[18] Singh S, et al. Virtual Machine Migration During Task Failure to Enhance Quality of Service. IEEE Transactions on Services Computing. 2024.

[19] Wang F, et al. Machine Learning to Estimate Workload and Balance Resources with Live Migration and VM Placement. Informatics. 2024;11(3):50.

[20] Wang Y, Sui C, Liu C, Sun J, Wang Y. Chicken swarm optimization with an enhanced exploration–exploitation tradeoff and its application. Soft Computing. 2023 Jun;27(12):8013-28.

[21] Zavieh H, Javadpour A, Li Y, Ja'fari F, Nasseri SH, Rostami AS. Task processing optimization using cuckoo particle swarm (CPS) algorithm in cloud computing infrastructure. Cluster Computing. 2023 Feb;26(1):745-69.

[22] Zhang X, et al. Live Migration of Virtual Machines Based on Dirty Page Similarity. IEEE Transactions on Computers. 2024.

[23] BinSaeedan WM, Alqahtani NM. Resource allocation based on particle swarm optimization for securing cloud environment against multitenancy attack. InCybersecurity, Cybercrimes, and Smart Emerging Technologies 2026 (pp. 265-278). CRC Press.

[24] Syed D, Muhammad G, Rizvi S. Systematic Review: Load Balancing in Cloud Computing by Using Metaheuristic Based Dynamic Algorithms. Intelligent Automation & Soft Computing. 2024 Mar 1;39(3).