Enhancing Convergence of Live VM Migration with Dynamic Workloads

Interim Report
SCS 4224: Final Year Project

Nadeesha Nethmini Epa
Student of University of Colombo School of Computing
Email: 2020cs049@stu.ucsc.cmb.ac.lk

May 3, 2025

Supervisor: Dr. Dinuni K Fernando

Co-Supervisor: Dr. Jerome Dinal Herath

Declaration

The interim report is my original work and has not been submitted previously for any examination/evaluation at this or any other university/institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text

Student Name : D.N.N. Epa
Registration Number : 2020/CS/049
Index Number : 20000499

Signature & Date

This is to certify that this interim report is based on the work of Ms.D.N.N.Epa under my supervision. The interim report has been prepared according to the format stipulated and is of an acceptable standard.

Supervisor Name : Dr. Dinuni K.Fernando

Signature & Date

Co-Supervisor Name : Dr. Jerome Dinal Herath

Signature & Date

Contents

List of Figures

List of Tables

Acronyms

CC Cloud Computing. 1, 5

CRM Cloud Resource Manager. 11

CUA Capacity and Utility Agent. 8

FFS Fast Fourier Transform. 10

IaaS Infrastructure as a service. 8

MAP Markovian Arrival Process. 10

ML Machine Learning. 3, 8

OS Operating Systems. 14, 15

RTM Real Time Monitoring. 8

SaaS Software as a service. 8

VM Virtual Machine. ii, 1, 3–5, 11, 14–18

VMs Virtual Machines. 1, 2, 5, 11, 13, 15

1 Introduction

1.1 Introduction

As the technology grows cloud computing(CC) has become essential for developing reliable and efficient applications. One of the foundational concepts in cloud computing is virtualization. Through virtualization, users can access and utilize multiple hardware resources from a single physical machine, maximizing the machine's capability and efficiency. As the name suggests, Virtualization (Xing & Zhan 2012, Malhotra et al. 2014) defines the process of generating a virtual representation of actual hardware, allowing users to place multiple hardware resources inside one physical machine. This virtual representation is a Virtual Machine (VM).

In cloud data centers, a single server can host multiple VMs. However, a failure in that server would render all VMs on it unavailable, leading to significant service disruptions. VM migration has been introduced as a solution to this problem. It defines the idea of transferring a VM running on one server(source host) to another server (destination host). Since the failures of a server can happen at any time migration is essential for low-level system maintenance, load balancing, and fault management (Choudhary et al. 2017).

A crucial factor to consider during VM migration is identifying the convergence point. It is the point when migration can be completed without ongoing changes disrupting the process. Convergence is essential because it ensures that the source and destination VMs are synchronized, which is necessary for a seamless switchover with minimal downtime and no data inconsistencies. Users might experience service interruptions, data loss, or degraded application performance without proper convergence. During the migration process, there is a point where the same set of pages gets modified for several iterations. In that point, it has been identified as the convergence point. Identifying the convergence point will provide provisions to deterministically predict how long migration lasts upon a triggering event. This prediction allows for better planning and optimization of migration processes, ensuring that resources are efficiently allocated and that downtime is minimized. However, the nature of the workload significantly influences how this convergence point is determined.

Workloads are generally classified into two main categories: static and dynamic. A static workload maintains a consistent demand for computing resources over an extended period while dynamic workloads frequently change their behavior, leading to varying demands on computing resources. This variability makes dynamic workloads more complex to predict and manage, as it becomes challenging to determine the workload's next state accurately.

For static workloads, identifying the convergence point is relatively straightforward. During migration, there comes a point where the same set of memory pages (known as the working set) stabilizes and no longer changes. This stable state marks the convergence point, allowing the migration to complete smoothly.

In contrast, dynamic workloads require additional effort due to their unpredictable nature. The working set in dynamic workloads can change at any time, making it challenging to identify a stable convergence point. To address this, machine learning and statistical techniques are commonly employed. These methods help predict and manage the variability of dynamic workloads, providing more reliable solutions for achieving a stable convergence point during migration.

1.2 Background

1.2.1 Workload Categorization

VMs need to be migrated to prevent interruptions caused by server node failures. However, the methods used for migration depend heavily on the workload type. Workloads are generally classified into two main categories: static and dynamic.

A static workload maintains a consistent demand for computing resources over an extended period. Due to its stable nature, the behavior of a static workload is relatively easy to predict based on the current usage patterns, making the migration process more straightforward.

Within the category of dynamic workloads, there are several specific types, each with unique characteristics:

1. **Persistent Workloads**: These workloads are continuous and stable in resource demand, though they can experience occasional spikes. Applications with long-running tasks, such as databases, often generate persistent workloads (YCSB 2024). **Database Management Systems (DBMS) like MySQL, PostgreSQL, or MongoDB** can be given as examples of persistent workloads.

2. **Real-Time Workloads**: Real-time workloads require immediate processing with minimal latency, often found in applications like video streaming or IoT sensors. These workloads are highly sensitive to delays, making migration techniques for real-time workloads particularly complex, as they must minimize any disruption during the migration process. **Video Streaming Platforms like YouTube** and **Augmented Reality (AR) Applications** are few examples of real-time workloads.

3. **Serverless Workloads**: Serverless workloads(AWS 2024, Oracle 2024) are characterized by rapid, unpredictable demand changes, as they only consume resources when triggered by specific events. Serverless applications, such as cloud-based functions, make workload prediction challenging since they can vary significantly over short periods. **AWS Lambda, Google Cloud Functions, and Oracle** can be given as examples of serverless workloads.

4. **Interactive Workloads**: These workloads involve direct user interactions, such as online gaming, virtual desktops, and collaborative applications. Interactive workloads require low latency and high responsiveness to ensure a

seamless user experience. **Google Docs, Microsoft Teams, and Slack** are some of the interactive workloads.

5. **Burstable Workloads**: Burstable workloads experience sudden and intense spikes in resource demand for short durations, followed by periods of low or no activity. Examples include web servers handling unexpected traffic surges or financial trading platforms during market opens. **Social Media Platforms and Websites like Twitter or Instagram** are a few examples of burstable workloads.

Understanding these dynamic workload categories is essential for designing effective migration strategies that can maintain performance and availability across diverse application types. By tailoring migration techniques to the specific characteristics of each workload type, cloud data centers can enhance reliability, optimize resource usage, and ensure seamless application performance even in the face of varying demands.

Handling the dynamic nature is a challenging task because the frequently changing behavior of dynamic workloads doesn't lead to straightway convergence or smooth transitions during the migration. Because of that dynamic workloads require specialized techniques to handle their variability, including Profiling (Ye et al. 2014, Wu & Wolf 2008, Han et al. 2020) which involves recording past behaviors to help predict future workload patterns. In addition, statistical and machine learning (ML) methods (Wei et al. 2018, Naik 2022) are commonly used to handle the unpredictable nature of dynamic workloads. These techniques contribute to more accurate forecasting of workload changes, thereby optimizing the migration process. Further details on these methods are discussed in section 2.1.1.

1.2.2 Live VM Migration

Live VM Migration is a process designed to maintain application continuity when failures occur mid-execution. During Live VM Migration, the Virtual Machine (VM) is transferred from a failing server (source) to another server (destination) without interrupting the applications running inside the VM. This ensures minimal downtime and preserves the seamless operation of applications despite the underlying server issues. Christopher (2005) has shown several advantages of live migration as

- In live migration the entire operation system and all of its applications are moved at once. This will avoid many challenges encountered by process-level migration methodologies. (Christopher 2005)
- After the migration the host machine does not depend on the source and this solves the problem of residual dependencies.
- Live VM migration has the ability to complete the migration process without providing an interruption to the users.

Live VM migration can be categorized broadly into three(3) main techniques as **Pre-Copy migration, Post-Copy migration, and Hybrid migration**(He 2021, Christopher 2005, Fernando et al. 2020, Hines et al. 2009).

1. **Pre Copy Migration**
   In Pre-Copy(Christopher 2005, Fernando et al. 2020)migration, the memory transferring will happen via three(3) main steps as **push phase, stop and copy phase, and the pull phase.**

   (a) Push phase
   Memory pages are iteratively sent from the source to the destination while the source VM continues to run. In the first iteration, all pages are transferred, and in subsequent iterations, only modified pages (dirty pages) are sent. For static workloads, the dirty page count eventually stabilizes, marking a convergence point. However, for dynamic workloads, the dirty pages may change continuously, making it hard to find a convergence point.

   (b) Stop and copy phase
   After identifying the convergence point the source VM holds its execution and quickly sends the remaining pages, CPU state, and the I/O state to the destination VM. From this point onwards the destination VM starts to be executed. The duration that source VM paused its execution and started resuming in the destination is known as the **downtime** of the VM migration.

2. **Post-Copy Migration**
   The execution nature of the Pre-Copy technique is used to send the dirty pages to the destination through multiple iterations. But in Post-Copy (Hines et al. 2009, Fernando et al. 2020) migration it uses a mechanism that ensures no duplicates of the same page will be sent to the destination. The steps of this technique are as follows.

   (a) First, the VM at the source server paused its execution.Then, a minimal set of memory pages required for execution of the destination VM will be moved to the destination server.

   (b) Then the destination VM starts its execution and the source quickly starts sending the remaining memory pages to the destination. This process is known as **pre-paging**.

   (c) Same as the pull-phase in the Pre-Copy if the destination VM may notice that some pages have not been successfully sent to the destination a page fault will occur and the source quickly starts sending the missing pages to the destination. This is known as the **Demand paging**

3. **Hybrid Migration**
   Hybrid migration(Sahni & Varma 2012, Altahat et al. 2020) can be known as a combination of both Pre-Copy and Post-Copy techniques. Hybrid migration aims to increase the performance for both read-intensive and write-intensive workloads by minimizing downtime and avoiding redundant data transfers. Hybrid algorithm stages can be described as given below.

   (a) Migration begins with Pre-Copy, where memory pages are sent from the source to the destination VM while the source VM remains active. Unlike traditional Pre-Copy, Hybrid Migration does not wait for a convergence point.

   (b) After a short Pre-Copy period, the source VM is paused, and its CPU state is transferred to the destination. The VM resumes on the destination immediately, without waiting for all pages to transfer.

   (c) As the final step the Post-Copy algorithm starts the execution and the remaining dirty pages will copy from the source to the destination.

1.3 Motivation

With advancements in technology, cloud computing(CC) has become essential for users globally. It offers the ability to create and customize applications via the Internet, providing cost-effectiveness and flexibility in accessing resources. Leading cloud service providers, such as Cloud(GCP 2024), Amazon Web Services (AWS)(AWS 2024), Microsoft Azure(Microsoft 2024), and Oracle(Oracle 2024) utilize virtual machines (VMs) to deliver scalable, on-demand computing resources.

Given the potential for server failures at any moment, it is vital to implement effective strategies for managing these disruptions. VM migration has emerged as a critical solution to ensure continuous application availability. Over the years, various mechanisms have been developed to facilitate the migration of VMs between servers, minimizing downtime during such transitions.

However, many applications today operate under dynamic workloads, characterized by variability and complexity. These dynamic environments present several challenges (Zhang & Boutaba 2014, Hossain & Song 2016, Cerotti et al. 2012) that complicate the migration process. A significant challenge is the difficulty in predicting future workload behavior, which hinders the selection of the most suitable migration strategy. Not selecting the best migration technique may lead to a longer Total Migration Time (TMT), increasing the risk that a server may fail before the migration process is completed. This unpredictability can lead to resource inefficiencies, highlighting the need to improve the performance of live migration techniques tailored for dynamic workloads.

2 Background

2.1 Literature Review

A literature review was conducted for the topic **Live VM Migration Convergences in a Dynamic Workload** and the report consists of a detailed explanation of live migration techniques, an explanation about the dynamic workload, and the different types of dynamic workload handling methods with their advantages and limitations.

2.1.1 Dynamic workload

A dynamic workload (Hossain & Song 2016, Cerotti et al. 2012) consists of several computational tasks that change the behavior of tasks very frequently over time. Because of its changing behavior, it is hard to predict workload patterns for dynamic workloads. This makes dynamic workload handling a challenging and complex task. Some of the key challenges of dynamic workloads are given below.

1. Dynamic workloads consist of different types of applications that consist of their own and different kinds of resource requirements which is known as the **Heterogeneity** (Zhang & Boutaba 2014).
2. Dynamic workloads have the problem of over-provisioning and under-provisioning. Because of the dynamic nature, it is hard to predict the resource demand hence most of the applications provide resources more than enough to execute the operations. This may lead to a waste of resources. In similar cases, the resources provided may not be enough for the applications to execute which leads to under-provisioning and the failures in application process. This is known as the applications do not meet its **peak demands**. In both cases, part of or the whole cost used for the application process has become a waste (Hossain & Song 2016).

Dynamic workloads can be handled based on the following approaches:

- Mathematical/Machine Learning Techniques
- Statistical Techniques

To deal with dynamic workloads, these techniques require capturing and collecting records of the current behavior of the workload to predict future behaviors. For that process, a well-known technique called **profiling** has been used by several researchers(Wu & Wolf 2008, Ye et al. 2014, Han et al. 2020).

2.1.2 Profiling

Profiling is a concept used to capture the behavior of workloads based on their characteristics. There are two profiling techniques runtime profiling and offline profiling (Wu & Wolf 2008). Runtime profiling tries to capture the characteristics of an application when the application starts its execution. Because of that, runtime profiling (Wu & Wolf 2008) is used to deal with dynamic workloads. The captured data will be used to increase application efficiency. CPU usage, memory usage, and network consumption are some of the runtime profiling information.

The studies conducted by Wu & Wolf (2008),Ye et al. (2014), and Han et al. (2020) have used profiling techniques to identify the behavior of a workload. The main idea behind the concept used by these research works is to gather runtime profiling information such as CPU time, memory usage, and service time and understand the workload behavior using the collected information. The study followed by Wu & Wolf (2008) collects real-time information such as task service times, edge utilizations, and task utilizations and then uses a duplication and mapping algorithm that can capture the changes in the workload by using the collected profiling information. So the tasks with high computational demands will be duplicated across multiple cores by the task duplication algorithm. The task mapping algorithm will assign tasks to the processors to minimize the overhead. Also, the study of Ye et al. (2014) collects profile information for various types of workloads such as CPU-intensive, memory-intensive, and network-intensive. Then by analyzing these details, the characteristics and resource demands of different types of workloads will be identified. Finally, with the use of these details, they implemented two(2) models as a consolidation planning module and a migration planning module to complete the migration process with minimal effect on the performance. Han et al. (2020) has introduced a combination of profiling systems and refinement frameworks to identify micro-services placement with dynamic workloads.

One of the most important points highlighted by the above studies (Wu & Wolf 2008, Ye et al. 2014, Han et al. 2020) is profiling techniques has the ability to measure the system performance for various workload conditions(CPU intensive, memory intensive, network intensive, etc). So the workload changes can be identified in real time. Also profiling incurs low overhead because profiling can directly monitor specific metrics and uses simple analytical techniques which do not require high computational power.

2.1.3 Workload Prediction

1. **Mathematical/Machine Learning Techniques**
   Various research efforts have addressed the challenges of managing dynamic workloads, utilizing methods such as dynamic thresholds (Lin et al. 2011), reinforcement learning (Wei et al. 2018), use of autonomic computing (Sah & Joshi 2014), etc.

   A study conducted by Lin et al. (2011) proposed a dynamic resource alloca-

tion schema that adjusts resources according to the application's changing workload. Given that dynamic workloads can fluctuate throughout the application's lifecycle, this schema uses a threshold value to determine optimal instances for resource allocation. Based on this threshold, resources are allocated dynamically, adapting to the application's immediate needs.

Similar systems can be found in the solution provided by Sah & Joshi (2014), a solution based on autonomic computing. Autonomic computing refers to the term that applications can change themselves according to the changes of the application environment without any external help(Without notifying the user). The algorithm created using autonomic computing, consists of a capacity and utility agent (CUA) and a resource controller. CUA is used to collect data about the resources and the capacity of the VMs. According to the collected data, an initial schedule was implemented, and if the workload changes then the resource controller will change the schedule to allocate resources in a better way.

Enhanced Dynamic Johnson Sequencing is another technique proposed by Banerjee et al. (2023), that uses a combination of ML and RTM to handle the nature of a dynamic workload. It is also known as OptiDJS+. The algorithm provides a schedule based on the dynamic quantum value (minimum execution time + maximum execution time)/2 of the tasks.

Also the framework proposed by Wei et al. (2018) claims a workflow scheduling algorithm using a sequential decision-making approach based on reinforcement learning. The ML approach used in this proposed algorithm is called the Q-learning. This algorithm aims to identify suitable Infrastructure as a Service (IaaS) providers for Software as a Service (SaaS) applications, optimizing resource allocation and minimizing costs. The Q-learning approach continually learns and adapts based on workload changes, making it particularly suited to dynamic environments.

As explained, several instances of using machine learning to handle dynamic workloads exist. However, there are a few important points to be considered when using ML techniques. To provide an accurate prediction, ML models need to be trained properly (Wei et al. 2018). This required extensive and accurate training data to make reliable predictions. But in this instance acquiring data from a dynamic workload can be challenging since its workload behavior will change frequently(Khelghatdoust et al. 2016). Additionally, dynamic environments increase the risk of overfitting, where a model becomes too specialized to the training data, resulting in reduced effectiveness for real-world applications(Wei et al. 2018). Also implementing a proper model can be resource-intensive because it requires a high computational power to train and run complex models.

2. **Statistical Techniques** Statistical techniques have gained popularity for predicting future workloads due to their simplicity, lower data requirements, and transparency in operations (Devi & Valli 2023, Calheiros et al. 2014). Many studies in this area focus on time series forecasting, which leverages historical data to identify patterns over time, providing insights into future workload trends. Another common approach is the use of Markov chains, which model workload transitions between states, allowing predictions based on probable sequences. These statistical methods are especially valuable for dynamic workloads, offering an accessible and computationally efficient alternative to more complex machine learning models.

   (a) **Time Series Forecasting**

   Calheiros et al. (2014) has conducted a study about workload prediction using **ARIMA** model. ARIMA stands for the Auto-Regressive Integrated Moving Average and it is used to predict the future workload. This paper has introduced an analyzer implemented using the ARIMA model. First, the data that needs to predict the future workload such as CPU usage, memory usage, etc will be fed into the ARIMA model. An important point considered here is these data must be stationary. For that process, a transformation method called differencing has been used. ARIMA model needs three(3) parameters to be fed the number of differences used, the auto-correlation value, and the partial auto-correlation value. The analyzer used in this paper will be responsible for updating the model with new data to increase the prediction accuracy of future workload behaviors. These predicted results have been used to ensure that enough resources are available for users to complete their tasks. With that Calheiros et al. (2014) has been able to increase the quality of service(QOS) by predicting future behavior using the ARIMA model.

   The study conducted by Devi & Valli (2023) has used a combination of the ARIMA model and ANN(Artificial Neural Network) to predict the behavior of future workloads. So this hybrid model can be introduced as an enhanced version of Calheiros et al. (2014). ARIMA model was used to deal with the linear components and ANN was used to model the non-linear components. With that, the accuracy of the future workload prediction has been increased. Finally, the study has evaluated the performance of the new method using metrics such as mean absolute error and root mean squared error and the results have proven that the proposed solution works efficiently in a dynamic cloud environment to predict future workloads.

   Ganapathi et al. (2010) has proposed a statistic-driven working model that can be used to predict the future behavior of a workload. Mainly the paper tries to predict the resource demands for applications that use MapReduce(Hashem et al. 2016). Mapreduce tasks consist of several

jobs which are known as Hadoop jobs. The proposed solution named as **Kernel Canonical Correlation Analysis (KCCA)** will predict the execution times, and resource demands for the Handoop jobs. KCCA consists of a special feature called **Feature Vectors** which are constructed using job configuration parameters and input data characteristics. With the use of feature vectors, KCCA was able to provide accurate predictions about the future workloads.

(b) **Markov Chains**

The research conducted by Kim et al. (2018) produced a framework called **CloudInsight**. The CloudInsight framework utilizes a hybrid approach that combines **Fast Fourier Transform (FFS)** and **Markov chains** to improve short-term workload prediction accuracy. FFS is used to detect repetitive workload patterns, while the Markov chains model transitions between states to account for immediate, short-lived fluctuations. By leveraging Markov models, CloudInsight adapts to sudden changes in workload behavior, providing real-time predictions for highly dynamic and bursty workloads. This combination enables CloudInsight to adjust its ensemble predictor to both periodic and irregular workload patterns, resulting in improved accuracy and reliability in cloud resource provisioning.

Pacheco-Sanchez et al. (2011) uses Markovian Arrival Processes (MAP) to model and predict the time-varying characteristics of cloud workloads, specifically for web traffic. This approach allows for performance prediction, including metrics such as server utilization, queue length, and response times. MAPs capture not only the distribution of workload arrival rates but also the temporal dependencies in these rates, accommodating heavy-tail distributions common in HTTP traffic. By fitting MAP parameters to real HTTP log data, the framework enables efficient, queueing-based performance analysis without needing extensive simulation. This model is then used to predict Quality of Service (QoS) metrics, supporting resource allocation decisions in cloud environments by ensuring the provision of sufficient resources for varying workload intensities

Also, the research conducted by Gambs et al. (2012) uses a n-Mobility Markov Chain (n-MMC) to predict future locations by modeling an individual's movement patterns based on the last **n** locations visited. Each location or "point of interest" (POI) represents a state, with transitions indicating the probability of moving from one POI to another. By considering sequences of locations rather than just the current location, the n-MMC model improves prediction accuracy, capturing more complex mobility patterns.

In recent years, application providers have increasingly favored statistical approaches to predict future workload behaviors due to their simplicity and ease

of interpretation. Unlike machine learning techniques, statistical methods require smaller datasets to produce effective models (Calheiros et al. 2014, Devi & Valli 2023), making them more accessible and efficient for real-time predictions. Additionally, statistical models demand less computational power (Hashem et al. 2016) during both training and execution, making them a practical choice for environments with limited resources or tight processing requirements. This combination of advantages makes statistical approaches highly suitable for many predictive applications in workload management.

2.2 Existing Dynamic Workload Migration Strategies

Naik (2022) has claimed a solution for dynamic workload migration using a technique called **Adaptive Push-Pull**. This algorithm consists of a combination of a push function and a pull function. In the concept of Adaptive Push-Pull the push function is used to distribute the workload among the resources and when a VM is overloaded . The pull function is invoked when the VM is underloaded . For this process, a VM manager and the CRM are used. VM manager will be responsible for managing the workload within the same resource and CRM will be responsible for managing the workload around all the resources. So with the use of this method, the workload will be shared among the resources.

Also similar systems can be found in Lu et al. (2015), an optimal scheduling algorithm called **VHaul** to solve the problem of migrating multi-tier applications. These multi-tier applications consist of a group of co-related VMs that makes the application inherently dynamic. So to create the scheduling algorithm the VMs will be categorized according to their relationships. VMs belonging to the same application will be assigned to the same group. Then the production of resource utilization and migration time are used to decide the impact from the current VM to the next VM to be migrated. Then the migration schedule will be created as the smaller value for the production of resource utilization and migration time means less impact for the next VM to be migrated.

Khelghatdoust et al. (2016) came up with a solution **GLAP**, which is a combination of a gossip-based learning algorithm and Q-learning. GLAP uses continuous monitoring of the resource demands of the VMs to predict the behavior of future workloads. The table 2.2 shows the comparison of the existing methods.

	Pros	Cons
Adaptive Push-Pull	• Provides an adaptive push-pull system to dynamically switch between push and pull functions to react to the changes in the workload. • Solves the problem of overloading and underloading in the VMs.	• With a highly dynamic workload CRM and VMM will not be able to handle the problem of overloading.
vHaul	• Provide an efficient scheduling algorithm by considering the resource utilization and the migration time.	• Have to calculate the resource utilization and migration time whenever the workload changes.
GLAP	• Provides a threshold-free approach to detect future behaviors and migrate a workload without overloading.	• Have to use a lot of computation power and resources to train an accurate model

Table 1: Comaprison for dynamic workload migration techniques

2.3 Research Gap

Migration operations are often triggered by sudden resource or power failures, maintenance needs, or load-balancing requirements. However, predicting the future behavior of a workload for migration is challenging, especially with dynamic workloads where CPU, memory, and storage demands fluctuate rapidly. This unpredictability complicates decisions on deciding the convergence point. Additionally, the variable behavior of workloads—whether they become CPU-intensive, memory-intensive, or network-intensive—can influence the success of the migration process. Selecting an incorrect migration strategy can further increase the total migration time (TMT) and elevate the risk of server failure during migration, making timely and accurate prediction essential.

Existing research on dynamic workload migration has primarily focused on developing complex mechanisms to adapt migration schedules based on workload changes. Many of these solutions rely on data such as resource utilization metrics,

migration durations, and the load status of VMs (whether they are overloaded or underloaded). This information, however, requires significant processing time to analyze and interpret, which can delay response to real-time changes in workload demands.

Machine learning and statistical techniques provide potential solutions for forecasting workload behavior before migration. Machine learning methods, while effective, require extensive training periods, large datasets, and considerable computational resources (Khelghatdoust et al. 2016, Wei et al. 2018). By contrast, statistical approaches demand smaller datasets (Calheiros et al. 2014, Devi & Valli 2023) and lower computational power (Hashem et al. 2016) for both training and execution. Studies by Calheiros et al. (2014), Hashem et al. (2016), and Devi & Valli (2023) demonstrate that even with limited computational resources, statistical techniques can reliably predict future workload behaviors.

In VM migration, the full state of the VM must be transferred to the target machine, which demands substantial bandwidth and is recognized as a resource-intensive process. Employing techniques with high computational overhead to handle workload dynamics can further strain the system, negatively impacting applications running on the VMs. Therefore, this research seeks to bridge the gap by developing a lightweight, performance-efficient solution for dynamic workload migration. By using profiling information (e.g., CPU and memory usage), this solution will employ statistical models to predict future workload behaviors and schedule migrations accordingly. The research aims to minimize both the total migration time and downtime, reducing the performance impact on applications during migration and mitigating the risk of server failure.

2.4 Research Questions

1. How to identify the convergence point of dynamic workloads in live migration with workload-specific behavior?
   This question examines using Markov Chains to detect the convergence point, considering how workloads like persistent, real-time, and serverless uniquely impact convergence.
2. How can the performance of vanilla pre-copy be improved upon detection of convergence point in terms of total migration time, downtime, and application performance?
   This focuses on improving the performance of the applications by reducing the total migration time, and downtime with minimal impact on the performance.

2.5 Aims and Objectives

2.5.1 Aim

The primary goal of this research is to develop a method for selecting the optimal migration strategy for a dynamic workload, aiming to achieve efficient migration while reducing total migration time and downtime compared to current approaches.

2.5.2 Objectives

The main objectives of the research are as follows:

1. Design and develop a method to predict the future behavior of a dynamic workload.
2. Design and develop an algorithm to select the best migration strategy to migrate a dynamic workload.
3. Identify the convergence point of a dynamic workload.
4. Evaluate the performance of migration using total migration time, downtime, and application performance metrics by incorporating industrial accepted benchmarks such as Sysbench, YCSB (Yahoo cloud serving benchmark), Stress, Memcached, etc. These benchmarks support generating workloads of different natures such as **persistent workloads, real-time workloads, CPU-intensive workloads, and memory-intensive workloads.**

2.6 Scope

2.6.1 In Scope

- **VM live migration with dynamic workloads**: The research will focus on developing a strategy to migrate a dynamic workload using Live VM migration.

- **Single VM migration**: The research focused on migrating a dynamic workload in a single VM.

- **Implementing a working prototype**: At the end of the research a working prototype will be developed to find the convergence point and migrate a dynamic workload with minimal total migration time and downtime.

- **QEMU-KVM Hypervisor**: The research uses QEMU-KVM as the hypervisor for implementing and evaluating the working prototype.

- **Ubuntu Host OS**: The research will focus on Ubuntu Servers as the host OS.

- **Linux guest OS**: Linux will be considered as the guest OS for this research.

2.7 Significance of the Research

This research project aims to enhance the existing live migration techniques for dynamic workloads by finding a new strategy to decide the convergence point of dynamic workload based on the data extracted and analyzed from profiling. In the migration process, a critical factor is how quickly the VM is moved to a safe destination which means the total migration time should be minimal. An incorrect migration strategy can significantly increase the time needed to identify this convergence point, thus extending the total migration time. This delay can heighten the risk of the source server failing during migration and the VMs inside that server will become unavailable. The goal of this research is to optimize live migration techniques for dynamic workloads, achieving minimal total migration time and enhanced system performance to ensure continuous availability of services. As most modern applications operate with dynamic workloads, the outcomes of this research will support a better user experience and improve service quality for end users.

3 Design and Methodology

3.1 Planned Research Approach

1. Test-bed setup: The test-bed setup consists of two physical servers that are interconnected using a Gigabit Ethernet. QEMU/KVM is chosen as the hypervisor and a Ubuntu Server is used as the host OS. The VMs consist of a Linux-based OS. The following figure provides a high-level view of the test-bed setup.

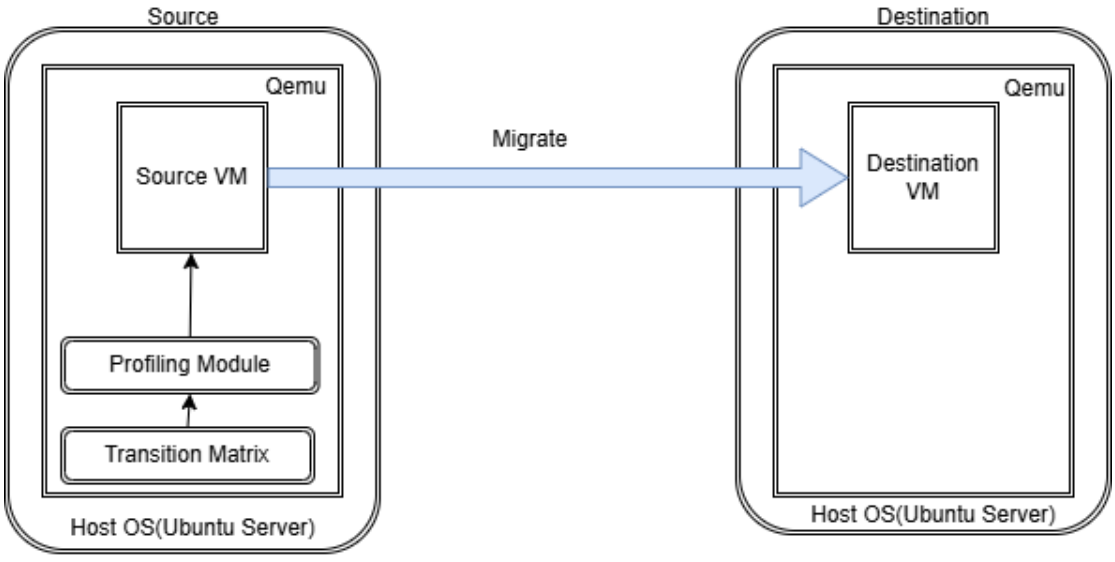

Figure 1: System Architecture

2. Implementation of the transition matrix and count matrix: A transition matrix and a count matrix were created to model workload behavior using Markov chains to define all possible state transitions. **CPU** usage, **memory** usage, **incoming network** usage, and **outgoing network** usage will be taken as the resources and **less than 25%**, **between 25% and 50%**, and **greater than 50%** will be taken as the resource usage levels(for simplicity). Based on these conditions, 81 possible states are identified, representing all combinations of resource usage levels.
Here are a few example states:

- **State 1**: CPU $< 25\%$, Memory $< 25\%$, Incoming Network $< 25\%$, Outgoing Network $< 25\%$
- **State 2**: CPU $< 25\%$, Memory between 25% and 50%, Incoming Network $< 25\%$, Outgoing Network $< 25\%$
- **State 3**: CPU $> 50\%$, Memory $> 50\%$, Incoming Network between 25% and 50%, Outgoing Network $> 50\%$

CPU, memory, incoming network, and outgoing network usage can be monitored and mapped into percentages to determine the **current state** of the VM accordingly.

The **Count Matrix** is used to represent the number of times the VM state changes from one state to another during the observation period. Since there are 81 possible states (based on CPU, memory, and network usage levels), this matrix will have dimensions of 81×81. Each entry (i, j) in the matrix records the count of transitions from state i to state j.
Example: Suppose during the observation, the VM moves from:

- State 1 to State 2 (5 times),
- State 2 to State 3 (3 times),
- State 1 to State 3 (2 times).

In this case, the count matrix will update as follows.

- $Count(1, 2) = 5$, representing 5 transitions from State 1 to State 2,
- $Count(2, 3) = 3$, representing 3 transitions from State 2 to State 3,
- $Count(1, 3) = 2$, representing 2 transitions from State 1 to State 3.

The **Initial Transition Matrix** is derived from the Count Matrix and represents the probability of moving from one state to another. It is also an 81×81 matrix. Each entry (i, j) is calculated as the ratio of the number of transitions from state i to state j to the total number of transitions out of state i.
Example: Using the counts from above, if there were 10 total transitions from State 1, then the transition probability is updated as follows:

- $Transition(1, 2) = \frac{Count(1,2)}{10} = \frac{5}{10} = 0.5$,
- $Transition(1, 3) = \frac{Count(1,3)}{10} = \frac{2}{10} = 0.2$.

If there were a total of 6 transitions from State 2, then:

- $Transition(2, 3) = \frac{Count(2,3)}{6} = \frac{3}{6} = 0.5$.

3. Implementation of the profiling module: A profiling module was implemented inside the source server which captures the VM state for each second and updates the count matrix according to its current state. Then the transition matrix will be updated accordingly.

Let's consider a VM with three possible states based on CPU and memory usage:

- **State 1**: Low CPU and memory usage
- **State 2**: Medium CPU and memory usage
- **State 3**: High CPU and memory usage

Initially, the **Count Matrix** is set as follows, representing previously observed transitions between states:

$$\text{Count Matrix} = \begin{bmatrix} 2 & 3 & 1 \\ 1 & 0 & 4 \\ 2 & 1 & 0 \end{bmatrix}$$

The **Transition Matrix** is calculated by dividing each element in a row of the Count Matrix by the total number of transitions from that state. For instance:

- For State 1, there are $2 + 3 + 1 = 6$ total transitions.
- For State 2, there are $1 + 0 + 4 = 5$ total transitions.
- For State 3, there are $2 + 1 + 0 = 3$ total transitions.

The resulting **Transition Matrix** is:

$$\text{Transition Matrix} = \begin{bmatrix} \frac{2}{6} & \frac{3}{6} & \frac{1}{6} \\ \frac{1}{5} & 0 & \frac{4}{5} \\ \frac{2}{3} & \frac{1}{3} & 0 \end{bmatrix} = \begin{bmatrix} 0.33 & 0.50 & 0.17 \\ 0.20 & 0 & 0.80 \\ 0.67 & 0.33 & 0 \end{bmatrix}$$

In the next second, the profiling module captures the following VM resource usage:

- **CPU Usage:** 60%
- **Memory Usage:** 55%

Based on these percentages, the current state of the VM is classified as **State 3** (High CPU and memory usage).

Assume that in the previous second, the VM was in **State 1** and has now transitioned to **State 3**. So the value of the Count(1,3) will be incremented by 1. This transition updates the Count Matrix as follows:

$$\text{Updated Count Matrix} = \begin{bmatrix} 2 & 3 & 2 \\ 1 & 0 & 4 \\ 2 & 1 & 0 \end{bmatrix}$$

Now, the Transition Matrix is recalculated using the updated Count Matrix:

- For State 1, there are now $2 + 3 + 2 = 7$ total transitions.
- For State 2, transitions remain at 5.
- For State 3, transitions remain at 3.

The **Updated Transition Matrix** is:

$$\text{Updated Transition Matrix} = \begin{bmatrix} \frac{2}{7} & \frac{3}{7} & \frac{2}{7} \\ \frac{1}{5} & 0 & \frac{4}{5} \\ \frac{2}{3} & \frac{1}{3} & 0 \end{bmatrix} = \begin{bmatrix} 0.29 & 0.43 & 0.29 \\ 0.20 & 0 & 0.80 \\ 0.67 & 0.33 & 0 \end{bmatrix}$$

This updated Transition Matrix now reflects the most recent observed behavior, providing an improved model for predicting future state transitions based on the latest profiling data.

4. Design and development of the algorithm: An algorithm was implemented to map all possible states to the most suitable migration decision. Algorithm 3.1 shows the pseudo-code of the algorithm.

This migration decision algorithm dynamically selects the most efficient migration technique for a VM based on its current and predicted workload. In the migration algorithm, the **current state** of the VM and the **transition matrix** are used to predict the VM's next behavior over n look-ahead steps.

The steps to calculate the next state are as follows. Consider the transition matrix created in section 3.1. Assume a look-ahead of $n = 2$ steps and an initial **current state vector** where the VM starts in **State 1** (represented as $[1, 0, 0]$).
1. Calculate T^n, the transition matrix raised to the power of $n = 2$:

$$T^2 = T \times T = \begin{bmatrix} 0.392 & 0.277 & 0.331 \\ 0.536 & 0.099 & 0.365 \\ 0.467 & 0.578 & 0.0 \end{bmatrix}$$

2. Multiply T^2 by the initial state vector $[1, 0, 0]$:

$$\text{Next state probabilities} = \begin{bmatrix} 0.392 & 0.277 & 0.331 \\ 0.536 & 0.099 & 0.365 \\ 0.467 & 0.578 & 0.0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.392 \\ 0.536 \\ 0.467 \end{bmatrix}$$

3. Select the Next State: Since **State 2** has the highest probability (0.536), it is chosen as the next predicted state after looking ahead two steps.

After calculating the next behavior the algorithm determines the initial migration method—pre-copy, post-copy, or hybrid. For that, the following four(4) conditions are used.

- If a VM is memory-intensive, it is migrated using **post-copy**
- For a network-intensive workload with mostly outgoing packets, the VM is migrated in **post-copy**
- If there are mostly incoming packets, they are migrated in **pre-copy.**
- All other VM workloads are migrated adapting the **hybrid method**.

If pre-copy is chosen, the algorithm monitors for high memory usage or if certain limits are met, switching to post-copy if necessary. For the hybrid method, it also evaluates workload and iteration conditions to determine whether a switch to post-copy is beneficial. If there is a converging behavior, it will be identified using exponential decay which shows that the memory dirty page rate has reduced for at least 3 consecutive intervals.

Algorithm 1 Migration Decision Algorithm

1: **Input:** Max_Iterations = m, Look_ahead_states = n
2: **Initialize:** Current_Migration_Decision ← Get the migration decision based on the current state
3: Next_behavior ← Calculate the behavior of the upcoming n states using the transition matrix
4: **if** Current_Migration_Decision == `post_copy` **then**
5: No switching during the migration
6: **else if** Current_Migration_Decision == `pre_copy` **then**
7: **if** Next_behavior is memory-intensive **and** (Iteration_count = m **or** Dirty page rate shows an exponential decay) **then**
8: Switch to `post_copy`
9: **else**
10: Vanilla `pre_copy`
11: **end if**
12: **else if** Current_Migration_Decision == `Hybrid` **then**
13: **if** Iteration_count = m **or** Dirty page rate shows an exponential decay **then**
14: Switch to `post_copy`
15: **end if**
16: **end if**=0

In the algorithm 3.1, the **Max Iterations** count will be a workload-specific parameter. Experiments have already been conducted to find the **Max iteration** count for persistent workloads and real-time workloads using the industrially accepted benchmarks and tools such as **YCSB(persistent workloads) and Stress(real-time workloads)**.More details about these experiments will be discussed in section 3.1.1

5. Implementation of the working prototype: Using the implemented algorithm a working prototype will be implemented for the KVM/QEMU platform.

6. System Evaluation: Finally the implemented prototype system will be evaluated through experiments conducted on the test bed to assess its efficiency. The final result of the research aims to reduce the total migration time and downtime while reducing the performance impact for the applications executed during the migration process.

3.1.1 Preliminary Results and Methodology

Time series forecasting was explored as a starting point to predict future workload behavior. There are two(2) main categories to be considered as univariate and multivariate time series models.

- **Univariate Models:** Models such as ARIMA, ETS, and FB Prophet can predict a single variable at a time. For this research, using univariate models would require four separate models, one each for CPU, memory, incoming network, and outgoing network.
- **Multivariate Models:** VAR (Vector AutoRegression) is a multivariate model capable of predicting multiple variables simultaneously, making it potentially more suitable for predicting all four resources together.

To determine the most resource-efficient approach, experiments were conducted to compare the resource usage of these models during predictions. These tests provided insights into the computational demands of each model type, guiding the selection of an optimal prediction method for dynamic workloads in cloud environments. Based on the results, multivariate forecasting was identified as the less resource-intensive prediction model. Results will be shown in figure 2.

However, several challenges arise when using time series forecasting. A significant issue was that even within the same workload type, different workloads required distinct parameter settings for the models. Consequently, the algorithm needed extra time to analyze past data and adjust parameters to fit each workload accurately, making it difficult to develop a generalizable solution.

Additionally, prediction accuracy varied across resources. Some resources were accurately forecasted, while others showed substantial prediction errors. For instance, as shown in figure 3, the model accurately predicted memory usage but struggled to forecast incoming network usage for the same workload.

To address these limitations and develop a more generalized solution, the approach was shifted to using **Markov chains**(Kim et al. 2018, Pacheco-Sanchez et al. 2011). Unlike time series models, which require tuning specific parameters for each workload, Markov chains offer a state-based approach that can adapt more easily to varying workload patterns without extensive reconfiguration. By modeling resource usage levels as states with probabilistic transitions, Markov chains enable a more flexible and scalable framework for predicting dynamic workloads across multiple resources, mitigating the need for constant parameter adjustments. This approach simplifies workload prediction and enhances consistency across different types of resources.

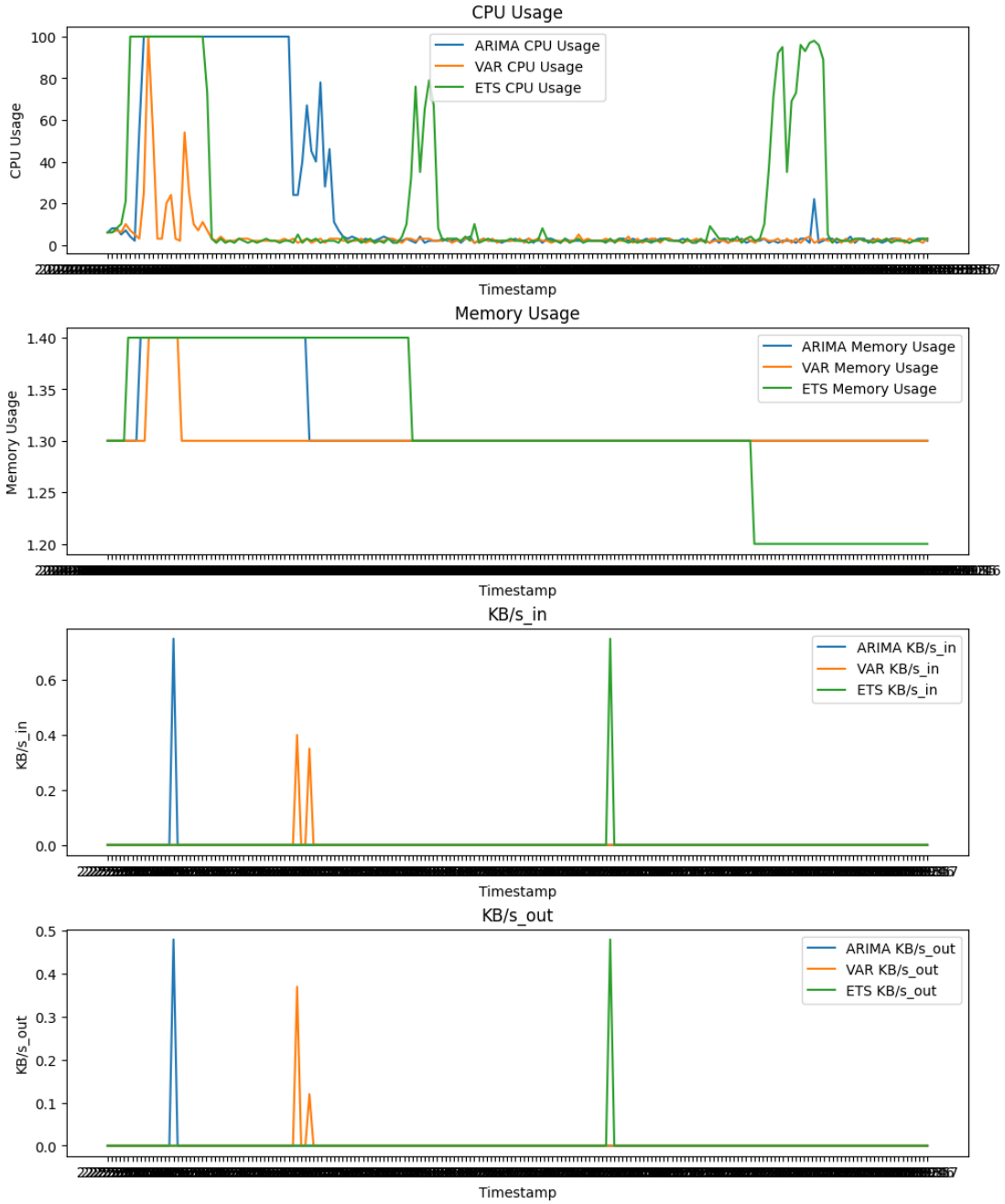

Figure 2: Univariate Forecasting Vs Multivariate Forecating

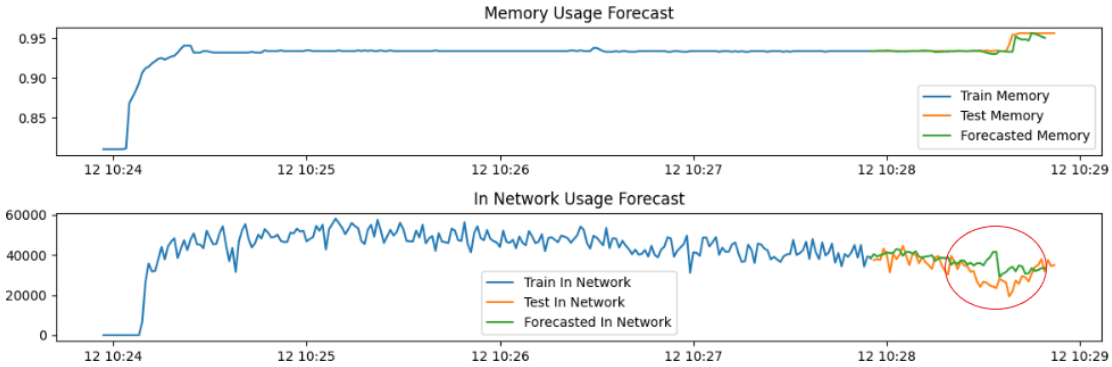

Figure 3: Forecasting Results for YCSB Workloads

To address these limitations and develop a more generalized solution, the approach was shifted to using **Markov chains**(Kim et al. 2018, Pacheco-Sanchez et al. 2011). Unlike time series models, which require tuning specific parameters for each workload, Markov chains offer a state-based approach that can adapt more easily to varying workload patterns without extensive reconfiguration. By modeling resource usage levels as states with probabilistic transitions, Markov chains enable a more flexible and scalable framework for predicting dynamic workloads across multiple resources, mitigating the need for constant parameter adjustments. This approach simplifies workload prediction and enhances consistency across different types of resources.
According to the algorithm shown in 3.1 the proposed solution worked with 2 parameters as the **Max Iterations and the no of looking ahead states**. Max Iteration count is used to handle the behavior of not converging workload. It is used with the hybrid migration. In a typical hybrid migration, the transition from pre-copy to post-copy occurs if the migration takes more than 50 iterations to converge. However, if it is determined that the workload will not converge, post-copy migration can be initiated without waiting for the 50 iterations. To identify this threshold, several experiments were conducted. Hybrid migration with five(5) maximum iterations is been represented as **Hybrid(5)** in the 4 and 5. According to the results shown in Figures 4 and 5, it is evident that for persistent workloads, the maximum number of iterations before transitioning is 4. For real-time workloads, the optimal migration technique is post-copy.

Further experiments were conducted to compare the resource usage taken by the models to predict future behaviors. The results are shown in figure 6 and 7. According to this information, Markov has the lowest memory consumption while the VAR model has the lowest CPU consumption. The next steps in the research will focus on exploring methods to reduce CPU usage.

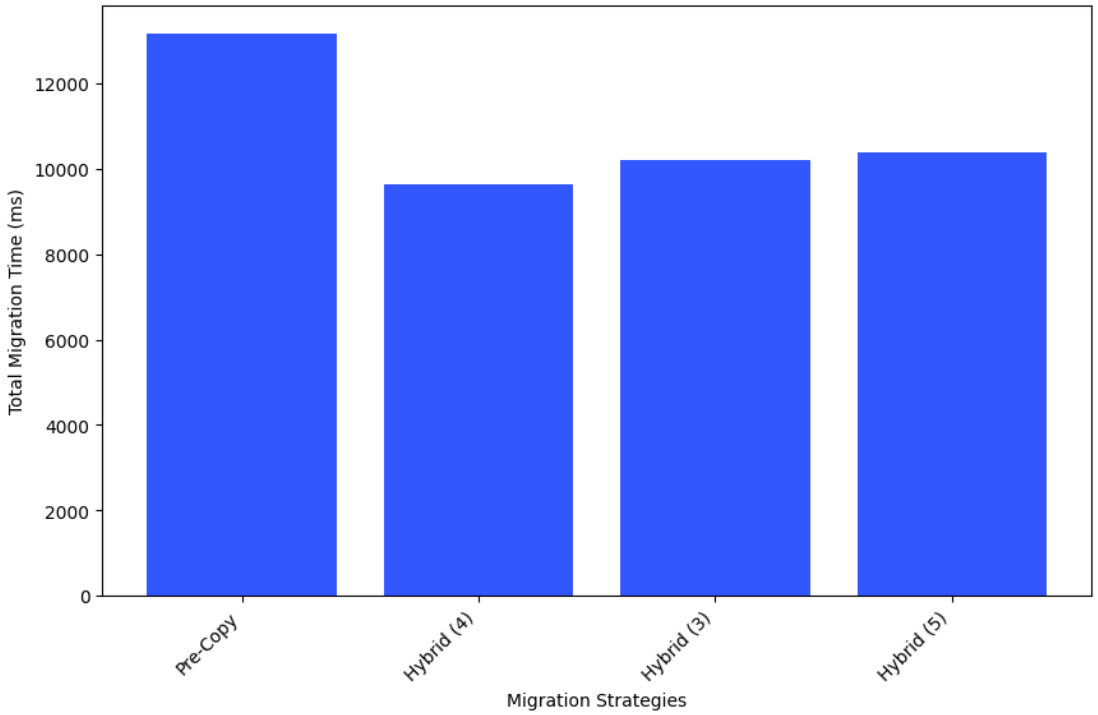

Figure 4: Total Migration Time Variation with Migration Strategy-YCSB workloads

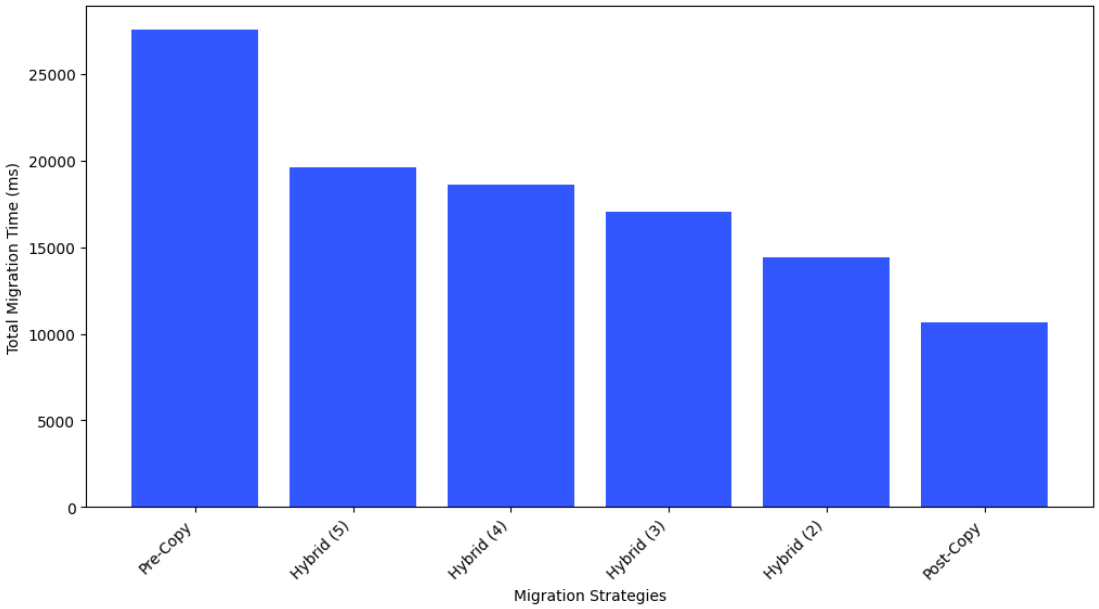

Figure 5: Total Migration Time Variation with Migration Strategy-Stress workloads

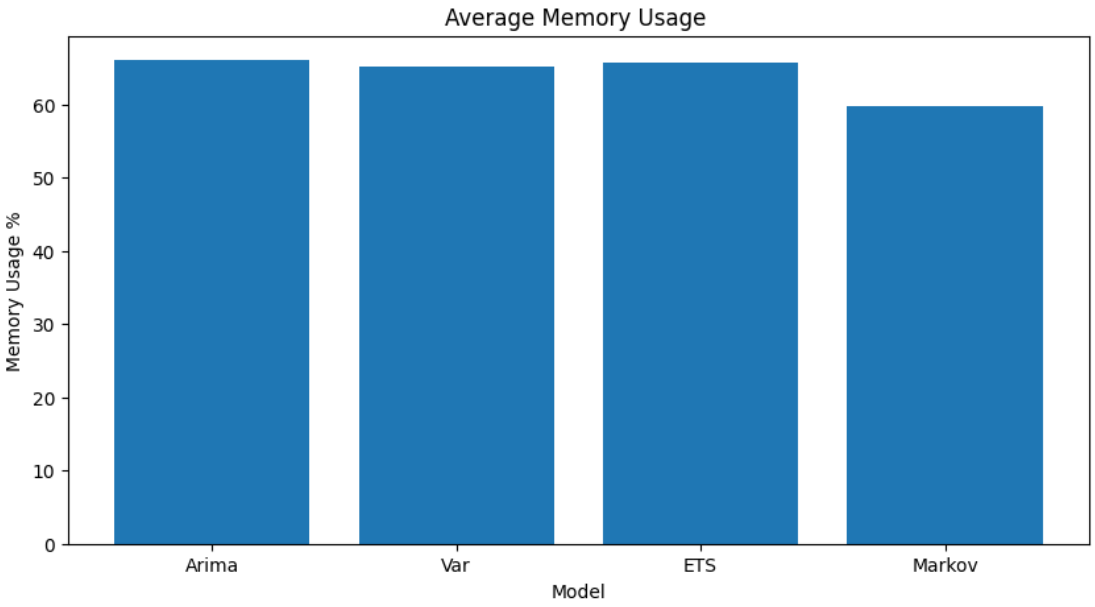

Figure 6: Average Memory Usage Consumed by Prediction Models

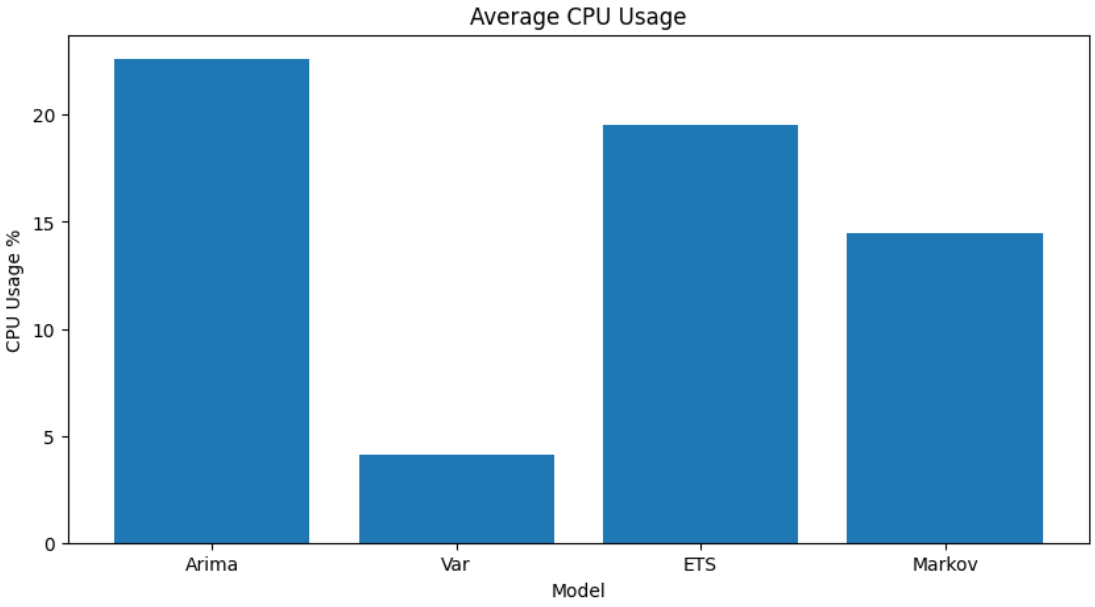

Figure 7: Average CPU Usage Consumed by Prediction Models

4 Evaluation

4.1 Reaserch Tools Used

To conduct this research, a combination of virtualization, monitoring, and benchmarking tools was employed to simulate and analyze workload behavior.

QEMU: An open-source virtualization tool used to create and manage virtual machines (VMs). QEMU enabled flexible testing and migration of VMs under various workloads, which was essential for workload prediction experiments.

RealVNC: This remote access software facilitated real-time monitoring and control of VM performance during testing. RealVNC allowed for seamless interaction with VMs, enabling observation and data collection for workload analysis.

NFS (Network File System): NFS was utilized to set up shared storage across networked environments. This system facilitated resource sharing among VMs, which was necessary for testing workload migration and replication scenarios.

YCSB (Yahoo! Cloud Serving Benchmark): This benchmark was used to generate persistent workloads. YCSB is a popular tool for testing storage systems, and it provides a reliable method for creating stable, long-running workloads to simulate database and application server tasks.

Stress Tool: The Stress tool was employed to create real-time workloads. This benchmark generates high CPU, memory, and I/O usage on demand, which is ideal for simulating the resource-intensive and bursty demands characteristic of real-time applications.

These tools collectively supported a comprehensive research setup, allowing for accurate workload simulation, monitoring, and analysis.

4.2 Evaluation Plan

1. Reducing Total Migration Time (TMT) and Downtime:
The primary objective is to achieve a reduction in total migration time (TMT) and downtime compared to the baseline provided by the vanilla Pre-Copy algorithm. This will be measured by conducting multiple migration trials and comparing the total migration time and downtime of the proposed method against those of the standard Pre-Copy approach.

2. Effect of the Look-Ahead Parameter:
Since the algorithm includes a look-ahead parameter (number of states considered in advance), this parameter's impact on migration time will be evaluated.

Different values of the look-ahead parameter will be tested to assess their effect on both prediction accuracy and migration performance. This allows for fine-tuning of the algorithm to optimize predictive accuracy and resource usage.

3. Initial Transition Matrix Configuration: The initial configuration of the transition matrix will be tested in two ways.

- **Equal Probability for All States:** Assigning equal transition probabilities across all states to create a neutral starting point.
- **Workload-Specific Transition Matrix:** Using a pre-defined matrix based on the characteristics of the current workload.

Performance in each scenario will be analyzed to determine the impact of initial matrix configuration on migration efficiency and predictive accuracy.

4.3 Project Timeline

The following chart displays the tentative timeline of the research.

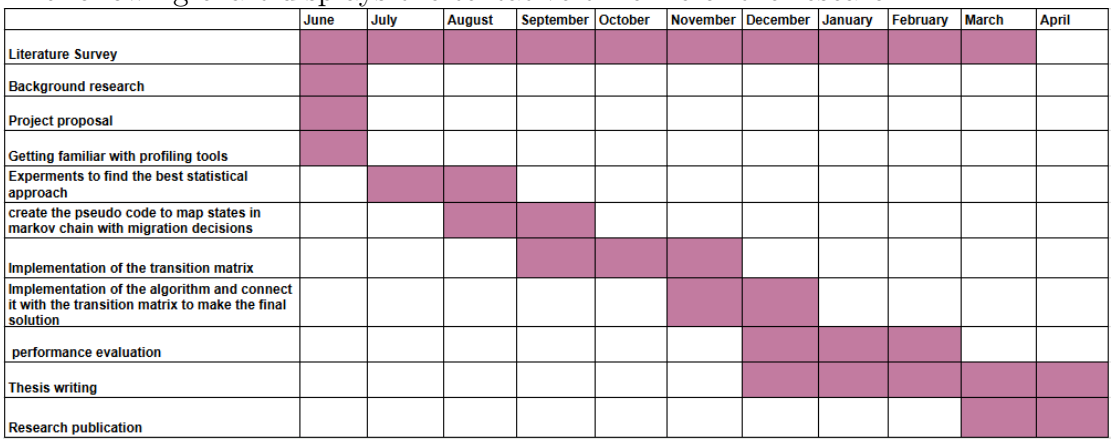

Figure 8: Project Timeline

4.4 Improvements From Feedback

- Following feedback from the proposal presentation, which recommended testing multiple types of dynamic workloads, the implementation began with two primary workload types: persistent workloads and real-time workloads. Persistent workloads, generated using the YCSB benchmark, simulate stable, long-running demands typical of database applications. Real-time workloads, created with the Stress benchmark, represent highly variable, latency-sensitive tasks.

Workload Type	**Examples**
Persistent Workloads	Database Management Systems (DBMS): MySQL, PostgreSQL, MongoDB Long-running Scientific Simulations: Weather forecasting, molecular simulations

Workload Type	Examples
Real-Time Workloads	Video Streaming: Netflix, YouTube Augmented Reality (AR) Applications: IKEA Place, other AR-based apps
Serverless Workloads	Event-Triggered Data Processing: AWS Lambda, Google Cloud Functions Notification Systems: Alerts on social media, e-commerce updates Image/Video Processing on Demand: Thumbnail generation, image processing
Interactive Workloads	Online Gaming: Fortnite, Call of Duty Collaborative Applications: Google Docs, Microsoft Teams, Slack
Burstable Workloads	E-commerce Websites: Amazon, Shopify Social Media Platforms: Twitter, Instagram

Table 3: Dynamic Workload Categories with Examples

References

Altahat, M. A., Agarwal, A., Goel, N. & Kozlowski, J. (2020), 'Dynamic hybrid-copy live virtual machine migration: Analysis and comparison', *Procedia Computer Science* **171**, 1459–1468. Third International Conference on Computing and Network Communications (CoCoNet'19).
URL: `https://www.sciencedirect.com/science/article/pii/S1877050920311352`

AWS (2024), 'Amazon web services'.
URL: `https://aws.amazon.com/`

Banerjee, P., Roy, S., Modibbo, U. M., Pandey, S. K., Chaudhary, P., Sinha, A. & Singh, N. K. (2023), 'Optidjs+: A next-generation enhanced dynamic johnson sequencing algorithm for efficient resource scheduling in distributed overloading within cloud computing environment', *Electronics* **12**(19), 4123.

Calheiros, R. N., Masoumi, E., Ranjan, R. & Buyya, R. (2014), 'Workload prediction using arima model and its impact on cloud applications' qos', *IEEE transactions on cloud computing* **3**(4), 449–458.

Cerotti, D., Gribaudo, M., Piazzolla, P. & Serazzi, G. (2012), Flexible cpu provisioning in clouds: A new source of performance unpredictability, *in* '2012 Ninth International Conference on Quantitative Evaluation of Systems', IEEE, pp. 230–237.

Choudhary, A., Govil, M. C., Singh, G., Awasthi, L. K., Pilli, E. S. & Kapil, D. (2017), 'A critical survey of live virtual machine migration techniques', *Journal of Cloud Computing* **6**(1), 1–41.

Christopher, C. (2005), Live migration of virtual machines, *in* 'NSDI, 2005'.

Devi, K. L. & Valli, S. (2023), 'Time series-based workload prediction using the statistical hybrid model for the cloud environment', *Computing* **105**(2), 353–374.

Fernando, D., Yang, P. & Lu, H. (2020), Sdn-based order-aware live migration of virtual machines, *in* 'IEEE INFOCOM 2020 - IEEE Conference on Computer Communications', pp. 1818–1827.

Gambs, S., Killijian, M.-O. & del Prado Cortez, M. N. (2012), Next place prediction using mobility markov chains, *in* 'Proceedings of the first workshop on measurement, privacy, and mobility', pp. 1–6.

Ganapathi, A., Chen, Y., Fox, A., Katz, R. & Patterson, D. (2010), Statistics-driven workload modeling for the cloud, *in* '2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)', pp. 87–92.

GCP (2024), 'Google cloud platform'.
URL: `https://cloud.google.com/`

Han, J., Hong, Y. & Kim, J. (2020), 'Refining microservices placement employing workload profiling over multiple kubernetes clusters', *IEEE access* **8**, 192543–192556.

Hashem, I. A. T., Anuar, N. B., Gani, A., Yaqoob, I., Xia, F. & Khan, S. U. (2016), 'Mapreduce: Review and open challenges', *Scientometrics* **109**, 389–422.

He, T. (2021), 'Migration management in software-defined networking-enabled edge and cloud computing environments', *degree of Doctor of Philosophy, School of Computing and Information Systems, THE UNIVERSITY OF MELBOURNE, ORCID: 0000-0002-5472-7681* .

Hines, M. R., Deshpande, U. & Gopalan, K. (2009), 'Post-copy live migration of virtual machines', *SIGOPS Oper. Syst. Rev.* **43**(3), 14–26.
URL: `https://doi.org/10.1145/1618525.1618528`

Hossain, M. A. & Song, B. (2016), 'Efficient resource management for cloud-enabled video surveillance over next generation network', *Mobile Networks and Applications* **21**, 806–821.

Khelghatdoust, M., Gramoli, V. & Sun, D. (2016), Glap: Distributed dynamic workload consolidation through gossip-based learning, *in* '2016 IEEE International Conference on Cluster Computing (CLUSTER)', IEEE, pp. 80–89.

Kim, I. K., Wang, W., Qi, Y. & Humphrey, M. (2018), Cloudinsight: Utilizing a council of experts to predict future cloud application workloads, *in* '2018 IEEE 11th international conference on cloud computing (CLOUD)', IEEE, pp. 41–48.

Lin, W., Wang, J. Z., Liang, C. & Qi, D. (2011), 'A threshold-based dynamic resource allocation scheme for cloud computing', *Procedia Engineering* **23**, 695–703.

Lu, H., Xu, C., Cheng, C., Kompella, R. & Xu, D. (2015), vhaul: Towards optimal scheduling of live multi-vm migration for multi-tier applications, *in* '2015 IEEE 8th International Conference on Cloud Computing', IEEE, pp. 453–460.

Malhotra, L., Agarwal, D., Jaiswal, A. et al. (2014), 'Virtualization in cloud computing', *J. Inform. Tech. Softw. Eng* **4**(2), 1–3.

Microsoft (2024), 'Microsoft azure'.
URL: `https://azure.microsoft.com/en-us`

Naik, K. J. (2022), 'An adaptive push-pull for disseminating dynamic workload and virtual machine live migration in cloud computing', *International Journal of Grid and High Performance Computing (IJGHPC)* **14**(1), 1–25.

Oracle (2024), 'Oracle — cloud applications and cloud platform'.
URL: `https://www.oracle.com/`

Pacheco-Sanchez, S., Casale, G., Scotney, B., McClean, S., Parr, G. & Dawson, S. (2011), Markovian workload characterization for qos prediction in the cloud, *in*

'2011 IEEE 4th International Conference on Cloud Computing', IEEE, pp. 147–154.

Sah, S. K. & Joshi, S. R. (2014), Scalability of efficient and dynamic workload distribution in autonomic cloud computing, *in* '2014 international conference on issues and challenges in intelligent computing techniques (ICICT)', IEEE, pp. 12–18.

Sahni, S. & Varma, V. (2012), A hybrid approach to live migration of virtual machines, *in* '2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)', pp. 1–5.

Wei, Y., Kudenko, D., Liu, S., Pan, L., Wu, L. & Meng, X. (2018), A reinforcement learning based workflow application scheduling approach in dynamic cloud environment, *in* 'Collaborative Computing: Networking, Applications and Worksharing: 13th International Conference, CollaborateCom 2017, Edinburgh, UK, December 11–13, 2017, Proceedings 13', Springer, pp. 120–131.

Wu, Q. & Wolf, T. (2008), Dynamic workload profiling and task allocation in packet processing systems, *in* '2008 International Conference on High Performance Switching and Routing', IEEE, pp. 123–130.

Xing, Y. & Zhan, Y. (2012), Virtualization and cloud computing, *in* 'Future Wireless Networks and Information Systems: Volume 1', Springer, pp. 305–312.

YCSB (2024), 'Ycsb: Yahoo! cloud serving benchmark'.
URL: `https://github.com/brianfrankcooper/YCSB`

Ye, K., Wu, Z., Wang, C., Zhou, B. B., Si, W., Jiang, X. & Zomaya, A. Y. (2014), 'Profiling-based workload consolidation and migration in virtualized data centers', *IEEE Transactions on Parallel and Distributed Systems* **26**(3), 878–890.

Zhang, Q. & Boutaba, R. (2014), Dynamic workload management in heterogeneous cloud computing environments, *in* '2014 IEEE Network Operations and Management Symposium (NOMS)', IEEE, pp. 1–7.