Traffic-Aware Live Virtual Machine Migration: A Priority and SLA-Sensitive Approach

H.B.V.D.Dayaratne

Traffic-Aware Live Virtual Machine Migration: A Priority and SLA-Sensitive Approach

H.B.V.D.Dayaratne

Index No : **20000286**

Supervisor: **Dr. D.K.Fernando**

April 2025

Submitted in partial fulfillment of the requirements of the

B.Sc in Computer Science Final Year Project (SCS4224)

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: H.B.V.D. Dayaratne

———————————————

Signature of Candidate

Date:

This is to certify that this dissertation is based on the work of Ms. H.B.V.D. Dayaratne under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Supervisor's Name: Dr. D.K. Fernando

———————————————

Signature of Supervisor

Date:

Abstract

Live migration of virtual machines (VMs) is critical in cloud environments for maintaining service continuity during load balancing, fault recovery, and infrastructure maintenance. However, traditional techniques such as pre-copy, post-copy, and hybrid migration overlook the dynamic interplay between migration traffic and application workloads, often resulting in increased downtime and degraded performance due to network contention. This paper proposes a Traffic-Aware Live VM Migration algorithm that intelligently selects the optimal migration strategy and performs adaptive bandwidth reservation based on real-time traffic conditions, migration urgency, and Service Level Agreement (SLA) constraints.

The proposed approach classifies network and workload traffic patterns to dynamically decide between pre-copy and post-copy techniques, and prioritizes migration tasks according to SLA sensitivity. It also distinguishes between high, medium, and low urgency migrations, adapting the strategy to either parallel or serial migration accordingly. Extensive empirical evaluation demonstrates that the traffic-aware algorithm significantly reduces total migration time for high-priority migrations and minimizes downtime for low-priority ones, outperforming traditional methods across a variety of traffic scenarios. This work highlights the importance of integrating traffic shaping, SLA compliance, and priority-awareness to achieve efficient, non-disruptive VM migration in modern data centers.

Acknowledgment

I am eternally grateful to my supervisor, Dr. Dinuni Fernando, for her invaluable guidance and unwavering support throughout this research journey. From the very beginning, helping us navigate the literature review to being there whenever we reached out for help, her mentorship has been a constant source of encouragement. Her regular check-ins ensured that we never felt stuck or lost along the way, and her dedication made a tremendous difference to the success of this work.

I would also like to sincerely thank the Network Operating Centre (NOC) for providing the necessary resources and technical support. My heartfelt appreciation goes to my fellow batchmates of the CloudNet Research Group, who readily offered their help and advice whenever I needed it, creating a truly supportive environment.

Finally, I am deeply grateful to my friends and family for their constant encouragement, patience, and belief in me throughout this journey.

Contents

List of Figures

List of Tables

Abbreviations

CDC Cloud Data Center. 9, 10

DT Downtime. 5

LAN Local Area Network. 1, 19

NAS Network Attached Storage. 1

NFS Network File System. 19

NIC Network Interface Card. 10

NTVMM Network Traffic-based Virtual Machine Migration algorithm. 15

OS Operating System. 1

QoS Quality of Service. 4, 5, 10

SLA Service Level Agreement. 4, 5, 10, 26

SLAs Service Level Agreements. 4

TMT Total Migration Time. 5

VM Virtual Machine. 1–6, 10, 13, 15–17, 19, 20, 26, 27, 31, 35, 37

VMs Virtual Machines. 1, 3–7, 9, 12, 15, 16, 20, 26, 29, 33

WAN Wide Area Network. 1

WWS Writable Working Set. 2, 5

1   Introduction

In today's digital landscape, the need for on-demand and scalable computing resources has driven the adoption of cloud computing. Virtualization enables the creation of virtual images of physical resources, allowing multiple machines to access them simultaneously. A Virtual Machine (VM) emulates a physical machine with its own Operating System (OS), CPU, memory, and functions. A physical machine serves as a server hosting one or many VMs. These VMs can be relocated, a process known as VM migration (Bahrami, Haghighat, and Gholipoor n.d.).

Migrating a VM instantaneously while it is running is called *Live Migration*. Migration of multiple VMs can be done parallely, which is referred to as *Gang Live Migration* (Deshpande, Wang, and Gopalan 2011), or sequentially, *Serial Live Migration*. This process typically involves transferring the VM's memory, disk storage, and network connections from the source host to the destination (Le 2020). Within a Local Area Network (LAN), VM migrations do not require the migration of disk storage due to the use of Network Attached Storage (NAS) (Jo et al. 2013). However, migrations over a Wide Area Network (WAN) necessitate the transfer of both the disk state and memory (Wood, Ramakrishnan, et al. 2014).

There are three (3) traditional live migration techniques namely pre-copy (Clark et al. 2005; Nelson, Lim, Hutchins, et al. 2005), post-copy (Hines and Gopalan 2009; Hines, Deshpande, and Gopalan n.d.), and hybrid-copy (Sahni and Varma 2012). These techniques are designed for single VM migrations. However, when adapting these techniques for multiple VM migrations, if all the VMs monotonously adopt a single migration algorithm, it overlooks the traffic contention. The traffic generated due to the VM migration adds to the existing application traffic sharing the same bandwidth leading to inefficiencies.

1.1 Background

1.1.1 Live VM Migration

Live migration involves transferring the memory pages and disk blocks associated with a VM from a source host to a destination host while the VM continues to operate (Clark et al. 2005). This process requires preserving the VM's runtime state, including CPU registers, disk I/O state, and memory contents, ensuring a seamless transition with minimal service interruption.

1.1.1.1 Pre-copy Migration Technique

The pre-copy migration technique initiates while the VM is still executing on the source host. During this phase, memory pages are iteratively copied to the destination host. A key indicator used to determine when to halt this phase is the identification of a Writable Working Set (WWS), pages that are frequently updated during migration (Hines, Deshpande, and Gopalan n.d.). Once a threshold is reached, the *stop-and-copy* phase begins. At this point, the VM is momentarily paused to transfer the remaining dirty pages and the CPU state. Following the transfer, network connections are redirected to the destination host, completing the migration process.

1.1.1.2 Post-copy Migration Technique

In the post-copy migration approach, the VM is first paused briefly to transfer non-pageable memory, essential CPU state, and network configuration to the destination. The VM is then resumed at the destination host. After resumption, the remaining memory pages are demand-paged over the network from the source as they are accessed (Bahrami, Haghighat, and Gholipoor n.d.).

1.1.1.3 Hybrid-copy Migration Technique

The hybrid-copy migration technique combines elements from both pre-copy and post-copy methods to balance performance and downtime. Initially, memory pages

are pre-copied while the VM is active on the source host, similar to the pre-copy strategy. The VM is then resumed on the destination host. Subsequent page faults are resolved by fetching the corresponding pages from the source, akin to the post-copy approach (Sahni and Varma 2012). This hybrid method aims to reduce the downtime associated with pre-copy and mitigate the page fault latency challenges of post-copy migration.

1.1.2 Multiple VM Migrations

Virtual machines that reside on the same physical host are referred to as co-located VMs. These VMs often interact with one another, complementing each other's services. Running on the same physical machine reduces communication costs (Huang et al. 2007) and enhances resource consolidation (Wood, Shenoy, et al. 2007). However, there are instances where migrating multiple correlated VMs becomes necessary for purposes such as energy efficiency and performance optimization (Sun, Liao, Zhao, et al. 2015). These migrations can be conducted either serially or in parallel (Callegati and Cerroni 2013; Chang, Walters, and Wills 2013; Sun, Liao, Anand, et al. 2016).

In the *serial migration* strategy, each VM is migrated sequentially using the migration techniques designed for single VM migration. Alternatively, the *gang migration* strategy enables parallel migration of VMs, where the available bandwidth is shared across the multiple VMs.

Although gang migration results in a longer total migration time compared to serial migration, it offers the advantage of reduced downtime, making it more suitable for scenarios where service availability is critical. In contrast, serial migration, while more efficient in terms of communication resources and transmission overhead, may result in longer service downtimes (Callegati and Cerroni 2013). This creates a trade-off: gang migration prioritizes minimizing downtime to maintain service continuity, while serial migration focuses on optimizing resource usage and reducing transmission overhead. The choice between these strategies depends on the specific requirements of the migration task.

1.1.3 Migration Priority

Migration priority defines the urgency with which VMs need to be migrated. High-priority migrations often necessitate the use of gang migration to minimize downtime, despite its potential performance impact. In contrast, lower-priority migrations may be better suited to the serial migration approach, which allows for a more controlled and gradual migration process. System administrators can specify migration priorities (Fernando, P. Yang, and H. Lu 2020), or priority may be dynamically assigned based on factors such as Service Level Agreement (SLA) violations and migration time constraints (Tsakalozos et al. 2017).

Different migrations have varying time requirements and deadlines, particularly when dealing with multiple VMs (Nadeem, Elazhary, and Fadel 2018). For instance, some virtual network functions (VNFs) require rapid migration to preserve low end-to-end latency, while web services with higher latency tolerance can afford to be migrated over a longer duration. For example, in the event of a failure, the migration of a VM must occur as quickly as possible, whereas routine maintenance can be scheduled more gradually, with a focus on minimizing service disruption. The considerations of Service Level Agreement (SLA)s and Quality of Service (QoS) play a critical role in determining the appropriate migration strategy, serial or parallel, and in deciding the bandwidth allocation for the migration. Depending on the specific use case, migration tasks may have strict deadlines, requiring a balance between priority levels and performance metrics.

1.1.4 Service Level Agreements (SLAs)

Service Level Agreements (SLAs) are formal contracts established between cloud service providers and their customers, defining the expected levels of service delivery, including performance, availability, and redundancy standards (K. Lu et al. 2013). These agreements outline the obligations of both parties and describe the penalties for any failure to meet the agreed-upon terms (Odun-Ayo, Udemezue, and Kilanko 2019). Adherence to metrics defined by SLAs such as response time, availability, and reliability is critical (Gao et al. 2014). Providers must avoid SLA

violations and ensure customer satisfaction while reducing resource usage and service interruptions.

Quality of Service (QoS) represents the measurable performance characteristics guaranteed by service providers to meet customer expectations (K. Lu et al. 2013). By ensuring consistent QoS levels, service providers can meet their SLA obligations while optimizing network and resource utilization during the migration process.

1.1.5 Performance Metrics

Performance metrics serve as an essential evaluation to assess the live VM migration methods. Among the numerous metrics identified, total migration time, downtime, total migrated data, migration overhead, application degradation, preparation time, and resume time certain metrics stand out as most commonly used (Voorsluys et al. 2009; Altahat et al. 2020; Soni and Kalra 2013; Hines and Gopalan 2009; Kuno, Nii, and Yamaguchi 2011). Specifically,

- Total Migration Time (TMT) - the time required to complete the entire migration process. In gang migration, it's the time between the beginning of the migration process at the source to the completion of the last VM's migration at the destination.

- Downtime (DT) - the duration during which the services provided by the VMs become inaccessible to users. The VM's CPU state and the WWS are transferred during this time and therefore the CPU execution is fully suspended.

- Application performance degradation - the slowdown of the applications running on the migrating VMs during the migration.

1.1.6 Traffic Contention Problem

In the context of migration, traffic can be classified between workload traffic and migration traffic. As for workload traffic, it includes workloads of the migrating

VM, other co-located VMs or business traffic generated by applications that reside in the VM or the host.

1.1.6.1   Migration traffic

- Pre-copy Migration iteratively transfers the state data (CPU, memory, and I/O state) of a VM from the source host to the destination host while the VM continues to run on the source. The traffic is primarily outbound, as data flows from the source to the destination (shown in Figure 1a).

- In Post-copy Migration, the VM is immediately moved to the destination host, and any missing data pages are fetched from the source as needed. The traffic is primarily inbound, with data being requested from the source to the destination (shown in Figure 1b).

1.1.6.2   Application Traffic

- **Incoming traffic:** Refers to the data packets received by a system or network interface. In the context of virtual machines (VMs) and applications, incoming traffic typically includes requests, data, or communications initiated by external entities (Cui, Zhu, et al. 2020). This traffic is critical for the operation of client-server applications and services hosted within the VMs, as it represents the demand side of the service interactions.

- **Outgoing traffic:** Refers to the data packets transmitted from a system or network interface. For VMs and applications, outgoing traffic includes responses, data, or communications sent from the VMs to external entities. Outgoing traffic is essential for delivering services and content from the VMs to end users, forming the supply side of the service interactions (Deshpande and Keahey 2017).

1.1.6.3   Traffic Contention Problem

Both migration traffic and application workload traffic share the same network bandwidth (illustrated in Figure 1). When these traffic types occur simultane-

(a) Pre-Copy   (b) Post-Copy
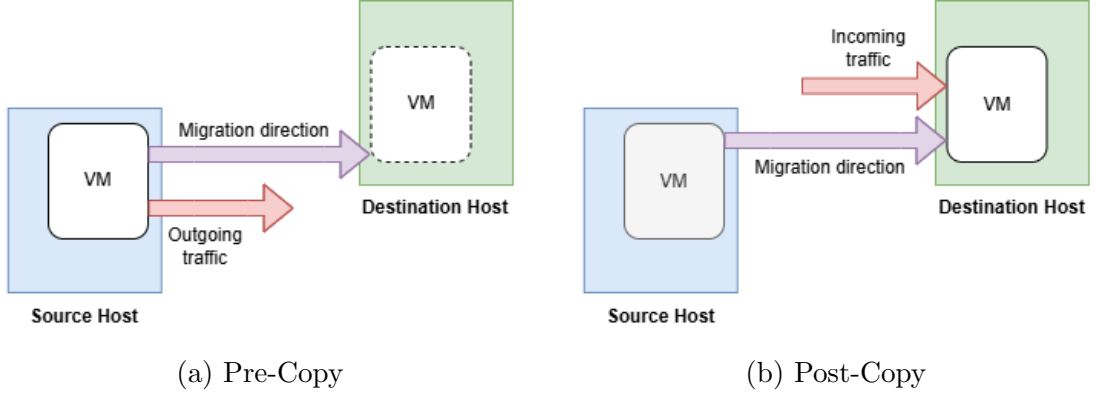
Figure 1: Traffic directions in Pre-Copy and Post-Copy migrations.
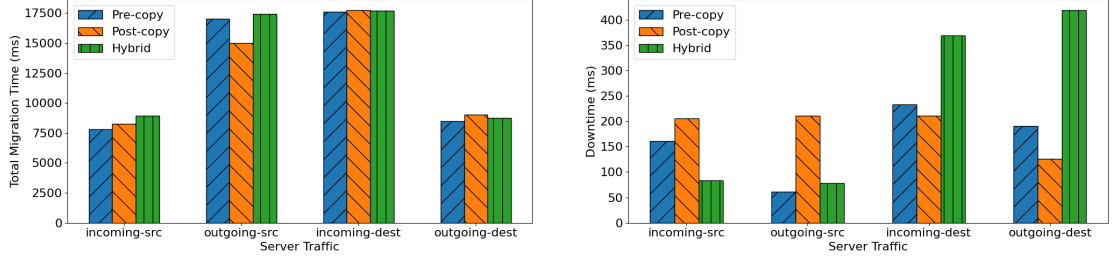
Figure 2: Total migration time and downtime with different traffic types.

ously, they compete for the limited available network resources (Deshpande and Keahey 2017). In the pre-copy migration approach, the outbound migration traffic competes with the outbound application traffic originating from VMs on the source host (depicted in Figure 3a). This contention can degrade both the migration process and the performance of the applications running on the VMs (illustrated in Figure 2). In the post-copy migration method, the inbound migration traffic at the destination host competes with incoming application traffic to the VMs (depicted in Figure 3d), leading to similar performance degradation and an increase in migration time (Cui, Zhu, et al. 2020).

Extended migration durations due to network contention can delay the completion of the migration process, thereby postponing the release of resources on the source host. In gang migration scenarios, where multiple VMs are migrated concurrently, the combined migration traffic exacerbates the issue, further stressing the available bandwidth and worsening the contention problem.

Network-bound applications are particularly affected by this issue, as the re-

(a) Outgoing traffic at the source host

(b) Incoming traffic at the source host

(c) Outgoing traffic at the destination host

(d) Incoming traffic at the destination host
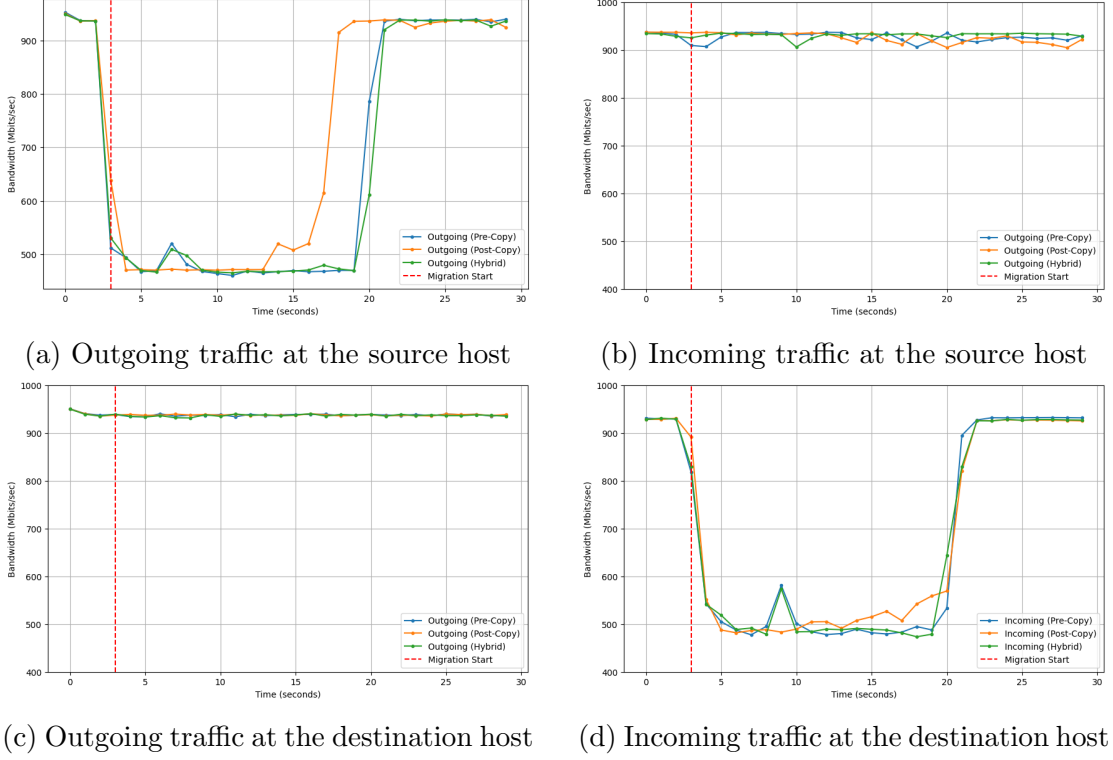
Figure 3: Bandwidth fluctuations of live migration with incoming and outgoing traffic.

duced available bandwidth leads to slower response times and may negatively impact the quality of service. Both client-server applications and other network-intensive applications running on the same host can experience reduced throughput and increased latency as a result of this directional traffic contention.

1.1.7 Traffic Shaping

Traffic shaping and bandwidth reservation are critical mechanisms for optimizing network performance. Effective management of network traffic can substantially alleviate congestion and enhance overall efficiency, particularly in environments where multiple applications or services share limited network resources. In such scenarios, it is essential to ensure that specific traffic types do not exceed the available network capacity or interfere with the proper functioning of other ongoing processes.

One of the most widely used tools for traffic control in Linux environments is

`tc` (Traffic Control) (*Linux tc* n.d.), a command-line utility that enables administrators to regulate the flow of network traffic. `tc` allows for the implementation of traffic shaping by controlling bandwidth allocation, delay, packet loss, and priority. This ensures that different traffic streams are allocated the necessary resources while preventing network congestion or the unfair distribution of resources. By defining classes and queues within `tc`, users can employ advanced traffic management strategies such as rate limiting, prioritization, and traffic classification based on criteria such as IP addresses, ports, and protocols.

In a typical network configuration, `tc` can be utilized to differentiate traffic types, ensuring that high-priority traffic, such as voice or real-time data, receives the necessary bandwidth without negatively affecting the performance of other services, such as file transfers or background tasks. For instance, network interfaces can be configured with traffic shaping policies that allocate bandwidth in accordance with the specific needs of each traffic type, while also considering the current network load.

Research has demonstrated that the combination of traffic shaping and intelligent bandwidth reservation techniques can significantly enhance network performance. By dynamically adjusting available bandwidth based on traffic characteristics, workload urgency, and network conditions, these strategies help mitigate network contention and improve overall efficiency. Various studies have explored adaptive bandwidth adjustments, highlighting the effectiveness of traffic shaping in reducing congestion and improving the quality of service across diverse network environments.

1.2   Motivation

The live migration of VMs within a Cloud Data Center (CDC) is essential for various operational reasons, including load balancing (Padala et al. 2007; Wood, Shenoy, et al. 2007), hardware failure management (Nagarajan et al. 2007), necessary hardware maintenance (Devi, Aruna, Priya, et al. 2011), and energy or power savings through workload consolidation (Nathuji and Schwan 2007; Hu et al. 2008).

In each of these scenarios, it is crucial that migrations are executed with minimal delay to prevent disruptions in service availability and to avoid performance degradation.

When VM migration shares the same Network Interface Card (NIC) interface as the VM's regular workload, network contention arises as migration traffic and workload traffic compete for bandwidth. This contention becomes particularly problematic in live migration techniques such as pre-copy and post-copy. In pre-copy migration, the outgoing network traffic from the source host competes with the migration traffic, while in post-copy migration, the incoming network traffic at the destination host contends with the migration stream (Deshpande and Keahey 2017). Such contention negatively impacts the performance of applications running on the migrating VM, slows down the migration process, and extends downtime. The increasing volume of network traffic within CDCs, driven by the rapid growth and evolution of cloud services, exacerbates these challenges, further complicating the migration process.

Furthermore, multiple VM migrations present complex trade-offs between total migration time and downtime, influenced by the migration strategy adopted. In serial migration, which involves migrating VMs sequentially, the overall migration time tends to be shorter, owing to reduced communication overhead and resource contention. However, in parallel migration, where multiple VMs are migrated simultaneously, downtime is minimized, leading to better service availability but at the cost of higher resource usage and longer total migration times (Callegati and Cerroni 2013). The choice between serial and parallel migration is therefore critical, especially in contexts where stringent deadlines, SLA compliance, and QoS requirements must be met (Nadeem, Elazhary, and Fadel 2018).

Addressing these challenges requires the development of a traffic-sensitive migration strategy that accounts for the contention between workload and migration traffic, as well as the dynamic trade-offs between migration time and downtime. By incorporating traffic shaping and prioritization mechanisms, such an approach can optimize the selection of migration techniques, ensuring a balance between

service availability and resource efficiency, while also meeting diverse constraints and deadlines.

1.3 Research Questions

1. How can traffic-sensitive live VM migration algorithms be developed to optimize migration performance by dynamically selecting the most appropriate migration technique (pre-copy, post-copy, or hybrid-copy) based on real-time network traffic metrics, available bandwidth, and traffic contention?

2. What is the impact of adaptive bandwidth reservation and prioritization strategies on migration efficiency, total migration time, and downtime during both single and multiple VM migrations under varying network traffic conditions?

3. How can traffic shaping and SLA compliance be integrated into live VM migration to ensure optimal resource allocation while minimizing migration delays and maintaining service level agreements in cloud environments?

1.4 Aims and Objectives

1.4.1 Aim

The aim of this research is to develop an adaptive, traffic-sensitive live migration algorithm for multiple virtual machines (VMs) that is aware of migration priorities, optimizes overall network performance, and effectively manages traffic contention. The algorithm should minimize migration time, downtime, and application performance degradation during the migration process.

1.4.2 Objectives

The key objectives of this research are:

- To identify and analyze the key network metrics that influence the performance of VM migration, with a focus on minimizing network congestion and

migration delays.

- To develop an adaptive decision-making algorithm that dynamically selects optimal migration strategies based on real-time network metrics, workload demands, and migration priorities.

- To investigate the impact of adaptive bandwidth reservation and traffic prioritization strategies on migration efficiency, including total migration time, downtime, and application performance, especially under varying network traffic conditions and traffic contention.

- To examine the integration of traffic shaping and SLA compliance in live VM migration, ensuring optimal resource allocation while minimizing migration delays and maintaining service level agreements (SLAs) in cloud environments.

1.5 Scope

1.5.1 In Scope

The scope of this research includes the following:

- Live Migration of Virtual Machines (VMs)

  - The focus will be on live migration within a KVM/QEMU virtualization environment.

  - The research will primarily address a single server with a single shared NIC port for both migration and application traffic. While some servers may employ separate NIC interfaces for application and migration traffic, which avoids traffic contention (Fernando, Terner, et al. 2019), such setups are less common and will not be the primary focus of this study.

  - The host operating system used for the servers will be Linux.

  - The study will cover the migration of a single VM from one host to another, as well as live migration of multiple co-located VMs.

– The tests will be conducted within local area network (LAN) environments, although the techniques developed may be adapted for wide-area network (WAN) scenarios, considering the additional complexities of disk migrations.

- Analysis of Traffic Effects on Live VM Migration

  – The research will focus on analyzing the effects of various traffic types (workload and migration traffic) on live VM migration, particularly network contention and its impact on migration efficiency.

- Development of an Optimized Algorithm

  – The research will develop an optimized algorithm for traffic-sensitive live migration of multiple co-located VMs, considering migration priorities, minimizing downtime, and reducing application performance degradation.

- Performance Evaluation

  – The algorithm will be evaluated based on industry-standard benchmarks, focusing on migration time, downtime, and application performance degradation, with a particular emphasis on traffic-sensitive scenarios.

1.6 Outline

The remainder of the thesis is structured as follows. Section 2 reviews the existing literature on live VM migration techniques, performance metrics, and traffic management strategies. Section 3 describes the preliminary study, detailing the methodology used for data collection and presenting the initial findings. Section 4 outlines the design and implementation of the proposed traffic-aware migration algorithm, emphasizing how migration strategies are adapted based on network metrics and priorities. Section 5 evaluates the performance of the developed algorithm under varying traffic conditions, comparing its performance with traditional

migration strategies using defined metrics. Section 6 discusses the behaviour of the algorithm in different scenarios, analyzes its performance, and explores potential improvements and directions for future research. Finally, Section 7 summarizes the findings of the thesis and proposes future research opportunities.

2   Literature Review

2.1   Traffic-Aware Migration

Shrivastava et al. 2011 introduced AppAware, a live migration approach that optimizes migration destinations by considering network topology, communication patterns of VMs, and physical server limitations, focusing on application-specific communication needs. Similarly, Cui, Z. Yang, et al. 2017 proposed a topology-adaptive Data Center Network (DCN) that constructs and adjusts network topologies dynamically based on VM demands and traffic patterns. Tso et al. 2014 developed SCORE, an approach designed to minimize overall inter-VM communication by treating communication as an optimization problem, using a distributed migration technique that adapts to traffic fluctuations. Collectively, these approaches aim to optimize network topology and communication patterns to reduce network traffic during migration.

Other strategies extend traffic-awareness to load balancing and resource monitoring. For example, Kanniga Devi, Murugaboopathi, and Muthukannan 2018 proposed the System and Traffic-Aware Live VM Migration for Load Balancing (ST-LVM-LB), a graph-based approach that monitors resources and balances load by migrating the least-loaded VMs while accounting for network bandwidth and active traffic flows. Fu et al. 2019 introduced NTVMM, which reduces traffic and network load through algorithms that prioritize high-traffic-generating VMs during selection and placement. Addressing network congestion, Liu et al. 2015 formulated cross-site live migration of multiple VMs as a Mixed-Integer Linear Programming (MILP) problem, factoring in migration traffic and inter-VM communication, and developed the *MinUti-O* algorithm to mitigate complexity. Furthermore, Nasim and Kassler 2015 proposed using multipath-TCP and queue management strategies to enhance traffic-aware live migration, reducing both downtime and migration duration. These approaches emphasize load balancing, resource management, and congestion mitigation through various optimization methods.

Additional traffic-aware solutions have targeted container and VM migration

optimization. Maheshwari et al. 2018 introduced ShareOn, a traffic-aware container migration algorithm that selects target hosts based on real-time traffic conditions to minimize network impact. Deshpande and Keahey 2017 proposed a traffic-aware VM live migration algorithm that monitors both application and migration traffic to determine the optimal use of pre-copy or post-copy techniques. Cui, Zhu, et al. 2020 developed an adaptive migration algorithm selection framework using fuzzy clustering to categorize VMs by business traffic, enabling dynamic selection of the most suitable migration approach. These methodologies adapt migration strategies based on current traffic conditions, optimizing migration performance through adaptive mechanisms.

2.2 Priority-Aware Migration

There exists research that focus on migration priority as well. Fernando, P. Yang, and H. Lu 2020 introduced a live migration approach that reserves bandwidth according to the urgency of migration. Nadeem, Elazhary, and Fadel 2018 proposed a prioritization-based approach where only low-priority tasks are selected for migration, retaining high-priority ones at the host. Dalvandi, Gurusamy, and Chua 2015 presented an algorithm that considers the requested migration time of VMs for optimal placement and routing. Haidri et al. 2022 explored migration strategies that consider request deadlines to balance load and meet specific timing requirements, while Son and Buyya 2018 focused on VM/flow placement based on migration deadlines.

2.3 Traffic Shaping

The use of traffic shaping to optimize VM migrations has been well-documented. The mVM Scheduler, for instance, uses a network model that continuously monitors traffic patterns and network capacity to determine optimal migration start times and bandwidth allocation, dynamically adjusting between parallel and serial migrations based on real-time conditions (Kherbache, Madelaine, and Hermenier 2017). Similarly, Software-Defined Networking (SDN) controllers offer real-time

traffic flow management, bandwidth allocation, and route adjustments to prevent link congestion. Network Function Virtualization (NFV) further enhances traffic optimization by prioritizing critical migration flows with virtualized middleboxes (Manzalini et al. 2013).

The Geometric Programming Model offers another perspective by adapting bandwidth to different phases of migration, using a cost function to balance downtime and resource usage, ultimately minimizing network strain and migration time. This model's rapid convergence to optimal values makes it suitable for real-time, multiple VM migration scenarios (Cerroni and Esposito 2016). Pre-copy migration techniques, which progressively increase transfer rates across rounds to minimize residual data, exemplify adaptive bandwidth adjustments that enhance migration efficiency (Choudhary et al. 2017).

Ongoing research in traffic-sensitive live VM migration emphasizes a range of approaches, from selecting migration techniques based on traffic patterns to reducing overall traffic and optimizing migration sequences. Traffic-aware solutions, adaptive topologies, and urgency-based strategies highlight advancements in enhancing migration efficiency. Despite these approaches, integrating traffic and urgency considerations to maximize network performance remains an area for exploration and further improvement.

2.4 Research Gap

Currently, existing algorithms either focus on traffic contention or prioritize migration urgency, but there is a lack of algorithms that effectively integrate both factors. A dynamic approach that considers both traffic contention and priority levels could greatly enhance migration performance by selecting the most appropriate migration technique, whether pre-copy, post-copy, or hybrid-copy, based on real-time network conditions and the urgency of the migration. This would enable the handling of different tasks with varying requirements, such as those needing minimal migration time, those focused on reducing downtime, and those willing to trade off between migration time and downtime to optimize application

17

performance.

Moreover, there is a gap in comprehensive studies that explore the decision-making process of selecting serial versus parallel migrations. Although both strategies offer distinct advantages, such as reduced migration time in serial migration and minimized downtime in parallel migration, there is limited research on how to choose between them based on real-time conditions. This research aims to fill these gaps by developing algorithms that take into account both network traffic and urgency, enabling more efficient and adaptive migration strategies that optimize migration time, downtime, and application performance.

3 Preliminary Study

3.1 Data Collection

This section discusses how the data were collected for the preliminary study, testing and evaluation purposes.
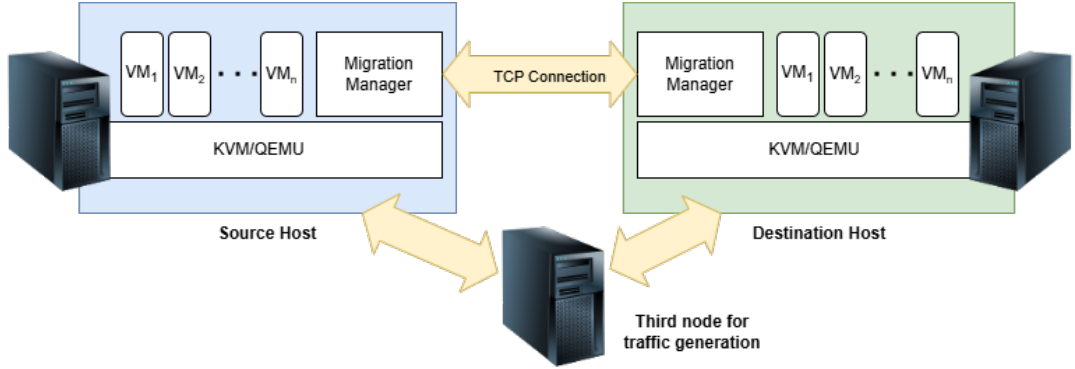
Figure 4: Testbed architecture.

We conducted experiments using two (2) machines configured as source and destination hosts (as shown in Figure 4), with the following specifications:

- Product - HP Z620 Workstation

- CPU - Intel(R) Xeon(R) CPU E5-1650 v2 @ 3.50GHz (12 Cores)

- Memory - 16GB

- Network - Gigabit Ethernet Switch

An Network File System (NFS) server was employed, sharing a directory between the source and destination servers. The VM hard disks were stored on this NFS server, meaning they were not migrated during the LAN migrations. The baseline models were implemented using QEMU Emulator version 8.1.2 installed on both the source and destination servers.

Single VM migrations were carried out using pre-copy, post-copy, and hybrid-copy techniques under a range of traffic conditions to validate the presence of traffic contention issues. Each migration involved an idle VM configured

with 8GB of RAM. Various combinations of outgoing and incoming traffic were generated at both the source and destination hosts to simulate realistic network scenarios. For each configuration, the migration was repeated ten (10) times, and the average values were computed to ensure consistency, excluding any anomalies. As shown in Figure 2, higher migration times were observed, particularly when outgoing traffic was present at the source and incoming traffic at the destination. These increases are attributed to contention between the migration traffic and the background network traffic, which adversely affects the efficiency of data transfer during migration.

Multiple VM migration experiments were conducted to evaluate the performance differences between serial and parallel migration approaches under varying traffic conditions. Two (2) VMs were deployed on the source host. A separate server was configured to operate as the iperf server and/or client, depending on the specific traffic scenario being tested. Traffic was generated between this third server and the source, destination, and the VMs. Following traffic generation, the VMs were migrated to the destination host, during which the *total migration time* and *downtime* were measured. Results were averaged over five (5) runs, with outliers excluded from the analysis.

In parallel migration, where multiple VMs are migrated concurrently, the total migration time is determined by the longest individual migration time among all migrating VMs. Conversely, in serial migration, where VMs are migrated sequentially, the total migration time corresponds to the cumulative sum of each individual migration time.

Downtime is similarly affected by the migration approach. In serial migration, total downtime is the sum of the individual downtimes of all VMs, whereas in parallel migration, it is defined by the maximum downtime observed among the concurrently migrating VMs.

Figure 5 illustrates the results of the experiments. It was observed that in most cases, both total migration time and downtime were consistently lower in parallel migration compared to serial migration, for both pre-copy and post-copy
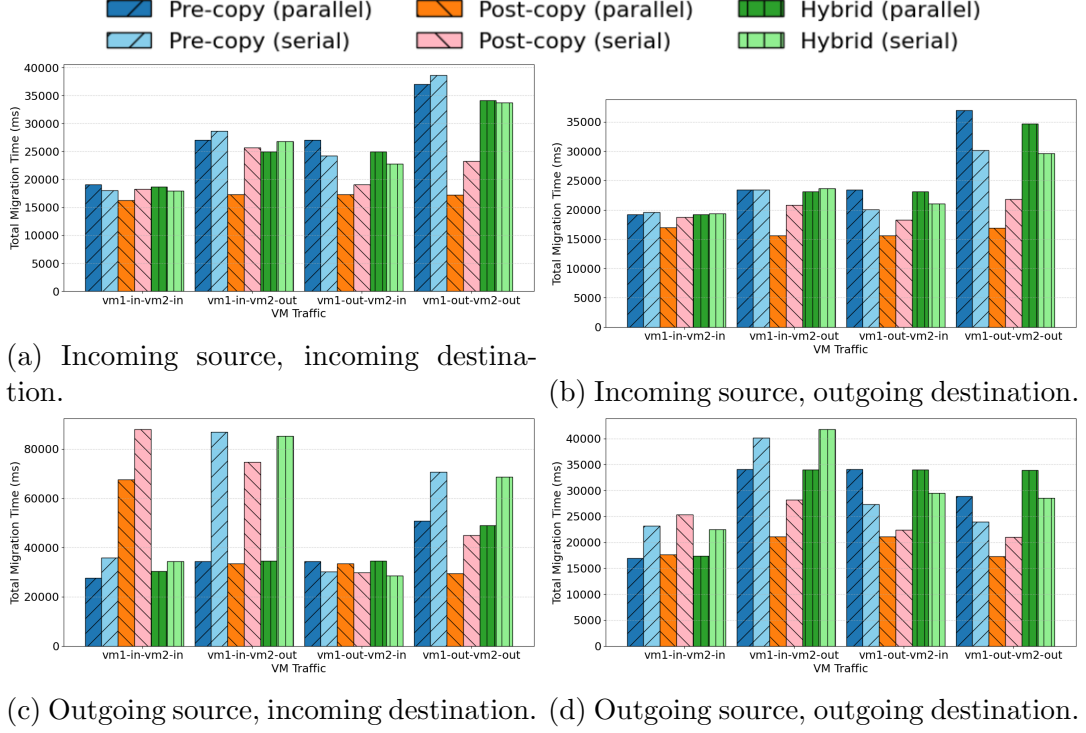
techniques.

(a) Incoming source, incoming destination.

(b) Incoming source, outgoing destination.

(c) Outgoing source, incoming destination. (d) Outgoing source, outgoing destination.

Figure 5: Total migration time of serial and parallel migrations using traditional migration techniques.

3.2   Migration Bandwidth Reservation

To identify the correct interface handling migration traffic, the command `ip route get  <DEST_IP>` was used. This revealed that traffic destined for the migration target (`DEST_IP`) is routed through the `br0` bridge interface, with the IP address of the source server (`SRC_IP`). To confirm this, `tcpdump` was used to monitor the interface with the command `tcpdump -i br0 host <DEST_IP> and <DEST_PORT>`, which verified that the migration data was indeed being transmitted over `br0` using the destination port (`DEST_PORT`). This port number is configured when starting the VM using QEMU. It is important to note that the source port of the migration traffic is an ephemeral port, which can vary between migration sessions and therefore cannot be reliably used to filter or limit traffic.

Once the migration path was confirmed, the Linux `tc` (Traffic Control) tool (*Linux tc* n.d.) was used to shape traffic. Migration traffic was identified based

21

on the combination of source IP, destination IP, and the fixed destination port (`DEST_PORT`). Using `tc`, packets matching these parameters were marked and assigned to a dedicated class with a predefined bandwidth limit, effectively isolating migration traffic from other application traffic. This setup ensured that bandwidth could be reserved specifically for migration without affecting the performance of concurrent workloads on the system. After reserving the required bandwidth for migration, the remaining available bandwidth on the interface could be assigned to handle all other traffic.

3.3 Empirical Bandwidth Models

To understand the effect of background traffic on migration bandwidth, we conducted controlled experiments using the three traditional migration techniques, pre-copy, post-copy, and hybrid-copy. In each case, the migration was performed while varying background traffic in one of three ways. Only outgoing traffic (from the VM), only incoming traffic (to the VM) and mixed traffic (both incoming and outgoing).

Each model (depicted in Figure 6) fits the form $y = mx + c$, where x is background traffic (in Mbps), and y is the estimated bandwidth required for migration.

Pre-copy Migration

- Incoming Traffic: $y = 903.47 + 0.0002x$

- Outgoing Traffic (Breakpoints: 0, 35, 472, 1000)

  S1: $y = -0.27x + 905.00$ S2: $y = -0.98x + 930.00$ S3: $y = -0.001x + 467.52$

- Mixed Traffic (Outgoing Varied, Incoming = 500 Mbps, Breakpoints: 0, 481, 1000)

$$\text{S1: } y = -0.91x + 906.80 \quad \text{S2: } y = -0.001x + 467.90$$

Post-copy Migration

- Incoming Traffic (Breakpoints: 0, 510, 1000)

$$\text{S1: } y = -1.02x + 861.00 \quad \text{S2: } y = -0.12x + 399.00$$

- Outgoing Traffic: $y = 848.10 + 0.0010x$

- Mixed Traffic (Incoming Varied, Outgoing = 500 Mbps, Breakpoints: 0, 528, 1000)

$$\text{S1: } y = -0.98x + 857.90 \quad \text{S2: } y = -0.09x + 386.40$$

Hybrid-copy Migration

- Incoming Traffic: $y = 896.82 + 0.0016x$

- Outgoing Traffic (Breakpoints: 0, 476, 1000)

$$\text{S1: } y = -0.90x + 897.66 \quad \text{S2: } y = -0.01x + 478.38$$

- Mixed Traffic (Outgoing Varied, Incoming = 500 Mbps, Breakpoints: 0, 485, 1000)

$$\text{S1: } y = -0.90x + 898.40 \quad \text{S2: } y = 0.003x + 462.86$$

These models provide valuable insight into how network traffic influences bandwidth demand for different techniques. They serve as a foundation for predicting migration feasibility in real time and ensuring SLA-compliant decisions.

(a) Pre-copy (In) (b) Pre-copy (Out) (c) Pre-copy (Mixed)

(d) Post-copy (In) (e) Post-copy (Out) (f) Post-copy (Mixed)

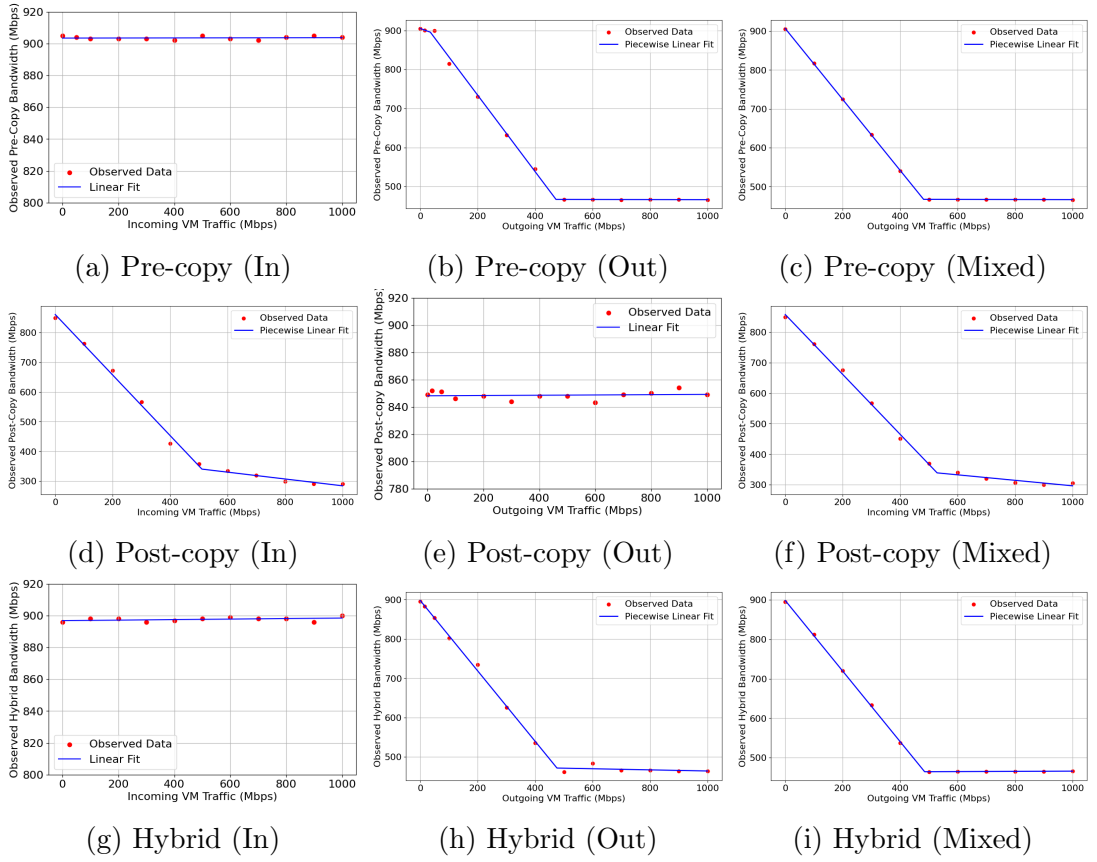(g) Hybrid (In) (h) Hybrid (Out) (i) Hybrid (Mixed)

Figure 6: Ideal traffic patterns for Pre-copy, Post-copy, and Hybrid under different traffic directions.

4   Design and Implementation

The proposed architecture for traffic-aware live VM migration, illustrated in Figure 7, is composed of three core modules that collectively aim to optimize the migration process by adapting to real-time network conditions. These modules are the *Network Tracker*, the *Bandwidth Reservation* module, and the *Migration Controller*.
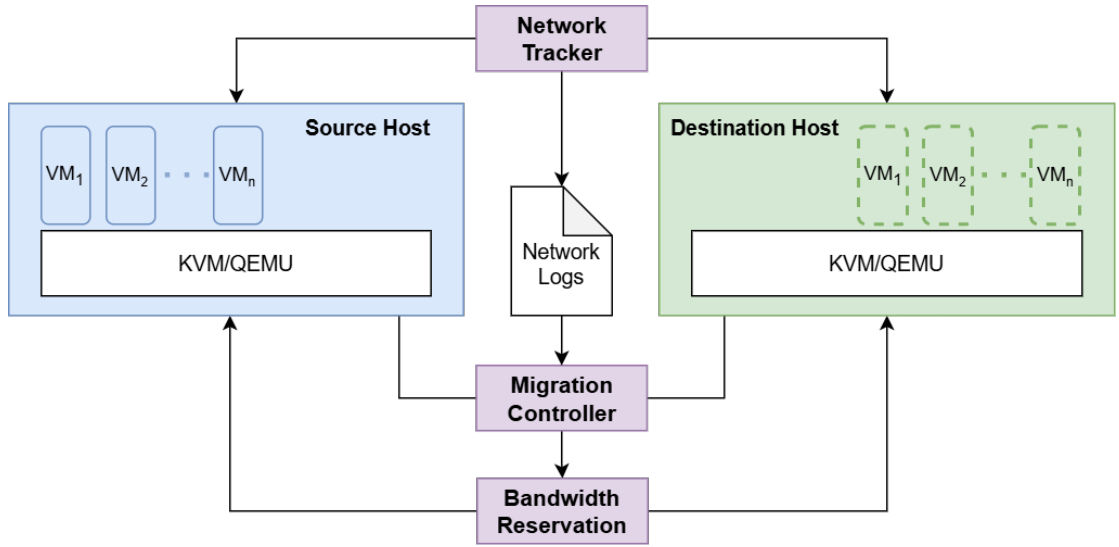
Figure 7: High-level architecture.

4.1   Network Tracker

This module is responsible for continuously monitoring network activity at both the source and destination servers. It tracks total traffic over the physical Ethernet interfaces, as well as per-VM traffic using tap interfaces. The data is periodically logged and stored on shared log files. Monitoring is conducted from both the source and destination ends. The implementation relies on the `ifstat` Linux command (*Linux ifstat* n.d.) for capturing traffic statistics.

4.2 Bandwidth Reservation

This module handles traffic shaping. It identifies migration traffic based on the migration port, source IP, and destination IP of the VM. A dedicated portion of bandwidth is allocated to migration traffic, depending on the selected migration technique and workload type. The remaining bandwidth is reserved for non-migration traffic to prevent interference. This module resides on the NFS server and is implemented using the `tc` utility as explained in the *Migration Bandwidth Reservation* section.

4.3 Migration Controller

The migration controller is responsible for making real-time migration decisions. Using network profiles from the source and destination servers, along with traffic characteristics of the involved VMs and their urgency levels, it selects an appropriate migration technique (pre-copy, post-copy, or hybrid). It also determines how bandwidth should be allocated. Once a decision is made, it triggers the bandwidth reservation module and initiates the migration process accordingly.

4.4 Traffic-Aware Migration Algorithm

Based on the preliminary study and its observations, the following algorithms were developed, initially for single VM migrations and later for multiple VM migrations.

4.4.1 Single VM Migrations

The proposed Algorithm 1 is designed to dynamically select the most suitable live migration technique for a single VM based on real-time network traffic conditions, while ensuring SLA compliance. It considers both the traffic characteristics of the source and destination hosts, as well as the workload behaviour of the migrating virtual machine.

Monitoring Network Parameters - The process begins by continuously monitoring key traffic parameters at the source, destination, and VM. These include

Algorithm 1 Traffic-Sensitive Single VM Migration Algorithm

1: Monitor Network Parameters: $src\_out, src\_in, dest\_out, dest\_in, vm\_out, vm\_in$
2: Compute available bandwidth at source: $src\_avail\_bw = tot\_bw - src\_out$
3: Compute available bandwidth at destination: $dest\_avail\_bw = tot\_bw - dest\_in$
4: Classify source and destination traffic types: $inbound, outbound, mixed, idle$
5: Classify workload type based on vm_out, vm_in
6: Retrieve SLA bounds: sla_upper, sla_lower
7: Determine migration technique:
8: **if** Source is outbound **then**
9:     Technique = Post-copy
10: **else if** Source is inbound **then**
11:     Technique = Pre-copy
12: **else if** Destination is idle **then**
13:     Technique = Post-copy
14: **else if** Source is idle **then**
15:     Technique = Pre-copy
16: **else**
17:     Technique = Post-copy
18: **end if**
19: Compute $ideal\_bw$ based on vm_out, vm_in and technique
20: **if** $avail\_bw \geq ideal\_bw$ **then**
21:     Migrate with available bandwidth
22: **else**
23:     Perform bandwidth reservation based on migration priority
24: **end if**
25: Perform migration

both incoming and outgoing bandwidth, which help capture the current load and potential contention on each network interface. The available bandwidth is then computed for the source and destination by subtracting observed traffic from the total available bandwidth, giving a realistic view of how much bandwidth can be safely allocated for migration.

Classification of Traffic Type and Workload Types - Next, the algorithm classifies the nature of network traffic on the source and destination into one of four types: idle, inbound, outbound, or mixed. The classification is based on the relative magnitudes of incoming and outgoing traffic rates.

1. Idle Traffic

    - Both incoming and outgoing traffic rate are less than 10.0 Mbps.

2. Inbound Traffic

    - The incoming rate is significantly greater than the outgoing rate, exceeding it by a factor defined by the threshold factor.

    - incoming_rate >THRESHOLD * outgoing_rate.

3. Outbound Traffic

    - The outgoing traffic rate is significantly greater than the incoming rate, exceeding it by the same threshold factor.

    - outgoing_rate >THRESHOLD * incoming_rate.

4. Mixed Traffic

    - Neither incoming nor outgoing traffic rate exceeds the other by the threshold factor.

    - The rates are within a range of approximately ±THRESHOLD factor

The threshold factor of 1.2 (representing a 20% difference between incoming and outgoing traffic rates) is chosen to meaningfully classify VM network traffic as inbound, outbound, or mixed. Research has shown that when bandwidth usage

reaches just 20%–30% of the available capacity, migration time can increase by nearly 49% highlighting how even moderate directional traffic can significantly degrade migration performance. By using this 20% margin, the algorithm can proactively identify when a VM exhibits directionally dominant behaviour. The threshold is also tuned to avoid false positives from natural traffic fluctuations while ensuring that genuine imbalances are not overlooked, thereby improving classification accuracy and enabling more efficient handling of network-intensive workloads during live migration.

Simultaneously, the workload type of the migrating VM is identified based on its traffic behaviour. This classification helps estimate how sensitive the VM is to bandwidth fluctuations and which migration technique will result in the least performance impact. The classification logic is derived from both internet sources and empirical analysis of workload behaviour. In our case, we rely on concrete traffic data that uniquely classifies VMs based on their inbound and outbound rates, allowing for a more precise mapping between observed traffic patterns and workload types.

Selection of Migration Technique - A rule-based decision system was implemented to determine the most appropriate migration technique based on network traffic conditions at the source and destination hosts.

- If the source is outbound or the destination is idle, the algorithm selects post-copy.

- If the source is inbound or idle, pre-copy is chosen.

- In all other cases, post-copy is used as the default due to its resilience to moderate traffic contention.

The hybrid technique was not considered in this decision process, as both pre-copy and post-copy consistently outperformed it when migrating network-intensive VMs.

Bandwidth Reservation - Once a migration technique is selected, the algorithm estimates the ideal bandwidth required for optimal migration (explained in Section

3.3). This estimation is based on empirical models developed through controlled experiments, which capture how migration bandwidth varies under different traffic scenarios for each migration technique.

Once the ideal bandwidth is estimated, the bandwidth reservation algorithm (presented in Algorithm 2) dynamically allocates bandwidth. The migration urgency levels determine how much bandwidth is reserved for the migration. In addition to the urgency level, the algorithm takes into account the available bandwidth and the VM's SLA requirements, which set upper and lower bounds for the migration bandwidth.

High Priority Migration - The full required bandwidth is reserved for the migration. This ensures that the migration process occurs as quickly as possible, without any delay, since high-priority migrations are critical and need to be completed with minimal interruption.

Medium Priority Migration - The bandwidth reserved is set to 75% of the ideal bandwidth. This allows for more flexibility in bandwidth allocation compared to high-priority migrations, as medium-priority migrations are less time-sensitive. The algorithm ensures that the bandwidth reserved does not exceed the available capacity, while also accounting for the lower bound of the VM's SLA.

Low Priority Migration - The ideal bandwidth is reduced to 50% of the required bandwidth. Low-priority migrations are less critical and can tolerate longer migration times. The algorithm adjusts the reserved bandwidth if the required bandwidth exceeds the available capacity, ensuring that the migration does not disrupt other network activities. If the available bandwidth is sufficient, the migration proceeds with available bandwidth; otherwise, the required bandwidth is reserved.

Finally, the migration is performed with the chosen migration technique utilizing the allocated bandwidth. Once the migration is complete, all traffic reservation rules are lifted on both the source and the destination servers.

Algorithm 2 Migration Bandwidth Reservation for Single VM Migration

---

    if Urgency = High **then**
        $mig\_bw = ideal\_bw$
        Reserve mig_bw for migration
    else if Urgency = Medium **then**
        $medium\_ideal\_bw = ideal\_bw \times 0.75$
        if $medium\_ideal\_bw > tot\_bw - sla\_lower$ **then**
            $mig\_bw = tot\_bw - sla\_lower$
        else
            $mig\_bw = medium\_ideal\_bw$
        end if
        if $mig\_bw \leq avail\_bw$ **then**
            Skip bandwidth reservation
        else
            Reserve mig_bw for migration
        end if
    else if Urgency = Low **then**
        $low\_ideal\_bw = ideal\_bw \times 0.5$
        if $low\_ideal\_bw > tot\_bw - sla\_upper$ **then**
            $mig\_bw = tot\_bw - sla\_upper$
        else
            $mig\_bw = low\_ideal\_bw$
        end if
        if $mig\_bw \leq avail\_bw$ **then**
            Migrate with $avail\_bw$
        else
            Reserve mig_bw for migration
        end if
    end if

Algorithm 3 Traffic-Sensitive Multiple VM Migration Algorithm

1: Monitor network parameters:
2:     $src\_out, src\_in, dest\_out, dest\_in$
3: Monitor VM traffic:
4:     $vm1\_out, vm1\_in, \ldots, vmN\_out, vmN\_in$
5: Compute available bandwidths:
6:     $src\_avail\_bw = tot\_bw - src\_out$
7:     $dest\_avail\_bw = tot\_bw - dest\_in$
8: Classify traffic types at source, destination and each VM: $inbound, outbound, mixed, idle$
9: Classify workload type of each VM based on vm_out, vm_in
10: Retrieve SLA bounds: sla_upper_i, sla_lower_i for each VM
11: Determine migration urgency level for each VM: $High, Medium, Low$
12: **if** Urgency = High **then**
13:     Strategy $\leftarrow$ Parallel
14: **else if** Urgency is Medium **and** $src\_avail\_bw, dest\_avail\_bw$ are sufficient **then**
15:     Strategy $\leftarrow$ Parallel
16: **else if** Urgency is Medium **and** bandwidth is limited **then**
17:     Strategy $\leftarrow$ Serial
18: **else if** Urgency = Low **then**
19:     Strategy $\leftarrow$ Serial
20: **end if**
21: **if** Strategy = Serial **then**
22:     Order VMs for migration (e.g., outbound traffic first)
23:     **for** each VM in sorted order **do**
24:         Decide technique (Pre-copy/Post-copy/Hybrid) based on server traffic
25:         Compute $ideal\_bw_i$ for the VM
26:         **if** $avail\_bw \geq ideal\_bw_i$ **then**
27:             Migrate VM with available bandwidth
28:         **else**
29:             Perform bandwidth reservation using priority and sla_lower_i
30:         **end if**
31:         Perform migration
32:         Update traffic profile of source and destination
33:     **end for**
34: **else**
35:     Decide technique based on current and post-migration source-destination traffic profiles
36:     Reserve bandwidth proportionally based on sla_lower_i and urgency
37:     Migrate VMs
38: **end if**

4.4.2 Multiple VM Migrations

Next, the algorithm was developed for the migration of multiple VM migrations as shown in Algorithm 3.

Monitoring Network Parameters - Inbound and outbound traffic rates at the source, destination, and each virtual machine (VM) are continuously monitored. Additionally, the available bandwidth at both the source and destination, potentially usable for migration, is calculated. This real-time monitoring ensures that the migration process is efficient and does not exceed the available bandwidth.

Classification of Traffic and Workload Type - Each VM is classified based on its traffic characteristics and workload type. This classification is similar to single VM migrations. The key classifications include inbound, outbound, idle and mixed. Each VM is associated with upper and lower SLA bounds corresponding to its workload type and urgency level.

Assignment of Migration Strategy (Serial or Parallel) - Each VM is assigned a migration strategy based on its priority and the level of traffic contention. The migration strategies are as follows.

- High Priority VMs - Assigned to Parallel Migration to minimize migration time.

- Medium Priority VMs with Low Contention - Assigned to Parallel Migration.

- Medium Priority VMs with High Contention - Assigned to Serial Migration to reduce contention.

- Low Priority VMs - Assigned to Serial Migration, as they have the least urgency.

Performing Serial Migrations - The VMs are sorted for migration in the order, with outbound and idle VMs first, and inbound VMs last. This sequence is based on the findings of Fernando, Terner, et al. 2019. Then, for each VM in the sorted list, the migration technique (Pre-copy or Post-copy) is selected based on the server traffic profiles at the time, bandwidth is reserved based on the VM's priority and

SLA bounds and the migration is performed similar to the single VM migration process (1). After each migration completes, the source and destination server traffic profiles are updated, and any traffic shaping rules are lifted. This process continues until all VMs are successfully migrated.

Performing Parallel Migrations - The migration strategy for each VM is determined based on the server traffic profiles and the available bandwidth.

- Post-copy is used if the source is outbound.

- Pre-copy is used if the destination is inbound.

- Pre-copy is used if the source is inbound or mixed and the VMs are inbound.

- Pre-copy is used if the destination is outbound or mixed and the VMs are inbound.

- Post-copy is used if the destination is idle.

- Pre-copy is used if the source is idle.

- If none of the above conditions apply, Post-copy is used by default.

Next, bandwidth is allocated based on the migration priority of the VMs and the sum of the SLA bounds of all VMs being migrated simultaneously. After the migration of all the VMs complete, any traffic shaping rules are lifted.

5   Evaluation

This section presents the performance evaluation of the proposed traffic-aware live VM migration algorithm. The evaluation focuses on two primary metrics: *total migration time*, *downtime*, and the *application performance degradation*.

5.1   Experimental Setup

The experiments were conducted on two (2) HP Z620 Workstation servers, each equipped with an Intel(R) Xeon(R) E5-1650 v2 CPU (12 cores, 3.50GHz) and 16 GB of memory. The servers were interconnected using a Gigabit Ethernet switch (1 Gbps full-duplex). A shared NFS server hosted the virtual disk images, eliminating the need for disk migration in LAN-based scenarios. The experimental environment used QEMU Emulator version 8.1.2, installed on both the source and destination hosts. Both migration and application traffic utilized the same network interface, inducing realistic contention scenarios.

5.2   Single VM Migrations

For single VM migrations, both synthetic and real-world VM workloads were used to test migration performance under varied conditions.

- *Transactional workloads* - Applications such as e-commerce and CRM require high availability, low latency, and strong data consistency (TPC-C, HammerDB).

- *Analytical workloads* - Used for data lakes and business intelligence, these workloads need high throughput for large data sets (TPC-H, BigBench benchmarks will assess performance).

- *Content Delivery Workloads* - Streaming media and CDNs demand high availability and low latency (Citron, wrk2)

- *Development and Testing Workloads* - These have lower availability and moderate latency needs, with intermittent bursts (Phoronix Test Suite, Jenkins Plugins)

- *Batch Processing Workloads* - Focused on tasks like ETL and report generation, these require cost efficiency and high throughput (HiBench, TPCx-BB).

- *Archival/ Idle Workloads* - Long-term data storage prioritizes high durability with low bandwidth usage (SPEC SFS2014, fio).

This selection ensures diverse workload scenarios to test the effectiveness of the migration algorithm under real-world conditions. Traffic at the source and destination servers were inbound, outbound, mixed and idle at a rate of 500 Mbps in the corresponding traffic direction.

Figure 8 plots compare the total migration time of the migration of a single VM running different workloads using the traffic-aware algorithm against the traditional live migration techniques, pre-copy, post-copy and hybrid. The traffic-aware migrations were tested under all three (3) priority levels, low, medium, and high. The figures show that, despite the background traffic profile combination at the source and the destination, the traffic-aware migration time is either lower or similar to the traditional techniques. According to the figures, the migration time is lowest when the migration is high-priority and migrated with the traffic-aware algorithm, followed by traffic-aware migrations for medium-priority migrations. The low-low priority traffic-aware migrations are seen to take time similar to or slightly higher than the lowest of the three traditional techniques. This is because while traffic-aware chooses the best algorithm based on the traffic profile, the bandwidth reservation is done to minimize the impact on the applications.

While traffic-aware consistently improves total migration time, its impact on downtime varies. In certain scenarios, downtimes are higher due to bandwidth being reserved to maintain application performance. For example, in the out-in (inbound VM) case, traffic-aware records a downtime of 516.7ms compared to 65.0ms with post-copy, a significant increase primarily due to stricter SLA adherence.

That said, traffic-aware outperforms in several cases. In the in-out (mixed) scenario, downtime drops to 27.5 ms, outperforming hybrid (31.7ms) and post-copy (109.7ms), a 13% and 75% reduction, respectively. Likewise, in the outbound application traffic at source and destination case, it records 70ms versus 661ms with hybrid, an 89% improvement.

These results suggest that although traffic-aware may trade off downtime in some SLA-sensitive cases, it still offers competitive or better performance when traffic patterns are more aligned.
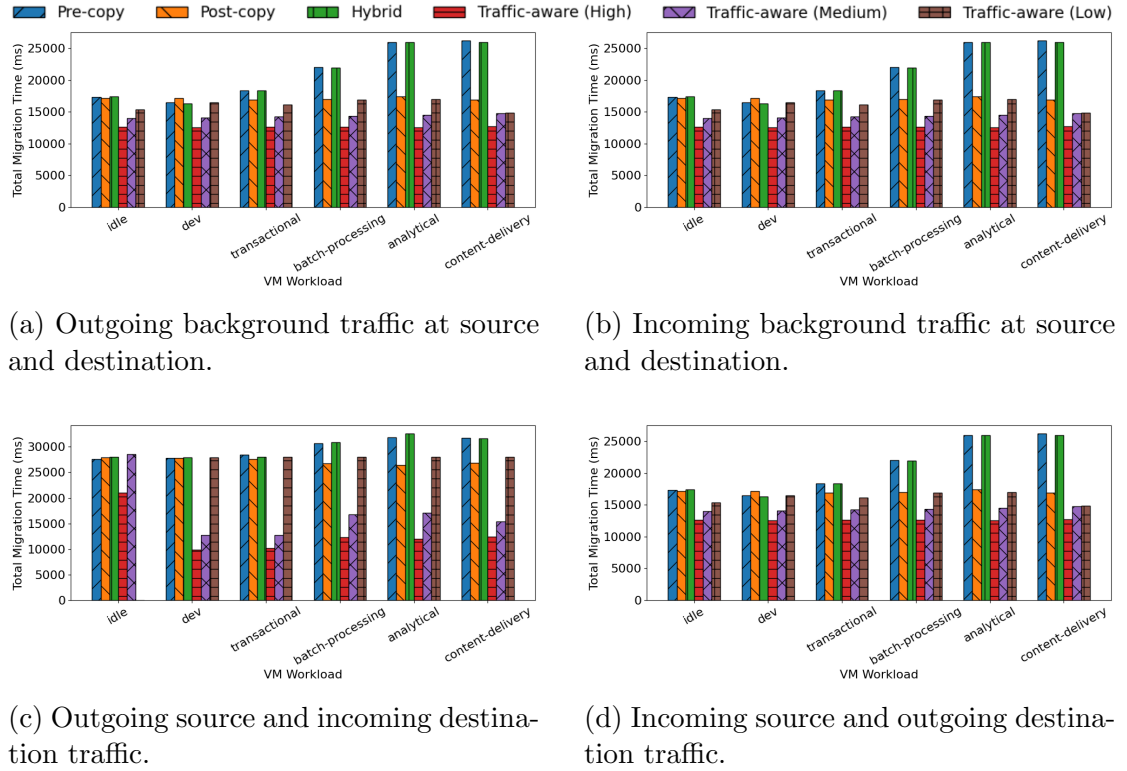

(a) Outgoing background traffic at source and destination.

(b) Incoming background traffic at source and destination.

(c) Outgoing source and incoming destination traffic.

(d) Incoming source and outgoing destination traffic.

Figure 8: Breakdown of background traffic directions during single VM migration.

5.3 Multiple VM Migrations

For multiple VM migrations, network-intensive workloads on VMs were generated using iPerf. VMs with inbound, outbound and mixed traffic were generated, each with incoming, outgoing network traffic at a rate of 200 Mbps. This value was chosen as the Ethernet bandwidth is 1000Mbps, and we wanted to have cases with

src-dest	VMs	Pre	Post	Hybrid	Traffic-aware
	in	32.7	33.0	30.4	21.8
idle-idle	out	24.9	21.1	24.3	21.1
	mixed	31.0	24.0	25.7	22.2
	in	25.0	48.4	26.1	23.5
in-in	out	39.9	140.5	43.8	34.4
	mixed	28.5	139.3	39.2	25.9
	in	19.5	88.4	25.8	17.9
in-out	out	30.6	32.4	33.3	18.4
	mixed	22.1	60.5	24.4	19.7
	in	29.5	71.5	32.1	27.2
out-in	out	21.2	57.4	22.3	20.4
	mixed	25.9	43.6	23.0	18.5
	in	20.7	104.0	24.3	20.9
out-out	out	22.9	25.8	22.0	21.3
	mixed	21.1	43.5	22.2	17.9

Table 1: Total Migration Time (s) for parallel VM migrations under high-priority settings.

outgoing traffic at both migrating VMs, and background traffic at the server at 300Mbps.

5.3.1 High Priority Migrations

Table 1 compares the total migration time of the traditional migration techniques against the traffic-aware aware approach. These experiments were performed to assess the speed of migrations in the traffic-aware approach under when the migration priority is high. Since traffic-aware adapts the parallel migration strategy in this scenario, it was compared with the parallel migration using traditional techniques. In all combinations of source-destination and VM traffic, traffic-aware gives the least migration time.

5.3.2 Low Priority Migrations

The traffic-aware algorithm was compared against serial migration of VMs using traditional techniques. The traffic-aware algorithm chose the technique that yields the least contention, resulting in the total migration time similar to the lowest of the traditional techniques. Most cases gave a slightly higher migration time as the

src-dest	VMs	Pre	Post	Hybrid	Traffic-aware
idle-idle	in-in	72	47	132	44
	in-out	60	62	646	36
	out-in	73	545	61	46
	out-out	240	53	619	45
in-in	in-in	99	25	122	68
	in-out	29405	34129	30378	94
	out-in	131	138	139	65
	out-out	228	22	233	38
in-out	in-in	266	22	709	27
	in-out	39	530	183	21
	out-in	100	96	180	21
	out-out	95	22	171	21
out-in	in-in	98	57	140	82
	in-out	60	710	189	152
	out-in	74	152	128	21
	out-out	208	22	403	30
out-out	in-in	266	154	363	23
	in-out	46	5950	97	24
	out-in	175	109	83	19
	out-out	335	23	105	23

Table 2: Downtime (ms) for serial VM migrations under low-priority settings.

bandwidth reservation is done to provide maximum bandwidth for the application traffic.

For low-priority migrations, while the traffic-aware approach consistently displayed higher migration times, it showed lower downtime (shown in Table 2) compared to traditional techniques across most traffic scenarios. This is because the algorithm prioritizes minimal disruption to co-hosted applications by choosing serial migration over parallel, resulting in significantly reduced interference. For instance, in the in-out scenario, traffic-aware achieves a downtime of just 21ms, whereas hybrid reaches 183ms, and post-copy spikes to 530ms. Similarly, in the out-in case, it brings downtime down to 21ms, compared to 128ms with hybrid. These results reflect the algorithm's ability to gracefully trade off migration speed for service continuity in low-priority contexts.

5.3.3 Medium Priority Migrations

When comparing traffic-aware migration of medium priority VMs to both high and low priority scenarios, the total migration time was found to be higher than for high-priority migrations but lower than for low-priority migrations (shown in Figure 9). The traffic-aware approach effectively balances migration speed and service continuity for medium-priority migrations. In terms of downtime, the traffic-aware method exhibited lower downtime than high-priority migrations, but slightly higher than for low-priority migrations. This suggests that while traffic-aware aims to optimize service continuity, it also ensures a more efficient migration process for medium-priority VMs than would typically be achieved by high-priority or low-priority configurations.

These results highlight the algorithm's flexibility in adapting to different priority levels, aiming for an optimal balance between migration speed and application impact.

5.4 CPU Utililzation

To evaluate the impact of different migration strategies on application performance, we analyzed the CPU utilization of a VM executing a `QuickSort` (Fernando, Terner, et al. 2019), a CPU-intensive workload during migration. We compared traditional pre-copy, post-copy, and the proposed traffic-aware migration strategy.

For a fair comparison with pre-copy, background traffic was introduced at the destination to simulate conditions under which the traffic-aware algorithm would also opt for pre-copy. The migrating VM generated outbound traffic, chosen due to the high contention this pattern introduces when using pre-copy. To balance the traffic and prevent the traffic-aware strategy from defaulting to post-copy, additional inbound background traffic was introduced to match the VM's outbound traffic.
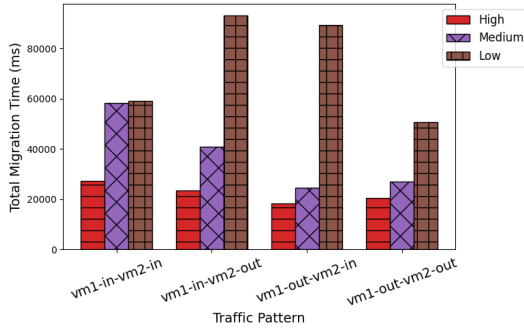
For the post-copy comparison, a VM with primarily inbound traffic was migrated, while outbound background traffic was generated at the source server.
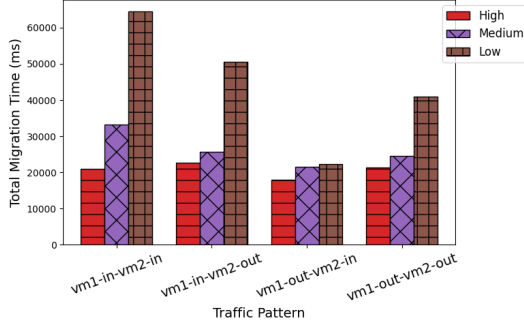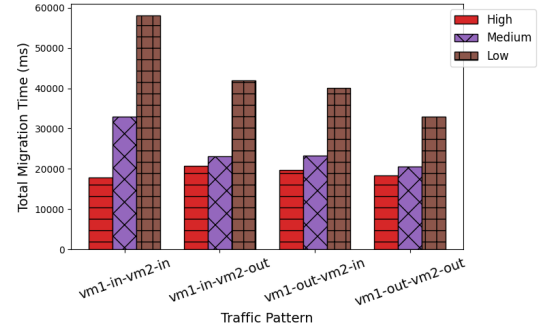
(a) Outgoing background traffic at source and destination
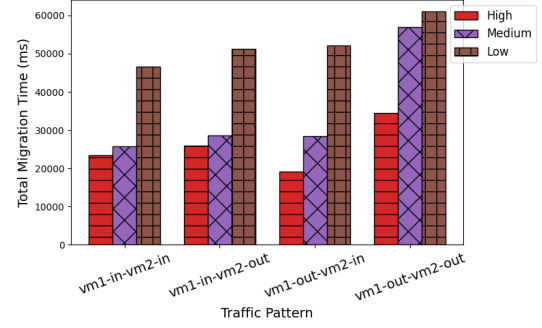
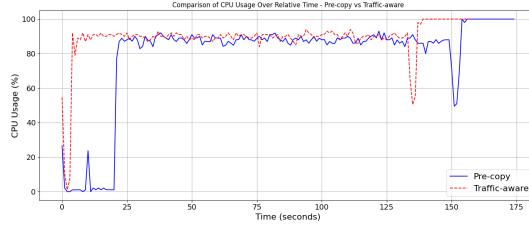(b) Incoming background traffic at source and destination

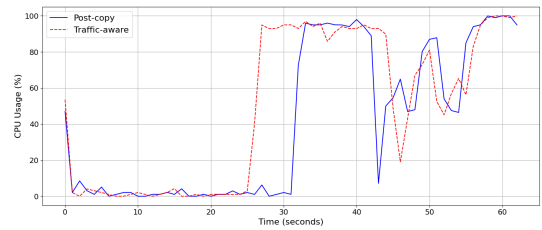(c) Outgoing source and incoming destination traffic.

(d) Incoming source and outgoing destination traffic.

Figure 9: Comparing total migration time of Traffic-aware migration under high, medium, low priority settings.

(a) Pre-copy vs traffic-aware.

(b) Post-copy vs traffic-aware.

Figure 10: CPU usage during QuickSort execution across migration strategies.

This setup represented a scenario where post-copy is typically favourable, but allowed assessment of the traffic-aware strategy's adaptability.

As shown in Figure 10, the traffic-aware approach consistently resulted in lower CPU usage impact across both scenarios. This improvement is attributed to more efficient bandwidth allocation for migration traffic, which reduces total downtime, which is the period when the VM is halted and CPU usage drops to 0%.

In the pre-copy scenario, the total migration time was 11.4 seconds, while the traffic-aware strategy reduced it to 9.5 seconds. Similarly, in the post-copy scenario, the migration duration decreased from 11.9 seconds to 10.9 seconds with the traffic-aware approach. Importantly, the traffic-aware strategy achieves this reduction without introducing additional overhead or disruptions. Dynamically selecting between pre-copy and post-copy based on current system conditions effectively mitigates performance degradation compared to traditional methods.

6 Discussion

The proposed traffic-aware live virtual machine (VM) migration framework represents a significant step toward resolving a persistent challenge in cloud data centers, network contention during migration. By incorporating real-time traffic monitoring, urgency-based prioritization, and SLA-aware bandwidth reservation, the algorithm adapts dynamically to diverse network conditions and workload sensitivities, resulting in improved performance across a range of operational scenarios. Each of the core research questions identified at the outset is addressed through empirical evaluation and system design, as discussed below.

Effectiveness of the Traffic-Aware Approach - In response to Research Question 1, the framework introduces a traffic-sensitive migration controller capable of dynamically selecting the most suitable migration technique, pre-copy, post-copy, or hybrid, based on current traffic conditions, available bandwidth, and congestion levels. The empirical evaluation demonstrates that this adaptive selection leads to consistently superior performance when compared to static migration strategies. For high-priority migrations, the system effectively utilizes parallelism and network bandwidth to minimize total migration time, whereas for low-priority migrations, it shifts to a more conservative, serial strategy to preserve application performance.

Empirical bandwidth models, another core contribution, play a key role in enabling this dynamic adaptability. By avoiding static thresholds and instead estimating bandwidth availability in real-time based on directional traffic intensity, the system achieves precise bandwidth reservations. This not only answers the technical challenge in Research Question 1 but also lays the groundwork for more intelligent and responsive migration planning.

Priority-Awareness and SLA Compliance - The integration of urgency-aware prioritization and SLA-compliant bandwidth reservation directly addresses Research Question 3, which focuses on how to balance migration efficiency with SLA guarantees. The proposed algorithm incorporates SLA bounds into its decision-

making process, ensuring that high-priority migrations are expedited while medium- and low-priority migrations proceed only when they do not compromise the performance of co-located workloads. This structured prioritization introduces a principled mechanism to achieve fairness and efficiency in resource allocation, particularly in multi-tenant environments.

This SLA-awareness allows the system to minimize service disruption during migration, preserve application responsiveness, and reduce the risk of SLA violations, outcomes that are critical for cloud service providers managing diverse customer workloads.

Comparative Advantage Over Existing Techniques - With respect to Research Question 2, the evaluation shows that the proposed adaptive bandwidth reservation and prioritization strategies significantly improve both total migration time and VM downtime, across single and multi-VM scenarios. Compared to traditional techniques, which often rely on uniform, non-adaptive strategies, the proposed system responds dynamically to traffic and workload variations.

- Context-Sensitive Strategy Selection: The framework monitors real-time traffic and selects migration techniques accordingly, avoiding the one-size-fits-all limitation of prior methods. This enables more efficient migrations even in high-traffic conditions.

- Adaptability to Multi-VM Scenarios: Unlike existing approaches that assume homogeneity in VM behaviour, the traffic-aware algorithm makes per-VM decisions, improving scalability and handling workload diversity more effectively.

These results confirm that adaptive bandwidth management not only improves migration speed but also lowers disruption, answering Research Question 2 with strong empirical backing.

Practical Implications and Deployment Feasibility - A major strength of the proposed framework is its deployability using existing tools like `tc` and `ifstat`, ensuring feasibility in production environments without requiring specialized in-

44

frastructure. This practicality makes the research outcomes directly translatable to cloud operations.

The current work focuses on LAN-based migration, and as such, wide-area migrations, where latency and jitter are more pronounced, are not yet addressed. Expanding the framework to handle WAN migrations, potentially using latency-aware SLA models, represents a valuable direction for future work.

Similarly, the current model primarily addresses SLA constraints at the source host. Future iterations could benefit from integrating destination-side considerations, including post-migration contention and inter-VM dependencies. This would allow for more holistic decision-making and further reduce the risk of SLA degradation after migration.

Finally, introducing predictive or learning-based models could significantly enhance the system's ability to anticipate traffic patterns and make preemptive migration decisions. This forward-looking capability would help optimize resource utilization and prevent migration bottlenecks, especially in highly dynamic environments.

7 Conclusion

This thesis presented a traffic-aware, SLA and priority-sensitive approach to live virtual machine migration, designed to mitigate network contention and enhance service continuity in cloud data centers. Unlike traditional migration techniques that adopt a static strategy regardless of context, the proposed algorithm dynamically selects between pre-copy and post-copy methods based on real-time traffic conditions at the source and destination hosts. It further integrates bandwidth reservation mechanisms that adapt to the urgency of the migration and ensure compliance with Service Level Agreements.

The system supports both single and multiple VM migrations, employing empirical bandwidth models to make accurate and efficient decisions under varying network loads. Evaluation results demonstrate that the proposed algorithm significantly reduces total migration time for high-priority tasks while minimizing downtime for low-priority workloads, outperforming conventional techniques across diverse scenarios.

This work underscores the value of integrating traffic-awareness, prioritization, and SLA considerations into live migration systems. It provides a scalable, deployable framework that improves the reliability and efficiency of virtualized infrastructure management in modern cloud environments.

Future research will extend this model to handle WAN-based migrations and incorporate post-migration SLA monitoring at the destination host. Additionally, integrating predictive analytics or learning-based methods may further enhance the adaptability and performance of the migration process in highly dynamic cloud ecosystems.

References

Altahat, Mohammad A. et al. (2020). "Dynamic Hybrid-copy Live Virtual Machine Migration: Analysis and Comparison". In: vol. 171. DOI: `10.1016/j.procs.2020.04.156`.

Bahrami, Marziyeh, Abolfazl Toorooghi Haghighat, and Majid Gholipoor (n.d.). *A Review of Virtual Machine Migration Techniques in Data Center*. URL: `https://www.researchgate.net/publication/372409749`.

Callegati, Franco and Walter Cerroni (2013). "Live migration of virtualized edge networks: Analytical modeling and performance evaluation". In: *2013 IEEE SDN for future networks and services (SDN4FNS)*. IEEE, pp. 1–6.

Cerroni, Walter and Flavio Esposito (2016). "Optimizing live migration of multiple virtual machines". In: *IEEE Transactions on Cloud Computing* 6.4, pp. 1096–1109.

Chang, Victor, Robert John Walters, and Gary Wills (2013). "Cloud Storage and Bioinformatics in a private cloud deployment: Lessons for Data Intensive research". In: *Cloud Computing and Services Science: Second International Conference, CLOSER 2012, Porto, Portugal, April 18-21, 2012. Revised Selected Papers 2*. Springer, pp. 245–264.

Choudhary, Anita et al. (2017). "A critical survey of live virtual machine migration techniques". In: *Journal of Cloud Computing* 6.1, pp. 1–41.

Clark, Christopher et al. (2005). "Live migration of virtual machines". In: *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 273–286.

Cui, Yong, Zhenjie Yang, et al. (2017). "Traffic-aware virtual machine migration in topology-adaptive dcn". In: *IEEE/ACM Transactions on Networking* 25.6, pp. 3427–3440.

Cui, Yong, Liang Zhu, et al. (2020). "An adaptive traffic-aware migration algorithm selection framework in live migration of multiple virtual machines". In:

International Journal of Performability Engineering 16 (2), pp. 314–324. ISSN: 09731318. DOI: `10.23940/ijpe.20.02.p14.314324`.

Dalvandi, Aissan, Mohan Gurusamy, and Kee Chaing Chua (2015). "Time-aware vmflow placement, routing, and migration for power efficiency in data centers". In: *IEEE Transactions on Network and Service Management* 12.3, pp. 349–362.

Deshpande, Umesh and Kate Keahey (July 2017). "Traffic-sensitive Live Migration of Virtual Machines". In: *Future Generation Computer Systems* 72, pp. 118–128. ISSN: 0167739X. DOI: `10.1016/j.future.2016.05.003`.

Deshpande, Umesh, Xiaoshuang Wang, and Kartik Gopalan (2011). "Live gang migration of virtual machines". In: *Proceedings of the 20th international symposium on High performance distributed computing*, pp. 135–146.

Devi, L Yamuna, P Aruna, N Priya, et al. (2011). "Security in virtual machine live migration for KVM". In: *2011 International Conference on Process Automation, Control and Computing*. IEEE, pp. 1–6.

Fernando, Dinuni, Jonathan Terner, et al. (2019). "Live migration ate my vm: Recovering a virtual machine after failure of post-copy live migration". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, pp. 343–351.

Fernando, Dinuni, Ping Yang, and Hui Lu (2020). "SDN-based Order-aware Live Migration of Virtual Machines". In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 1818–1827. DOI: `10.1109/INFOCOM41043.2020.9155415`.

Fu, Xiong et al. (2019). "Network traffic based virtual machine migration in cloud computing environment". In: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, pp. 818–821.

Gao, Yongqiang et al. (2014). "Service level agreement based energy-efficient resource management in cloud data centers". In: *Computers & Electrical Engineering* 40.5, pp. 1621–1633.

Haidri, Raza A et al. (2022). "A deadline aware load balancing strategy for cloud computing". In: *Concurrency and Computation: Practice and Experience* 34.1, e6496.

Hines, Michael R, Umesh Deshpande, and Kartik Gopalan (n.d.). *Post-Copy Live Migration of Virtual Machines*.

Hines, Michael R and Kartik Gopalan (2009). "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning". In: *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 51–60.

Hu, Liting et al. (2008). "Magnet: A novel scheduling policy for power reduction in cluster with virtual machines". In: *2008 IEEE International Conference on Cluster Computing*. IEEE, pp. 13–22.

Huang, Wei et al. (2007). "Virtual machine aware communication libraries for high performance computing". In: *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pp. 1–12.

Jo, Changyeon et al. (2013). "Efficient live migration of virtual machines using shared storage". In: *ACM Sigplan Notices* 48.7, pp. 41–50.

Kanniga Devi, R, G Murugaboopathi, and M Muthukannan (2018). "Load monitoring and system-traffic-aware live VM migration-based load balancing in cloud data center using graph theoretic solutions". In: *Cluster Computing* 21.3, pp. 1623–1638.

Kherbache, Vincent, Eric Madelaine, and Fabien Hermenier (2017). "Scheduling live migration of virtual machines". In: *IEEE transactions on cloud computing* 8.1, pp. 282–296.

Kuno, Yosuke, Kenichi Nii, and Saneyasu Yamaguchi (2011). "A study on performance of processes in migrating virtual machines". In: *2011 Tenth International Symposium on Autonomous Decentralized Systems*. IEEE, pp. 567–572.

Le, Tuan (Nov. 2020). *A survey of live Virtual Machine migration techniques*. DOI: 10.1016/j.cosrev.2020.100304.

Linux ifstat (n.d.). URL: https://www.man7.org/linux/man-pages/man8/ifstat.8.html.

Linux tc (n.d.). URL: https://man7.org/linux/man-pages/man8/tc.8.html.

Liu, Jiaqiang et al. (2015). "Traffic aware cross-site virtual machine migration in future mobile cloud computing". In: *Mobile Networks and Applications* 20, pp. 62–71.

Lu, Kuan et al. (2013). "QoS-aware VM placement in multi-domain service level agreements scenarios". In: *2013 IEEE Sixth International Conference on Cloud Computing*. IEEE, pp. 661–668.

Maheshwari, Sumit et al. (2018). "Traffic-aware dynamic container migration for real-time support in mobile edge clouds". In: *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, pp. 1–6.

Manzalini, Antonio et al. (2013). "Clouds of virtual machines in edge networks". In: *IEEE Communications Magazine* 51.7, pp. 63–70.

Nadeem, Hanan A, Hanan Elazhary, and Mai A Fadel (2018). "Priority-aware virtual machine selection algorithm in dynamic consolidation". In: *International Journal of Advanced Computer Science and Applications* 9.11.

Nagarajan, Arun Babu et al. (2007). "Proactive fault tolerance for HPC with Xen virtualization". In: *Proceedings of the 21st annual international conference on Supercomputing*, pp. 23–32.

Nasim, Robayet and Andreas J Kassler (2015). "Network-centric performance improvement for live VM migration". In: *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, pp. 106–113.

Nathuji, Ripal and Karsten Schwan (2007). "Virtualpower: coordinated power management in virtualized enterprise systems". In: *ACM SIGOPS operating systems review* 41.6, pp. 265–278.

Nelson, Michael, Beng-Hong Lim, Greg Hutchins, et al. (2005). "Fast Transparent Migration for Virtual Machines." In: *USENIX Annual technical conference, general track*, pp. 391–394.

Odun-Ayo, Isaac, Blessing Udemezue, and Abiodun Kilanko (2019). "Cloud service level agreements and resource management". In: *Adv. Sci. Technol. Eng. Syst.* 4.2, pp. 228–236.

Padala, Pradeep et al. (2007). "Adaptive control of virtualized resources in utility computing environments". In: *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pp. 289–302.

Sahni, Shashank and Vasudeva Varma (2012). "A hybrid approach to live migration of virtual machines". In: *2012 IEEE international conference on cloud computing in emerging markets (CCEM)*. IEEE, pp. 1–5.

Shrivastava, Vivek et al. (2011). "Application-aware virtual machine migration in data centers". In: *2011 Proceedings IEEE INFOCOM*. IEEE, pp. 66–70.

Son, Jungmin and Rajkumar Buyya (2018). "Priority-aware VM allocation and network bandwidth provisioning in software-defined networking (SDN)-enabled clouds". In: *IEEE Transactions on Sustainable Computing* 4.1, pp. 17–28.

Soni, Gulshan and Mala Kalra (Dec. 2013). "Comparative Study of Live Virtual Machine Migration Techniques in Cloud". In: *International Journal of Computer Applications* 84 (14), pp. 19–25. DOI: 10.5120/14643-2919.

Sun, Gang, Dan Liao, Vishal Anand, et al. (2016). "A new technique for efficient live migration of multiple virtual machines". In: *Future Generation Computer Systems* 55, pp. 74–86.

Sun, Gang, Dan Liao, Dongcheng Zhao, et al. (2015). "Live migration for multiple correlated virtual machines in cloud-based data centers". In: *IEEE Transactions on Services Computing* 11.2, pp. 279–291.

Tsakalozos, Konstantinos et al. (2017). "Live VM migration under time-constraints in share-nothing IaaS-clouds". In: *IEEE Transactions on Parallel and Distributed Systems* 28.8, pp. 2285–2298.

Tso, Fung Po et al. (2014). "Scalable traffic-aware virtual machine management for cloud data centers". In: *2014 IEEE 34th International Conference on Distributed Computing Systems*. IEEE, pp. 238–247.

Voorsluys, William et al. (2009). *Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation.* URL: `http://www.cloudbus.org`.

Wood, Timothy, KK Ramakrishnan, et al. (2014). "CloudNet: Dynamic pooling of cloud resources by live WAN migration of virtual machines". In: *IEEE/ACM Transactions On Networking* 23.5, pp. 1568–1583.

Wood, Timothy, Prashant J Shenoy, et al. (2007). "Black-box and Gray-box Strategies for Virtual Machine Migration." In: *NSDI*. Vol. 7, pp. 17–17.