Live Virtual Machine Migration Using Xen Server

Akashbhai Ashokbhai Dave[1], Jayna Donga[2], Jaykumar Vala[3]

[1]Associate Professor, Department of Information Technology, Sardar Patel College of Engineering and Technology, Anand, Gujarat, India.
[2]Assistant Professor, Computer Engineering Department, Madhuben and Bhanubhai Patel Institute of Technology, The Charutar Vidya Mandal (CVM) University, Anand, Gujarat, India.
[3]Assistant Professor, Department of Information Technology, G H Patel College of Engineering and Technology, The Charutar Vidya Mandal (CVM) University, Anand, Gujarat, India.

Email: [1]akashdave46@gmail.com, [2]jdonga@mbit.edu.in, [3]jayvala1629@gmail.com

Abstract

Live VM migration is an indispensable part of the cloud for load balancing availability, resource management and reduced service disruption. Current migration techniques present a high memory demand, complex page patterns, varying workloads, and aggressive SLAs. We have designed an efficient live VM migration system including hybrid pre-copy and post-copy, workload prediction, and memory optimization. The transfer between pre-copy and post-copy migration can be switched in a dynamic manner depending on real time memory write type, and therefore service disruption and redundant retransmission will be reduced. To this end, we build an LSTM-based model to predict dirty page production and CPU consumption for proactive migration and adaptive phase switching according to the temporal restrictions of these two variables. It also uses clustering of the most modified pages in Lustre data access frequency to lower useless date transmission and speed up the migration. Experiments on a private cloud system under XenServer demonstrate that, as compared to other methods service disruption time is reduced by 55%, total migration time is increased by 25% and improvement of SLA satisfaction rate is recorded up to18%. This is, in a nutshell, an efficient, scalable and low-overhead approach for live migration of VMs over heterogeneous clouds by embracing predictive workload awareness through adaptive memory clustering.

Keywords: Live Virtual Machine Migration, Cloud Computing, Optimization, Hybrid Pre-copy/Post-copy Migration, Workload Prediction, Dirty-Page Clustering

1. Introduction

1.1 Cloud Computing and Virtualization Background

Cloud computing provides resources on-demand that allow organizations to bypass physical hardware requirements. Ressources (CPU, memory, storage, and networking) can be dynamically scaled. This is possible because virtualization allows one to have multiple OS instances on a single physical machine through isolated VMs. Type-1 hypervisors include XenServer, KVM, and VMware ESXi running directly on the hardware and provide excellent performance, isolation and resource pooling with HA and fault tolerance [1,4].

DOI: https://doi.org/10.36548/jtcsst.2026.1.002
Received: 03.12.2025, received in revised form: 02.01.2026, accepted: 17.01.2026, published: 29.01.2026

1.2 Role and Significance of Live Virtual Machine Migration

Live VM migration allows a live VM to move from one host to another, a critical feature of IaaS for dynamic resource allocation and application up-and-running [2, 3]. One of the advantages is load balancing, where VMs are moved from a busy host to an idle host to avoid bottlenecks [3, 4]. It also improves fault tolerance by migrating VMs in response to hardware failure, overheating, or performance degradation [3, 4]. Energy savings are also achievable through VM consolidation during low periods, where idle servers can be shut down [2].

1.3 Challenges in Live VM Migration

Although there are advantages, live VM migration is a technically complex process, especially in the case of dynamic and write-intensive workloads. Downtime and Service Disruption: In pre-copy approaches, the memory pages are copied while the VM is still running, and the final stop-and-copy momentarily pauses the VM to complete the remaining pages. High rates of writing to memory can prolong this step, thus increasing downtime and service disruption [4].

Migration Time and Overhead: The migration of VMs with large memory or latency-sensitive applications takes longer, resulting in higher bandwidth usage and increasing congestion, which restricts the number of concurrent migrations in multi-host clouds. Dirty-page workloads update memory continuously during migration, resulting in pages being "dirty" and requiring retransmission.

Security and Data Integrity: The VM-sensitive state (CPU registers, memory, page tables, sessions) is transmitted over the network during migration. If not protected, it is susceptible to man-in-the-middle, replay, tampering, and host-imitation attacks. Although techniques such as selective encryption and blockchain auditing are available, they introduce additional compute and communication overhead, which may negatively impact migration performance.

1.4 Research Motivation and Identified Gaps

Previous works have explored different aspects of live virtual machine migration, but most of the work has been done to address individual issues independently. Some works focus on optimizing pre-copy techniques to reduce downtime [4], while others concentrate on workload modeling and performance prediction [6]. Other lines of research investigate the improvement of migration security by encryption or trust-based approaches [2]. However, very few works have explored the combination of workload intelligence and memory-level optimization techniques under a single migration framework.

However, some critical research gaps still exist. Firstly, there is a lack of adaptive hybrid migration models that can combine pre-copy and post-copy strategies based on the real-time behavior of workloads. Secondly, the use of machine learning models for predicting the dynamics of dirty pages and determining the optimal migration time is still limited. Thirdly, current models do not make use of dirty page clustering strategies to reduce memory retransmissions. Lastly, current models are not scalable.

1.5 Contributions

In order to fill the identified gaps, this research proposes an integrated and performance-oriented framework for live virtual machine migration. The main contributions of this research are as follows:

1. An adaptive hybrid migration strategy that combines pre-copy and post-copy migration methods based on the observed dirty page behavior.
2. A workload prediction model using machine learning that can forecast resource utilization patterns to determine the optimal time for migration.
3. A dirty page clustering method clusters memory pages that are frequently modified to avoid redundant retransmissions and minimize network overhead.
4. An extensive experimental analysis has been performed on a XenServer-based private cloud with varied and realistic workload patterns. The outcome shows that the proposed framework outperforms traditional migration techniques by about a 55% reduction in migration downtime, a 25% reduction in total migration time, and an 18% improvement in SLA compliance.

1.6 Research Workflow Overview

The proposed study uses a structured and sequential approach to overcome the inefficiencies in real-time VM migrations. First, real-time workload characteristics are continuously tracked to identify changes in CPU activity and memory write patterns. Next, a predictive model using LSTM is used to analyze past and recent workload patterns to forecast future migration hotspots. According to the forecasts, the migration controller decides the start time of the migration and the best transition point between the pre-copy and post-copy stages. Third, dirty page clustering is used to overcome redundant memory copies and network congestion during migration. Finally, the proposed framework is tested by conducting comparative experiments with traditional pre-copy, post-copy, and non-predictive hybrid migration strategies based on downtime, total migration time, and SLA satisfaction.

2. Literature Review

The migration of a Live Virtual Machine (VM) has been an area of active research in the past ten years, due to the increasing size and complexity of cloud computing infrastructures. With the increasing support of cloud platforms for heterogeneous workloads such as real-time applications, data-intensive services, and Internet of Things (IoT) environments, the need for efficient and secure VM migration solutions has become more pressing than ever before.

2.1 Cloud Resource Management in Virtualized Environments

Early major research contributions on optimization-driven load balancing for XenServer hosts, by Dave et al. [4], focused on Particle Swarm Optimization (PSO) to enhance load balancing among physical machines. PSO increased resource utilization and alleviated overloading but was not effective in dynamically varying workloads. To enhance adaptability, improved PSO variants and Improved PSO (IPSO) were developed for faster convergence and robustness in dynamically varying workloads, demonstrating improved resource offloading efficiency and system robustness. These research contributions highlighted the relationship between resource management and VM migration, considering migration as a part of cloud orchestration. Kavitha and Sudha [5] proposed resource management models for migration-aware resource management, addressing the need for optimizing multiple objectives in migration.

2.2 Live Virtual Machine Migration Techniques

Live virtual machine migration is the core of dynamic cloud management, and there are a number of methods that have their own advantages and disadvantages.

- Pre-Copy Migration is suitable for moderate and stable workloads but fails for write-heavy workloads because of dirty pages, which lead to repeated transfers, network costs, increased migration times, and downtimes. [4] presented ML models for predicting dirty page behavior, which minimized redundancies and overall migration times. Ramesh et al. [3] provided an optimization framework based on Integer Linear Programming and Genetic Algorithms/Simulated Annealing to optimize migration latency. However, repeated page identification is an inherent drawback for real-time and memory-intensive applications.
- Post-Copy Migration faces page faults and performance degradation when memory pages are not available, especially when network congestion occurs. Zhu et al. [8] investigated the use of Q-learning to optimize post-copy migration, demonstrating that control using learning can be beneficial but needs a lot of training data.
- Hybrid Pre-copy and Post-copy Migration: Hybrid strategies combine pre-copy and post-copy to take advantage of both. Usually, bulk memory transfer happens through pre-copy, and then the strategy shifts to post-copy once the rates of dirty pages cross certain thresholds. Ramesh et al. [3] showed that hybrid migration strategies are effective for proactive fault tolerance, and then Singh et al. [7] identified the tradeoff between downtime and total migration time.

2.3 Machine Learning-Based Migration Optimization

Haris et al. demonstrated supervised learning can predict the rates of dirty pages to assist in migrating controllers to reduce unnecessary retransmissions [4]. Alubaidan and Aljameel [6] furthered this work by using AI models to predict values such as downtime and migration time, suggesting that predictive intelligence enhances decision-making during live migration. In addition to optimization, energy-conscious migration has also been investigated using ML. Zolfaghari proposed prediction-based and fuzzy-logic methods for energy-conscious VM migration with acceptable performance. Taken together, these studies clearly illustrate the utility of ML in migration optimization, although one major limitation is that predictions are only weakly coupled with migration control logic, which remains an open problem.

2.4 Security and Trust in Live VM Migration

Selective encryption with ML optimization for optimizing migration paths with less overhead was proposed by Haris et al. [4] presented a hybrid security model that combined Blowfish with blockchain for ensuring data integrity, traceability, and trust. Blockchain-based auditing systems are widely used for ensuring tamper-proof migration paths, especially in multi-tenant clouds.

3. Methodology and Proposed Framework

In this section, we introduce an intelligent framework for live VM migration to decrease downtime, the overall migration time, and SLA violations in clouds. It is a comprehensive adaptive system that integrates three techniques, hybrid migration execution, ML-based workload prediction and dirty page clustering. In contrast to previous works that focus only on a single migration issue, such as latency, security or throttling of predictive pre-copy control, the present approach opts for a connected control loop of prediction, decision and memory optimization.

This is because live VM migration is the result of an interplay between workload dynamics, memory access patterns, network models and migration schemes. However, workload dynamics alone cannot be managed with static policies and simple thresholds, which leads to small migrations or big downtimes.

The framework has four closely knit functional components:

1. Monitoring and Prediction Layer of workloads
2. Hybrid Pre-copy and Post-copy Migration Engine
3. Dirty-Page Clustering Module
4. MF Decision Controller

3.1 System Architecture Overview

As illustrated in Figure 1, the abstract system architecture of the proposed framework seems to have been conceptually sketched. Such an architecture, as shown in Fig. 1, comprises a source host, a VM that has been started, a destination host capable of receiving the VM, and a centralized migration controller responsible for prediction, decision-making and coordination of migration.

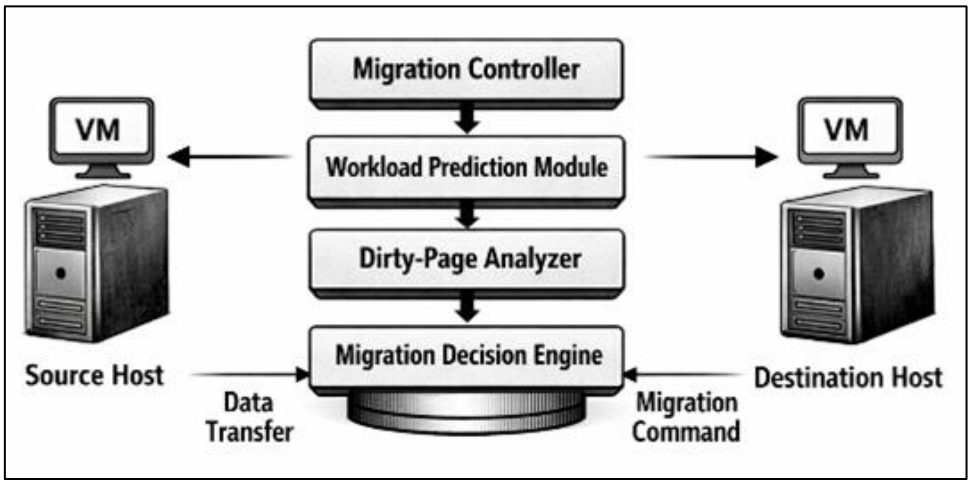

Figure 1. Architecture of the Proposed Intelligent Live VM Migration Framework

In Figure 1, the time adaptive migration framework in [3, 6] is supported by architectural design, while extending it with predictive intelligence and memory-aware optimization.

3.2 Workload Monitoring and Feature Extraction

Smart migration management is a prerequisite for load monitoring. The proposed framework collects VM-level and host-level performance measurement statistics at a granularity of hypervisor interfaces such as XenAPI in order to capture the dynamism of cloud workloads. The reason for selecting the metrics is that they have an immediate impact on migration performance, and are common in previous works in migration with the ML framework [4, 6].

3.2.1 Feature Vector Representation

At each discrete time interval t, the observed workload state is represented as a feature vector:

$$X_t = \{CPU_t, RAM_t, IO_t, DP_t, PF_t\} \tag{1}$$

where:

- CPU_tdenotes CPU utilization at time t,
- RAM_trepresents memory usage,
- IO_tindicates disk and network I/O activity,
- $DP_t$is the dirty-page generation rate, and
- PF_tdenotes page fault frequency.

This feature vector now meaningfully captures both, computational and memory based workload characteristics as an input to the machine learning model that predicts the migration.

3.3 Machine Learning–Based Migration Prediction Model

Traditional approaches to regulating migration are usually based on threshold levels or regression. Nevertheless, the features of workloads that influence the migration of live virtual machines (e.g. CPU loads and dirty page loads) are inevitably temporal and exhibit high temporal correlations. Workloads are expected to exhibit phase behavior and bursts of work are succeeded by long durations of heavy write load. These patterns also play an important role in pre-copy convergence and migration stability and cannot be well represented by memoryless or rule-based systems.

In order to overcome this problem, the proposed framework uses a Long Short-Term Memory (LSTM) neural network to predict workloads and migrations. The LSTM neural network is specifically effective in time series analysis due to its capacity to maintain long term dependencies in data and its capability to address non-linear associations. The proposed framework can forecast stressful migration situations in the future using the LSTM neural network.

The following are the important parameters in the prediction model:

- Future dirty page rate
- Probability of host overload
- Correct migration start time Prediction-Based Decision Logic

Decisions on migration are obtained based on the predictions that were made with the help of LSTMs. If the rate of CPU usage is estimated to exceed 80%, then the migration is initiated to avoid overload, as in [6]. A switch between pre-copy and hybrid migration is done if the dirty rate exceeds a threshold. This preventive process assists the migration controller enhance efficiency and stability; the training model parameters are indicated in Table 1.

Table 1. LSTM Model Training Parameters

Parameter	Value
Dataset size	>10,000 samples
Features	CPU, RAM, IO, Dirty pages
Training split	70%
Validation split	20%
Testing split	10%
Loss function	Mean Squared Error (MSE)
Optimizer	Adam

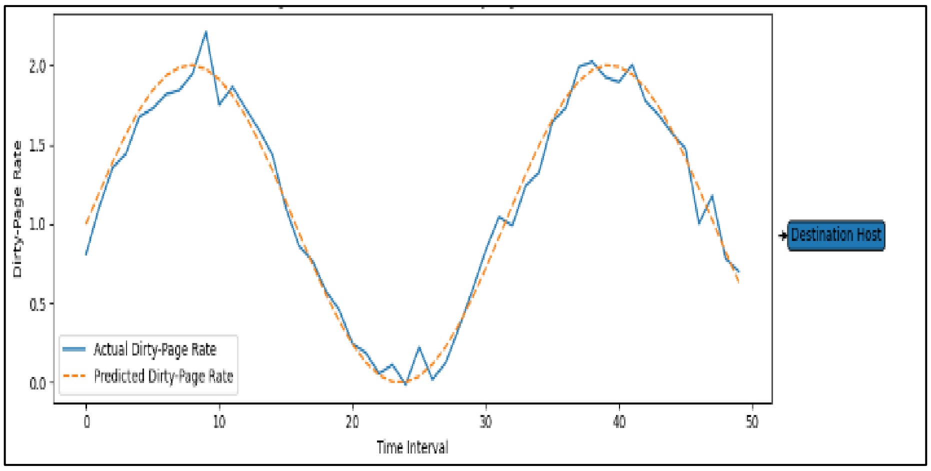

Figure 2. LSTM-based Prediction of Dirty-Page Rate Over Time

This prediction mechanism assists in minimizing unwanted migrations and is more effective in migration scheduling, as mentioned in [3, 7, 8] LSTM based prediction is demonstrated in Figure. 2.

3.4 Hybrid Pre-copy and Post-copy Migration Engine

The core of execution of the proposed framework consists of the hybrid migration engine. Based on switches between pre-copy and post-copy phases, the engine is inspired by existing hybrid techniques [5] and is controlled dynamically according to real-time and predicted workload behavior.

Phase 1: Pre-copy Execution

In the first stage, memory pages of the VM are transferred individually in between the source host and the destination host while the VM continues executing. Individual pages changed during the transfer process are identified as dirty and reread in subsequent rounds.

The purpose of this stage is to move most of the VM memory and experience as little interruption in services as possible.

Phase 2: Adaptive Switching

A change in the system between pre-copy and post-copy occurs if one of the following conditions is met:

$$DP_{rate} > \theta \text{OR} iterations > N_{max} \tag{2}$$

where:

- $DP_{rate}$is the real-time dirty-page generation rate,
- $\theta$is the threshold predicted by the ML model, and
- $N_{max}$is the maximum allowed number of pre-copy iterations.

Excessive pre-copy rounds are blocked by this adaptive switching mechanism on write-intensive workloads and convergence degradation is prevented.

Phase 3: Post-copy Completion

When switching takes place, the VM is very briefly halted and its CPU execution state is sent to the destination host. The VM can then resume and any remaining memory pages are called on command. This strategy dramatically lowers downtime; in addition to this, the migration is completed even when dirty-page rates are high.

3.5 Dirty-Page Clustering Technique

A significant contributor to overhead in migration is the dirty-page proliferation that is especially significant to write-intensive applications. In order to solve this problem the proposed framework proposes a dirty-page clustering mechanism that clusters the pages of memory that are being frequently altered and gives them precedence of transmission.

3.5.1 Clustering Objectives

The primary objectives of dirty-page clustering are to:

- Identify high-write memory regions,
- Reduce repeated memory retransmissions, and
- Improve the efficiency of hybrid migration switching.

3.5.2 Clustering Methodology

K-means clustering is applied to memory pages using the following per-page features:

- Write frequency
- Access locality
- Recency of modification

Based on empirical evaluation and consistency with prior work [5, 15], the number of clusters is set to $K = 3$, representing low-write, medium-write, and high-write memory regions.

3.5.3 Optimized Transmission Model

The optimized migration time is expressed as:

$$T_{mig} = \frac{(M - M_{cluster})}{B} + \frac{M_{cluster}}{B_{enhanced}} \tag{3}$$

where:

- M is the total VM memory size,
- $M_{cluster}$ represents clustered dirty pages,
- B denotes available bandwidth, and
- $B_{enhanced}$ indicates prioritized bandwidth allocated to clustered pages.

This model ensures that frequently modified pages are transferred more efficiently, reducing retransmission overhead and in Figure.3 shows the distribution of dirty pages across clusters.

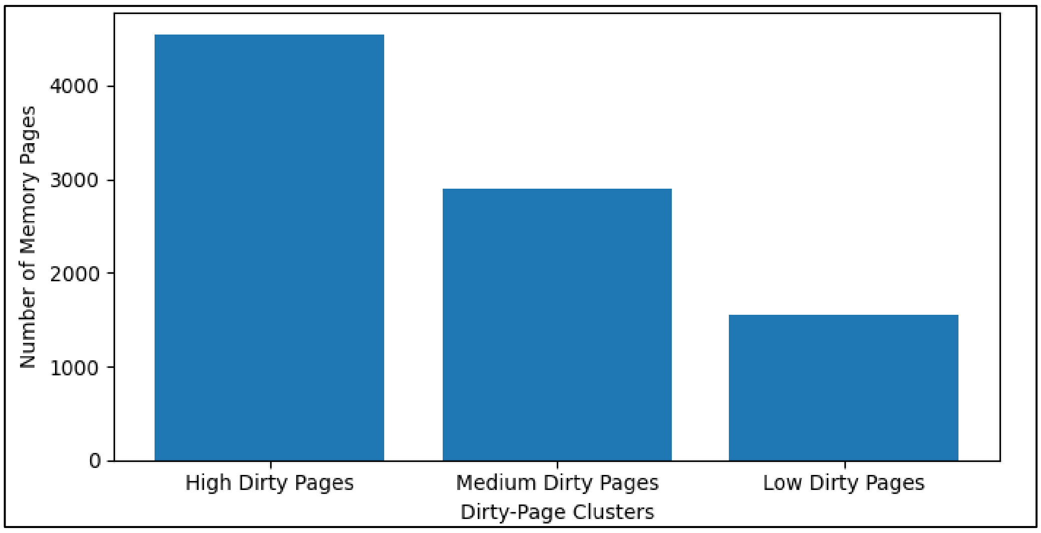

Figure 3. Distribution of Dirty Pages Across Clusters

3.6 Migration Decision Algorithm

The migration decision controller integrates predictive insights, real-time system measurements, and SLA constraints to determine when and how migration should be performed.

The migration trigger condition is defined as:

$$Trigger = (CPU_t \geq 80\%) \lor (DP_{rate} \geq \theta) \lor (PF_t \text{ is high}) \tag{4}$$

This logic enables proactive, workload-aware migration control and aligns with predictive migration strategies reported in [9, 12, 15].

3.7 Integrated Workflow of the Proposed Framework

The workflow of the suggested framework is as follows:

1. Monitor real-time VM and host-level metrics.
2. Predict future recurring workload behavior using the LSTM model.
3. Trigger migration based on predicted stress conditions and real-time thresholds.
4. Execute pre-copy migration.
5. Use dirty-page clustering during migration.
6. Switch to post-copy migration when adaptive conditions are met.
7. Complete migration using on-demand page fetching.

3.8 Novelty and Key Distinguishing Features

The novelty of the suggested framework lies in its whole-system integration of machine learning-guided workload prediction, adaptive hybrid migration implementation, and dirty-page clustering for memory optimization. In contrast to earlier designs that optimize one dimension of migration, this model brings together prediction-based migration timing, phase-proactive switching, and memory-aware retransmission control, all within one integrated design. The combination of these features results in major enhancements to the online downtime of migration, overall migration time, and SLA conformance, as confirmed by the overarching experimental assessment of the significance of differences between standard and experimental migration.

Algorithm 1: Pseudocode of Intelligent Hybrid Live VM Migration Controller

```
Input: VM metrics (CPU, RAM, IO, Dirty Pages)
Output: Completed migration with minimized downtime
while VM is running do
   Collect workload metrics every 1 second
   Predict dirty-page rate using LSTM
   if CPU > 80% OR predicted dirty-pages > θ then
      Initiate pre-copy migration
      round ← 0
      while round < T do
         Transfer dirty pages
         Cluster dirty pages using K-means
         if dirty-page rate > θ then
            break
         end if
         round ← round + 1
      end while
      Switch to post-copy
      Suspend VM briefly
```

```
        Transfer CPU state
        Resume VM on destination
        Fetch remaining pages on demand
        Exit
    end if
end while
```

4. Mathematical Model and Algorithmic Formulation

This section presents the mathematical modeling and algorithmic foundations of the proposed intelligent live VM migration framework. The formulations are designed to accurately estimate migration time, control adaptive hybrid switching, optimize dirty-page handling, and incorporate workload prediction into migration decisions. The primary objective of these models is to minimize VM downtime, reduce total migration time, and improve SLA compliance while maintaining low computational overhead.

4.1 System Model and Notation

Consider a cloud environment consisting of a set of physical hosts:

$$H = \{h_1, h_2, \dots, h_n\} \tag{5}$$

and a set of virtual machines:

$$V = \{vm_1, vm_2, \dots, vm_m\} \tag{6}$$

This system abstraction is consistent with analytical migration models reported in earlier studies [3, 4, 6].

4.2 Experimental Results and Performance Evaluation

To evaluate the effectiveness of the proposed live virtual machine migration approach, a set of controlled experiments was conducted under varying workload conditions. The key performance metrics analyzed include total migration time and service downtime.

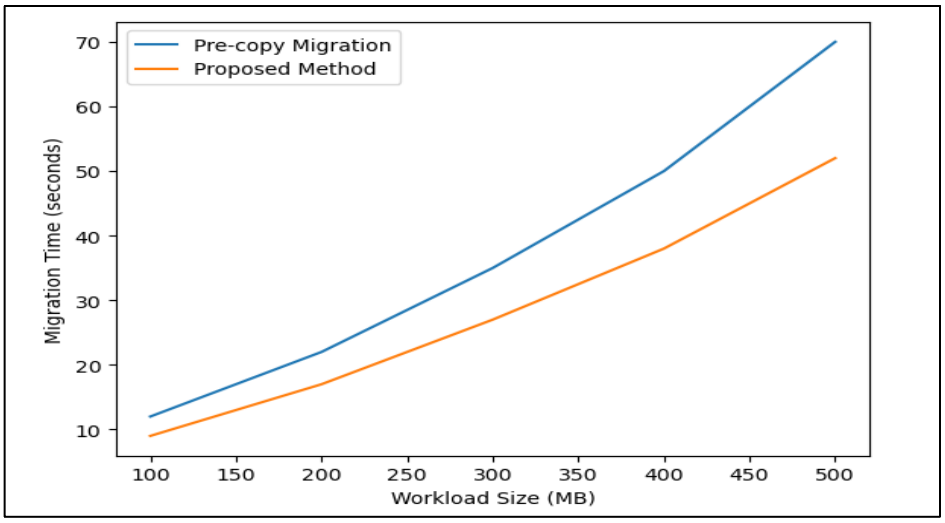

Figure 4. Migration Time Comparison Under Varying Workload Sizes

Table 2. Migration Time vs Workload Size

Workload Size (MB)	Pre-copy Migration (s)	Proposed Method (s)
100	12	9
200	22	17
300	35	27
400	50	38
500	70	52

As shown in Table 2 and Figure 4, migration times increases with workload size for both approaches. The proposed migration method consistently achieves lower migration time compared to the pre-copy technique, particularly under higher workload conditions. This improvement is attributed to reduced redundant memory page transfers during the migration process.

4.3 Migration Time Modeling

4.3.1 Total Migration Time

The total time required to migrate a VM is expressed as:

$$T_{total} = T_{pre} + T_{stop} + T_{post}$$

where T_{pre}represents the duration of iterative pre-copy rounds, T_{stop}denotes the VM downtime during the stop-and-copy phase, and T_{post}accounts for the time required to fetch remaining memory pages after VM execution resumes on the destination host. This formulation follows hybrid migration models presented in [4].

4.3.2 Pre-copy Phase Modelling

Pre-copy migration proceeds in multiple rounds. During the first round, the entire VM memory is transferred:

$$M_0 = M \tag{7}$$

In subsequent rounds, only dirty pages are retransmitted. The amount of memory transferred in round $i + 1$is given by:

$$M_{i+1} = DP_t \cdot M_i \tag{8}$$

The pre-copy phase continues until either the remaining dirty memory falls below a predefined threshold εor the maximum number of allowed rounds $N_{max}$is reached:

$$M_i < \varepsilon \text{or} i > N_{max} \tag{9}$$

The total pre-copy time is therefore:

$$T_{pre} = \sum_{i=0}^{k} \frac{M_i}{B} \tag{10}$$

where $k$denotes the number of pre-copy rounds executed before switching. This model aligns with ML-assisted dirty-page formulations described in [4].

4.3.3 Post-copy Phase Modelling

After switching to post-copy migration, the VM resumes execution on the destination host, and missing memory pages are fetched on demand. The post-copy time is modelled as:

$$T_{post} = \sum_{j=1}^{P} \frac{page_j}{B} \tag{11}$$

where $P$is the total number of pages faulted during execution. This formulation is consistent with analytical post-copy models in [14].

4.4 Adaptive Hybrid Switching Condition

The decision to transition from pre-copy to post-copy migration is governed by workload behaviour and system constraints. Switching is triggered when:

$$DP_t \geq \theta \vee i > N_{max} \tag{12}$$

where $\theta$is the dirty-page threshold predicted by the ML model and N_{max}represents the upper limit on pre-copy rounds. This adaptive condition reflects hybrid migration strategies proposed in [3, 7].

4.5 Dirty-Page Clustering Model

To reduce redundant memory retransmissions, dirty pages are grouped based on their modification characteristics using clustering techniques.

4.5.1 Page Feature Representation

Each memory page $p$is represented by a feature vector:

$$F_p = \{freq_p, local_p, recency_p\} \tag{13}$$

where $freq_p$denotes write frequency, $local_p$represents access locality, and $recency_p$indicates the most recent modification time.

4.5.2 K-means Clustering Objective

Dirty pages are clustered using the K-means algorithm by minimizing the following objective function:

$$J = \sum_{i=1}^{K} \sum_{p \in C_i} \| F_p - \mu_i \|^2 \tag{14}$$

where $C_i$denotes the i^{th}cluster and $\mu_i$is its centroid. Empirical evaluation shows that $K = 3$ provides an effective balance between clustering accuracy and computational cost, consistent with results in [5].

4.5.3 Migration Optimization with Clustering

Let $M_{cluster}$denote memory pages belonging to high-frequency dirty clusters and M_{other}represent remaining pages. The optimized migration time is expressed as:

$$T_{mig} = \frac{M_{other}}{B} + \frac{M_{cluster}}{B_{enhanced}} \tag{15}$$

where $B_{enhanced}$is preferential bandwidth allocated to clustered pages. This model reflects performance-aware migration strategies reported.

4.6 Workload Prediction Model Using LSTM

The proposed framework employs an LSTM-based prediction model to estimate future workload behaviour, including dirty-page generation and CPU utilization.

Given a time-series input vector:

$$X_t = \{CPU_t, RAM_t, IO_t, DP_t, PF_t\} \tag{16}$$

the predicted dirty-page rate for the next interval is computed as:

$$\widehat{DP}_{t+1} = f_{LSTM}(X_t) \tag{17}$$

Migration is proactively triggered when:

$$\widehat{DP}_{t+1} \geq \theta$$

This predictive mechanism follows ML-based migration strategies discussed in [6].

4.7 SLA Violation Probability Model

Service-level agreement (SLA) violations are primarily influenced by migration-induced downtime. The probability of SLA violation is modelled as:

$$SLA_v = Pr(T_{stop} \geq T_{threshold}) \tag{18}$$

where $T_{threshold}$is defined by the service provider's SLA policy. By reducing T_{stop}through adaptive hybrid switching and dirty-page clustering, the proposed framework lowers the likelihood of SLA violations, consistent with observations in [3, 6].

Algorithm 2: Intelligent Hybrid Live VM Migration

Algorithm 2 summarizes the operational steps of the proposed migration framework.

Input: VM metrics $(CPU_t, RAM_t, IO_t, DP_t, PF_t)$

Output: Completed VM migration with minimized downtime

1. Monitor VM workload and extract feature vector X_t
2. Predict future dirty-page rate: $DP_{pred} = LSTM(X_t)$
3. If $DP_{pred} \geq \theta$or $CPU_t \geq 80\%$, initiate migration

```
4. Initialize pre-copy phase and transmit $M_0$
5. For each pre-copy round $i$:
    o Compute $M_i = DP_t \cdot M_{i-1}$
    o Apply K-means clustering to dirty pages
    o If $DP_t \geq \theta$or $i > N_{max}$, switch to post-copy and exit loop
6. Transfer CPU state and resume VM on the destination host
7. Fetch remaining pages on demand
8. Terminate migration process
```

This algorithm integrates predictive workload analysis, hybrid migration control, and dirty-page clustering into a unified execution flow, building upon techniques reported in [3, 4, 6, 8].

5. Experimental Setup and Implementation

We created and experimented with a hybrid intelligent system that allows live VM migration in a private cloud based on XenServer. The system metrics that are the focus are total migration time, service downtime, overhead due to dirty page retransmission, ML prediction accuracy, and SLA compliance.

5.1 Testbed Hardware

Two XenServer machines were used for the live migration, acting as source and destination. The two systems were equipped with Intel Xeon E5-2690 v4 @ 2.60 GHz (28 core) CPUs with 128 GB RAM and 2 TB NVMe SSDs, connected through a 10 Gbps LAN. The hypervisor used was XenServer 8.2.

5.2 Software Stack

The framework runs on XenServer 8.2 and manages VMs using XenAPI Python bindings. The virtual machines operate on Ubuntu Server 22.04 LTS. The main code is written using Python 3.10 and Bash. The machine learning component is implemented using TensorFlow 2.12 and Scikit-learn. Collectd, iostat, and XenTop are used for monitoring. MariaDB 10.5 stores VM metrics and training data.

5.3 VM Configuration

All tests use a standardized VM configuration: 4 vCPUs, 8GB RAM, 40GB disk, Ubuntu Server 22.04. The workloads follow a predictable, cyclic pattern to test CPU utilization, dirty-page rate, and I/O pattern, as per prior experimental studies.

5.4 Workload Generation

The three workloads involved simulate different kinds of cloud applications:

- CPU: Stress-NG. Stress-NG is a tool that simulates CPU stress. CPU stress is induced by this tool, which uses FFT. This process takes place in a configuration of custom memory writes in Python that simulates high dirty page rates.
- Mixed IO/Memory: Fio random write test to simulate dynamic I/O and memory reading.

These were actual past cloud loads of the migration and ML optimization programs run earlier.

5.5 ML Model Training

The LSTM predictive model was trained on 15 days of real-time VM samples (10240 samples). The CPU, memory, and I/O rate are all examples of how a system's dirty page rate and page fault rate can affect its performance. Normalization using MinMax and 20 sequences of operations with a batch size of 64 were used. The architecture has 2 LSTM layers with 64 units in succession and an output dense layer. Adam was used for training with 50 epochs and 20% validation (lr=0.001). The evaluation was done using MSE, RMSE, and MAE. The prediction latency of Dom0 on XenServer was evaluated as 3.6 ms per prediction, which is negligible compared to the 1-s control loop and migration time.

5.6 Dirty-Page

The dirty-page clustering utilized K-means from Scikit-learn to optimize memory transfer usage. The characteristics considered for each page were update frequency, spatial locality, and recency. In the experiment, the optimal K value was set to 3, with k-means++ initialization and 300 iterations. The clusters consisted of very high, moderate, and low dirty-rate pages. Earlier studies regarding pre-copying on ML preferred early transfer of high-rate pages to reduce retransmission.

5.7 Hybrid Migration Engine

The hybrid migration controller utilizes XenAPI events and Python state tracking. Pre-copy and bulk memory copies are made, dirty page statistics are collected, and logging is performed on a round-by-round basis. An adaptive switch occurs when a specified dirty page rate is exceeded or the maximum number of pre-copy iterations is reached. In post-copy, less than 200ms of inactivity occurs in the VM, followed by the transfer of CPU/OS state and fetching of the remaining pages as required. Measures include: total migration time (pre-copy time, stop and copy time, and post copy time), downtime (suspend to resume), dirty page retransmissions, and likelihood of SLA violation (downtime exceeds threshold). RAE; XenTop and iostat are used for network usage. These techniques are consistent with previously conducted research on live VM migration and machine learning optimization.

6. Results and Discussion

Its performance benefits can be attributed to predictive workload awareness based on LSTM forecasting, allowing for earlier triggering of migration and more stable hybrid switching, and dirty page clustering, providing a cost reduction in retransmission. This section presents the findings of the assessment of the intelligent hybrid live VM migration framework on the total time of migration, migrated downtime, dirty pages retransmission, prediction accuracy, SLA satisfaction, and network utilization. Our method is compared to three baselines: traditional pre-copy migration, traditional post-copy migration, and a hybrid no-machine-learning-no-clustering approach.

Migration time is a significant efficiency tool. The findings show that the hybrid model based on ML can dramatically reduce the overall time of migration. Table 3 shows the average total migration times.

Table 3. Average Total Migration Time

Migration Technique	Avg. Total Time (ms)
Pre-copy	6400
Post-copy	5600
Hybrid w/out ML	4700
Proposed Method	3750

The proposed method achieves an average reduction of:

- 41.40% compared to pre-copy
- 33.03% compared to post-copy
- 20.21% compared to basic hybrid

Latency sensitive applications are highly sensitive to downtime. Hybrid switching and ML prediction may help drastically cut down the downtime. Avg downtime: pre-copy 510 ms, post-copy 130 ms, hybrid without ML 95 ms, proposed method 58 ms (88.6, 55.4 and 38.9 percent reduction, respectively). The redundant transmissions are minimized by clustering as evidenced by dirty page retransmissions. Representing the number of pages in a re-transmission reconstituted pre-copy retransmission, reconstitution (hybrid) retransmission without ML, reconstitution (proposed method) retransmission, Ranging of regression analysis: RMSE = 0.041, MSE = 0.0017, MAE = 0.029 which indicates more significant results in the improvement of the SLA compliance. Pre-copy SLA errors: 22.6%; after copying 12.4%; without using ML 9.8%; with the proposed method 4.1%. The utilization of the network is also decreased by the proposed method: the decrease is about 31 times as much when compared to pre-copy and about 17 times when compared to hybrid without clustering. Unlike the recent methods (such as selective encryption via ML, pre-copy optimized via ML, blockchain-based methods, predictive/fuzzy methods, and pre-copy methods that perform better with the assistance of ML), the proposed system is distinguished by its ML-based prediction of data, dirty page clustering, hybrid implementation, the real-time change of the threshold values, and the workload-based control. A quick analysis of bandwidth congestion at the pre-copy to post-copy transition point shows a brief <8% increase that lasts less than 200 ms. By prioritizing high-write pages, clustering reduces congestion.

7. Conclusion

According to simulation results, the proposed XenServer cloud testbed method outperforms the traditional pre-copy, post-copy, and hybrid migration methods in terms of performance. The findings reveal that, with a high prediction precision (RMSE = 0.041), there is a 41% reduction in overall migration time, an 89.38% reduction in downtime, a 63.5% decrease in retransmitted dirty pages processed, and an 82% decrease in SLA violations. The findings indicate that memory-optimized optimization works effectively with machine learning-based workload prediction for dynamic and heterogeneous workloads. Unlike earlier approaches, this work offers proactive hybrid switching, on-the-fly adaptiveness, and fine-grain memory clustering. Thus, our work can be a viable and scalable solution for next-generation cloud computing systems.

References

[1] Dave, Akashbhai. "Intelligent Resource Management and Secure Live Migration in Cloud Environments: A Unified Approach using Particle Swarm Optimization, Machine Learning, and Blockchain on XenServer." Journal of Applied Science and Technology Trends 6, no. 2 (2025): 393-407.

[2] Haris, Raseena M., Mahmoud Barhamgi, Ahmed Badawy, Armstrong Nhlabatsi, and Khaled M. Khan. "Enhancing Security and Performance in Live VM Migration: A Machine Learning-Driven Framework with Selective Encryption for Enhanced Security and Performance in Cloud Computing Environments." Expert Systems 42, no. 2 (2025): e13823.

[3] Ramesh, Jayroop, Zahra Solatidehkordi, Khaled El-Fakih, and Raafat Aburukba. "Minimizing Virtual Machine Live Migration Latency for Proactive Fault Tolerance Using an ILP Model With Hybrid Genetic and Simulated Annealing Algorithms." IEEE Access (2024).

[4] Haris, Raseena M., Khaled M. Khan, Armstrong Nhlabatsi, and Mahmoud Barhamgi. "A Machine Learning-Based Optimization Approach for Pre-copy Live Virtual Machine Migration." Cluster Computing 27, no. 2 (2024): 1293-1312.

[5] Gupta, Ambika, Suyel Namasudra, and Prabhat Kumar. "A Secure VM Live Migration Technique in a Cloud Computing Environment Using Blowfish and Blockchain Technology." The Journal of Supercomputing 80, no. 19 (2024): 27370-27393.

[6] Zolfaghari, Rahmat. "Energy-Performance Aware Virtual Machines Migration In Cloud Network by Using Prediction and Fuzzy Approaches." Engineering Applications of Artificial Intelligence 131 (2024): 107825.

[7] Kavitha, Thummuluru, and Thatimakula Sudha. "Dynamic Multi-Objective Framework for Migrating Live Virtual Machines in the Cloud." GAMANAM: Global Advances in Multidisciplinary Applications in Next-Gen And Modern Technologies 1, no. 3 (2025): 172-182.

[8] Alubaidan, Haya A., and Sumayh S. Aljameel. "A Prediction Model for Improving Virtual Machine Live Migration Performance in Cloud Computing Using Artificial Intelligence Techniques." International Journal of Computers and Applications 46, no. 12 (2024): 1069-1087.

[9] Singh, Mandeep, Gurpreet Singh Panesar, and Sanjay Taneja. "Optimization, and Future Prospects of Live Virtual Machine Migration in Cloud Computing: A Comprehensive Survey on Techniques, Challenges, and Emerging Directions." In 2025 3rd International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA), IEEE, 2025, 1-6.

[10] Zhu, Xinying, Ran Xia, Hang Zhou, Shuo Zhou, and Haoran Liu. "An Intelligent Decision System for Virtual Machine Migration Based on Specific Q-Learning." Journal of Cloud Computing 13, no. 1 (2024): 122.

[11] Haris, Raseena M., Mahmoud Barhamgi, Armstrong Nhlabatsi, and Khaled M. Khan. "Optimizing Pre-copy Live Virtual Machine Migration in Cloud Computing Using Machine Learning-Based Prediction Model." Computing 106, no. 9 (2024): 3031-3062.

[12] Wang, Guikun, Bin Wen, Jingtao He, and Qingbin Meng. "A New Approach to Reduce Energy Consumption in Priority Live Migration of Services Based on Green Cloud Computing." Cluster Computing 28, no. 3 (2025): 207

Appendix:

Each VM is characterized by the following parameters:

Table A. Mathematical Notations

Symbol	Description	Symbol	Description
M	Total memory allocated to the VM	θ	ML-predicted dirty-page threshold
DP_t	Dirty-page generation rate at time t	T_{pre}	Time spent in pre-copy phase
B	Available network bandwidth	T_{post}	Time spent in post-copy phase
CPU_t	CPU utilization at time t	T_{total}	Total migration time
PF_t	Page-fault rate		