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

VENUDI DAYARATNE[1], DINUNI FERNANDO[2]

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.

INDEX TERMS VM Migration, Traffic-Aware, Bandwidth Reservation, SLA, Pre-Copy, Post-Copy, Cloud Computing

I. INTRODUCTION

THE rapid growth of cloud computing has significantly increased the demand for scalable and on-demand computing resources. Central to this paradigm is virtualization, which enables the abstraction of physical resources into virtual instances that can be accessed and utilized concurrently by multiple clients. A Virtual Machine (VM) emulates a physical computing environment, including its own operating system (OS), CPU, memory, and peripheral functionalities. These VMs are hosted on physical servers, each capable of supporting multiple VMs simultaneously. The ability to relocate VMs across physical hosts, termed *VM migration*, is a critical feature of modern cloud infrastructures [1].

A specific form of VM migration, known as *live migration*, allows a VM to be transferred between hosts with minimal service disruption. This process can be executed sequentially, *serial live migration*, or concurrently, *gang/ parallel live migration* [2]. Live migration typically involves transferring the VM's memory state, disk storage, and network connections from the source to the destination host [3]. In Local Area Networks (LANs) which utilizes a Network Attached Storage (NAS), disk migration is often unnecessary [4]. However, in Wide Area Network (WAN) scenarios, both memory and disk states must be transferred [5].

Live VM migration within Cloud Data Centers (CDCs) is essential for a variety of operational requirements, including load balancing [6], [7], fault tolerance [8], system maintenance [9], and energy efficiency through workload consolidation [10], [11]. To preserve Quality of Service (QoS) and minimize Service Level Agreement (SLA) violations, these migrations must be executed swiftly and efficiently.

Three principal live migration techniques have been established: *pre-copy* [12], [13], *post-copy* [14], [15], and *hybrid-copy* [16]. These techniques are primarily designed for single VM migrations. However, in multi-VM scenarios, current implementations often apply a uniform migration strategy to all VMs, disregarding their individual characteristics or workload demands.

A critical challenge arises when migration traffic shares the same Network Interface Card (NIC) as the VM's operational workload. In such cases, network contention can occur, particularly affecting pre-copy and post-copy techniques. For pre-copy, outgoing traffic from the source host competes with migration traffic, while in post-copy, incoming traffic to the destination host faces similar contention [17]. This interference degrades application performance and prolongs both migration and downtime durations. As cloud services continue to scale, the intensity of such network contention escalates, further complicating efficient migration management.

Moreover, the choice between serial and parallel migration introduces trade-offs. Serial migration, which processes VMs one at a time, can reduce overall migration duration due to lower resource contention. Conversely, parallel migration enhances availability and reduces downtime, albeit with higher total resource consumption and extended migration time [18]. These trade-offs are particularly relevant in environments with strict SLA and QoS requirements [19].

To overcome these limitations, a traffic-aware migration strategy is required. One that dynamically accounts for the contention between workload and migration traffic, while balancing migration time and downtime. Integrating traffic shaping and prioritization into the migration decision-making process can enable more efficient and adaptive migration scheduling.

The remainder of this paper is organized as follows: Section II reviews the foundational live migration techniques, SLA considerations, and VM prioritization. Sections III and IV detail the design and implementation of the proposed traffic-aware migration algorithm. Section V evaluates the algorithm's performance under diverse traffic conditions using defined performance metrics. Section VI discusses related work, and Section VII concludes the paper while outlining future research directions.

II. BACKGROUND

A. LIVE VM MIGRATION

Live migration refers to the process of transferring a virtual machine (VM) from one physical host to another while it remains operational. During this process, memory pages, CPU state, and disk I/O states are replicated to the destination host without interrupting the running services [12].

a: Pre-copy Migration Technique

In the pre-copy approach, the VM continues to execute on the source host while its memory pages are transferred to the destination. This process continues iteratively until a Writable Working Set (WWS) is identified [15]. Once this threshold is reached, the migration enters the stop-and-copy phase, during which the VM is briefly paused. During this phase, the remaining memory pages and CPU state are transferred to the target, and network connections are redirected to the new host.

b: Post-copy Migration Technique

In the post-copy method, the VM is first suspended and minimal CPU and non-pageable memory state are transferred to the destination host. The VM is then resumed at the destination, and the remaining memory pages are fetched from the source host on demand [1].

c: Hybrid-copy Migration Technique

The hybrid-copy technique combines the strengths of both pre-copy and post-copy methods to reduce downtime and avoid performance degradation. Initially, memory pages are copied to the destination as in pre-copy. The VM is then resumed at the destination, with any remaining dirty pages retrieved from the source host on demand, as in post-copy [16].

B. MULTIPLE VM MIGRATIONS

Co-located VMs, which reside on the same physical host, often interact closely, providing complementary services. Consolidating them on the same machine reduces communication overhead and improves overall system efficiency [7], [20]. However, scenarios such as energy optimization and performance maintenance may necessitate the migration of multiple correlated VMs [21]. These migrations can be performed either sequentially (serial migration) or concurrently (parallel migration) [18], [22], [23].

In serial migration, each VM is migrated one at a time using standard single-VM migration protocols. In contrast, gang migration refers to parallel migration, where multiple VMs are transferred simultaneously, sharing the available bandwidth.

Gang migration typically results in lower overall downtime, improving service availability. However, this approach often leads to longer total migration times due to higher resource contention. Serial migration, on the other hand, achieves shorter total migration durations by minimizing communication overhead and network competition [18]. The trade-off between reduced downtime and efficient resource use makes the choice of migration strategy application-specific.

C. MIGRATION PRIORITY

Migration priority defines the urgency with which a VM should be transferred. High-priority migrations, such as those triggered by imminent hardware failures, benefit from gang migration to minimize service disruption, even at the cost of greater bandwidth usage. Conversely, low-priority tasks, such as routine maintenance, are better suited to serial migration, which is less intrusive [24], [25].

Different applications have distinct latency sensitivities and migration deadlines [19]. For instance, virtual network functions (VNFs) often require low-latency transitions, while web services can tolerate longer migration durations. In such cases, system administrators must consider SLA constraints and application criticality when selecting between serial and parallel strategies and determining the necessary bandwidth reservations.

D. SERVICE LEVEL AGREEMENTS

Service Level Agreements (SLAs) are formal contracts between cloud providers and customers that define specific performance guarantees, such as availability, latency, and fault tolerance [26]. Violating these terms can lead to penalties and diminished trust, making compliance essential [27], [28].

Quality of Service (QoS) metrics provide quantifiable standards for evaluating SLA adherence. These metrics enable cloud providers to manage network and computational resources effectively while upholding service guarantees during events like live migrations.

E. PERFORMANCE METRICS

Evaluating the efficiency of live VM migration requires comprehensive performance metrics. Commonly used indicators include total migration time, downtime, and application degradation [14], [29]–[32]:

- **Total Migration Time (TMT):** The time from the initiation to the completion of the migration process.
- **Downtime (DT):** The period during which the VM's services are unavailable to users.

These metrics are critical for assessing the trade-offs between migration efficiency and service continuity.

F. TRAFFIC CONTENTION PROBLEM

Live migration introduces two primary traffic types: application (workload) traffic and migration traffic. Application traffic includes communication between VMs, client requests, and internal system tasks. Migration traffic involves the transfer of VM state data across the network.

a: Migration Traffic

- **Pre-copy migration** produces primarily outbound traffic as it pushes memory pages from the source to the destination.
- **Post-copy migration** generates mostly inbound traffic, with the destination host pulling memory pages as needed.

b: Application Traffic

- **Incoming traffic** consists of external data or client requests sent to the VM [33].
- **Outgoing traffic** includes data served by the VM to clients or other systems [17].

Since both migration and application traffic share the same network interface, simultaneous activity can lead to bandwidth contention [17]. For instance, in pre-copy migration, outgoing application traffic competes with migration traffic at the source host. In post-copy migration, the incoming application traffic at the destination competes with the migration stream, which can increase latency and slow the migration process [33].

This contention delays the release of resources at the source host and may degrade the performance of network-bound applications. Gang migration exacerbates this problem, as multiple VMs simultaneously add to the overall traffic, straining bandwidth resources and impacting service quality.

G. TRAFFIC SHAPING

Traffic shaping and bandwidth reservation are essential for mitigating network congestion and ensuring fair resource distribution in environments with shared infrastructure. Effective traffic control improves overall network efficiency and protects service quality during events like live VM migration.

In Linux-based systems, the `tc` (Traffic Control) utility is commonly used to manage traffic flows [34]. This tool allows administrators to regulate bandwidth usage, introduce controlled delays, prioritize traffic, and limit packet loss. By defining hierarchical classes and queues, `tc` enables granular control over different types of traffic based on attributes such as IP address, port, or protocol. Such mechanisms are critical in allocating sufficient bandwidth to migration processes without compromising the performance of active applications.

III. DESIGN

The proposed architecture for traffic-aware live VM migration, illustrated in Figure 1, is composed of three core modules that collaboratively 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*.

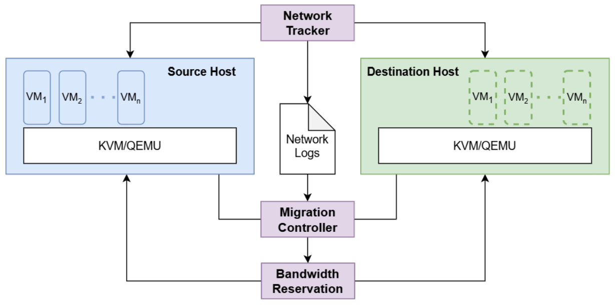


FIGURE 1: High-level architecture of the proposed traffic-aware live VM migration system.

A. NETWORK TRACKER

The *Network Tracker* module continuously monitors network activity at both the source and destination servers. It captures metrics on total traffic over physical Ethernet interfaces and per-VM traffic through tap interfaces. This information is periodically logged and stored in shared log files accessible to other modules.

Monitoring is conducted independently at both source and destination, allowing for a comprehensive view of network utilization. The implementation leverages the Linux utility `ifstat` to gather traffic statistics in real time [35]. These statistics form the basis for informed decision-making in subsequent modules.

B. BANDWIDTH RESERVATION

The *Bandwidth Reservation* module performs traffic shaping to mitigate contention between migration and non-migration traffic. It identifies migration-related flows based on a combination of parameters including the designated migration port, the source IP, and the destination IP of the migrating VM.

Based on the migration technique and the workload characteristics, this module reserves a dedicated portion of the available bandwidth for migration traffic. The remainder is allocated to regular workload traffic to prevent degradation in application performance. This functionality is implemented on the source host using the `tc` (Traffic Control) utility, which enables fine-grained bandwidth allocation.

C. MIGRATION CONTROLLER

The *Migration Controller* serves as the decision-making unit of the architecture. It utilizes input from the Network Tracker, specifically the current network load and traffic patterns at both source and destination nodes, and combines it with the characteristics and urgency of the VMs to determine the optimal migration strategy.

Based on this information, the controller selects an appropriate migration technique, such as pre-copy or post-copy, and defines bandwidth allocation policies accordingly. Upon finalizing a decision, it activates the Bandwidth Reservation module and orchestrates the initiation of the VM migration process.

This adaptive, feedback-driven mechanism ensures that migrations are conducted in a manner that minimizes service disruption, respects QoS and SLA requirements, and efficiently utilizes available network resources.

IV. IMPLEMENTATION

This section presents the design of the proposed traffic-aware migration algorithm. The algorithm is structured in two parts: the first addresses the decision-making process for migrating a single virtual machine (VM), while the second focuses on the strategy for managing the migration of multiple VMs.

The design is based on several assumptions. The algorithm specifically aims to reduce network contention at host servers rather than resolving CPU or memory bottlenecks. Migration priority is assigned at the host level, implying that all VMs selected for migration during a particular cycle share the same priority classification. The network environment assumes the use of 1 Gbps full-duplex Network Interface Cards (NICs), with migration and application traffic sharing the same bidirectional bandwidth. This setup reflects typical conditions found in contemporary data centers [17]. The scope of the algorithm is limited to Local Area Network (LAN)-based migrations between fixed source and destination hosts. Furthermore, in many data center environments, VMs providing related services are co-located on a single physical machine and are typically migrated together to preserve service locality and performance.

A. TRAFFIC-AWARE MIGRATION ALGORITHM FOR A SINGLE VM

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 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

Algorithm 1 Traffic-Sensitive Single VM Migration Algorithm

```
Monitor Network Parameters: $src\_out, src\_in, dest\_out, dest\_in, vm\_out, vm\_in$
Compute available bandwidth at source: $src\_avail\_bw = tot\_bw - src\_out$
Compute available bandwidth at destination: $dest\_avail\_bw = tot\_bw - dest\_in$
Classify source and destination traffic types: $inbound, outbound, mixed, idle$
Classify workload type based on $vm\_out, vm\_in$
Retrieve SLA bounds: $sla\_upper, sla\_lower$
Determine migration technique:
if Source is outbound then
    Technique = Post-copy
else if Source is inbound then
    Technique = Pre-copy
else if Destination is idle then
    Technique = Post-copy
else if Source is idle then
    Technique = Pre-copy
else
    Technique = Post-copy
end if
Compute $ideal\_bw$ based on $vm\_out, vm\_in$ and technique
if $avail\_bw \geq ideal\_bw$ then
    Migrate with available bandwidth
else
    Perform bandwidth reservation based on migration priority
end if
Perform migration
```

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 $\pm$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 IV-A1). 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 based on the urgency of the migration and the available network resources. 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. If the reserved bandwidth exceeds the available network resources, it is adjusted to fit within the remaining capacity. If sufficient bandwidth is available, the migration can proceed without an explicit reservation.

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.

1) Empirical Bandwidth Models

Empirical models were constructed by analyzing bandwidth demands under different traffic scenarios using pre-copy, post-copy, and hybrid-copy techniques. Each model is represented as a linear equation of the form $y = mx + c$, where$x$ denotes the background traffic (in Mbps), and y is the required bandwidth for migration.

Separate models were developed for scenarios involving only incoming traffic, only outgoing traffic, and mixed traffic (both directions). Breakpoints were introduced in cases where traffic behaviour changed significantly across different load ranges. These models allow the algorithm to make accurate bandwidth predictions based on real-time traffic observations, thereby supporting more effective migration decisions.

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)

  S1: $y = -0.91x + 906.80$ S2: $y = -0.001x + 467.90$

Post-copy Migration

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
```

- Incoming Traffic (Breakpoints: 0, 510, 1000)
  S1: $y = -1.02x + 861.00$ S2: $y = -0.12x + 399.00$
- Outgoing Traffic: $y = 848.10 + 0.0010x$
- Mixed Traffic (Incoming Varied, Outgoing = 500 Mbps, Breakpoints: 0, 528, 1000)
  S1: $y = -0.98x + 857.90$ S2: $y = -0.09x + 386.40$

Hybrid-copy Migration

- Incoming Traffic: $y = 896.82 + 0.0016x$
- Outgoing Traffic (Breakpoints: 0, 476, 1000)
  S1: $y = -0.90x + 897.66$ S2: $y = -0.01x + 478.38$
- Mixed Traffic (Outgoing Varied, Incoming = 500 Mbps, Breakpoints: 0, 485, 1000)
  S1: $y = -0.90x + 898.40$ S2: $y = 0.003x + 462.86$

B. TRAFFIC-AWARE MIGRATION ALGORITHM FOR MULTIPLE VMS

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

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 ← Parallel
14: else if Urgency is Medium and $src\_avail\_bw, dest\_avail\_bw$ are sufficient then
15:     Strategy ← Parallel
16: else if Urgency is Medium and bandwidth is limited then
17:     Strategy ← Serial
18: else if Urgency = Low then
19:     Strategy ← 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
```

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 lasThis sequence is based on the findings of Fernando et al [36]. Then, for each VM in the sorted list, the migration technique (Pre-copy or Post-copy) is selected based on the type of VM, 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 its traffic profile 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.

V. 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* and *downtime*.

A. 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.

B. SINGLE VM MIGRATIONS

1) Total Migration Time

Figure 2 compares the total migration times for a single VM using the proposed traffic-aware algorithm against traditional live migration techniques: pre-copy, post-copy, and hybrid. The traffic-aware method was evaluated across three priority levels: low, medium, and high.

Results indicate that traffic-aware migration, particularly at high priority, achieves the lowest migration times across all background traffic configurations. Medium-priority migrations also outperform most traditional techniques. Low-priority migrations perform comparably or slightly slower than the most efficient traditional method due to intentional bandwidth reservations to protect application performance.

An exception is observed in Figure 2d, where traditional and traffic-aware approaches yield similar migration times. This is due to the minimal contention introduced by the specific traffic direction—migration traffic is orthogonal to workload traffic, reducing interference.

2) Downtime

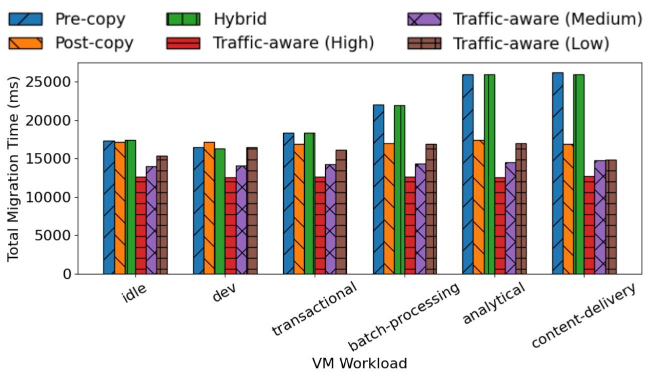
While traffic-aware migration improves total migration time, its impact on downtime varies. In SLA-sensitive scenarios, such as the inbound-VM case, traffic-aware records a higher downtime (516.7 ms) compared to post-copy (65.0 ms), as bandwidth is preserved for application traffic.

However, traffic-aware significantly reduces downtime in many cases. In the in-out (mixed) scenario, downtime drops to 27.5 ms—13% lower than hybrid (31.7 ms) and 75% lower than post-copy (109.7 ms). In the out-out scenario, it achieves 70 ms compared to 661 ms with hybrid—an 89% improvement.

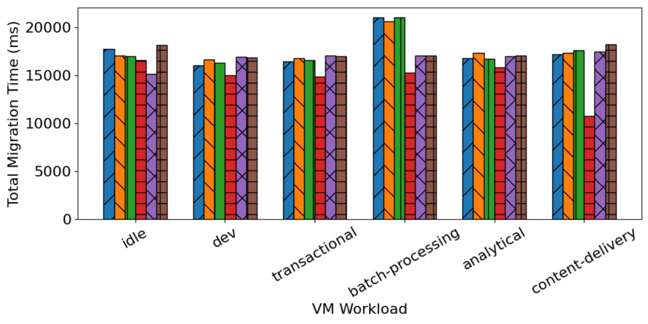
These results confirm traffic-aware's effectiveness in balancing service continuity with migration efficiency.

C. MULTIPLE VM MIGRATIONS

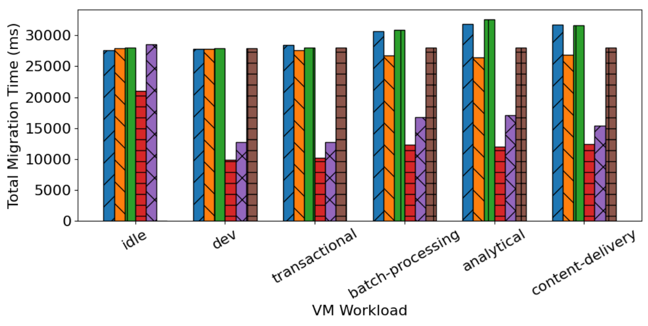
Experiments were extended to multiple VMs to evaluate performance under serial and parallel migration strategies. Two VMs were simultaneously hosted and migrated. A third server generated background traffic using `iperf`, simulating varying network conditions. Each scenario was tested five times; outliers were excluded, and averages were reported.



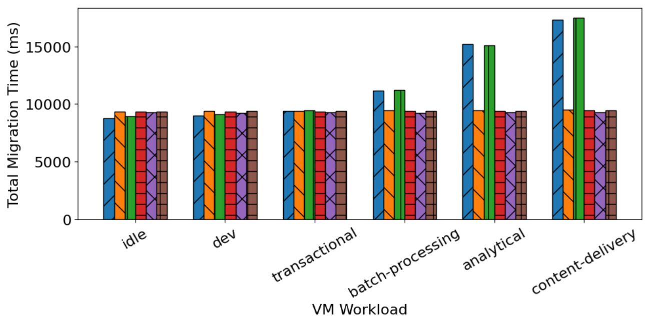
(a) Outgoing background traffic at source and destination.



(b) Incoming background traffic at source and destination.



(c) Outgoing source and incoming destination traffic.



(d) Incoming source and outgoing destination traffic.

FIGURE 2: Breakdown of background traffic directions during single VM migration.

In parallel migration, total migration time corresponds to the longest single VM migration. In serial migration, it is the cumulative time. Downtime is similarly defined as either the maximum (parallel) or sum (serial) of individual downtimes. For parallel migrations, the traffic profiles of the two VMs were classified into three categories:

- Both VMs experienced predominantly incoming application traffic (in).
- Both VMs generated mainly outgoing traffic (out).
- One VM had incoming traffic, while the other had outgoing traffic (mixed).

For *serial migrations*, where VMs were migrated one after the other, more granular traffic profile combinations were evaluated:

- Both the first and second VMs experienced incoming traffic (in-in).
- The first VM experienced incoming traffic, while the second had outgoing traffic (in-out).
- The first VM had outgoing traffic, followed by an incoming workload on the second (out-in).
- Both VMs generated outgoing traffic (out-out).

In addition to the VM-level traffic profiles, background traffic between the source and destination hosts was also categorized to evaluate the impact of host-level contention. These *source-destination traffic combinations (src-dest)* were classified as,

- Both the source and destination hosts experienced predominantly incoming background traffic (in-in).
- The source host received incoming traffic while the destination host generated outgoing traffic (in-out).
- The source host generated outgoing traffic, and the destination host received incoming traffic (out-in).
- Both the source and destination hosts generated outgoing traffic (out-out).

1) High-Priority Migrations

Table 1 compares total migration times for high-priority migrations. The traffic-aware algorithm uses parallel migration in this scenario and consistently achieves the lowest migration time across all traffic configurations and VM directions.

2) Low-Priority Migrations

Table 2 presents downtime comparisons for low-priority migrations. The traffic-aware algorithm uses serial migration to minimize application impact, resulting in consistently lower downtimes than traditional methods in most cases. For instance, in the in-out case, downtime is reduced to 21 ms compared to 183 ms (hybrid) and 530 ms (post-copy).

3) Medium-Priority Migrations

For medium-priority VMs, the traffic-aware strategy balances migration speed and downtime. It selects between serial and parallel approaches based on current network conditions and workload impact. Migration times were higher than

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

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

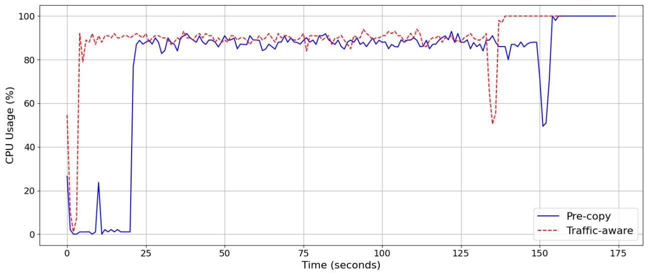
TABLE 2: Downtime (ms) for 2 serial VM migrations under low-priority settings.

high-priority cases but lower than low-priority ones. Downtime was similarly positioned—better than high-priority but slightly worse than low-priority scenarios. These results demonstrate the algorithm's adaptive nature in achieving an optimal trade-off between performance and service continuity.

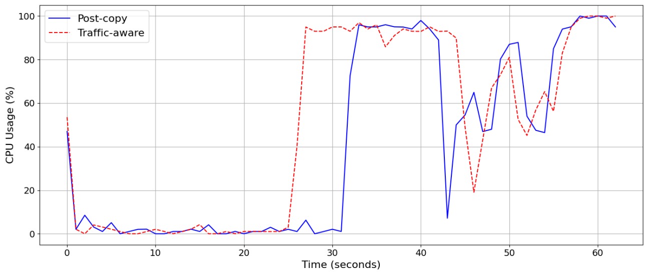
D. CPU UTILILZATION

To evaluate the impact of different migration strategies on application performance, we analyzed the CPU utilization of a VM executing a `QuickSort` [36], 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.



(a) Pre-copy vs traffic-aware.



(b) Post-copy vs traffic-aware.

FIGURE 3: CPU usage during QuickSort execution across migration strategies.

For the post-copy comparison, a VM with primarily inbound traffic was migrated, while outbound background traffic was generated at the source server. 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 3, 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.

VI. RELATED WORK

This section reviews prior work in traffic-sensitive live VM migration, highlighting a range of strategies designed to improve migration efficiency and reduce network impact.

Nevertheless, the combined integration of traffic-awareness and urgency-based prioritization remains relatively underexplored, offering opportunities for further enhancement of network performance during live migrations.

Traffic-Aware Migration: Migration strategies have been widely studied to enhance VM and container migration by reducing network traffic and improving resource efficiency. Several works [37]–[39] focus on optimizing migration decisions by considering network topology, inter-VM communication, and traffic patterns. Others [40]–[43] extend traffic-awareness to load balancing and congestion control using various optimization and scheduling techniques. Furthermore, recent approaches [17], [33], [44] target adaptive migration strategies by monitoring real-time traffic conditions and dynamically selecting suitable algorithms. Collectively, these methods emphasize minimizing network impact, improving migration performance, and ensuring efficient resource utilization in dynamic environments.

Priority-Aware Migration: Several studies have addressed live migration by incorporating task or VM priorities to improve decision-making. Solive et al. [24] proposed a bandwidth reservation mechanism based on migration urgency, while others [19], [45] focused on selecting VMs for migration based on task priority or requested migration time. Additional works [46], [47] considered migration deadlines and request timing constraints to optimize placement and routing, aiming to balance load while meeting service-level requirements. These approaches enhance migration efficiency by aligning resource allocation with task criticality.

Traffic Shaping: Traffic shaping has been extensively employed to optimize VM migration by managing bandwidth and network flow dynamics. Approaches such as the mVM Scheduler [48] and SDN/NFV-based solutions [49] leverage real-time traffic monitoring, route adjustments, and prioritization of critical flows to minimize congestion and improve migration timing. Additionally, bandwidth adaptation models [50], [51], including geometric programming and adaptive pre-copy techniques, dynamically allocate bandwidth across migration phases to reduce downtime, balance resource usage, and support efficient multi-VM migration under varying network conditions.

VII. 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.

Effectiveness of the Traffic-Aware Approach
The empirical evaluation demonstrates that the proposed approach consistently outperforms traditional migration techniques in terms of both total migration time and downtime, particularly under high and low priority settings. For high-priority migrations, the traffic-aware strategy leverages parallelism to achieve significant reductions in total migration time, without severely compromising application performance. In contrast, for low-priority cases, the algorithm uses serial migration and prioritizes application traffic, leading to drastically lower downtimes. This dual-mode adaptability addresses the competing needs of responsiveness and stability, which are often at odds in cloud management.

The empirical bandwidth models further enhance the system's responsiveness, allowing the migration controller to make nuanced, traffic-sensitive decisions without resorting to static thresholds or hardcoded rules. These models capture the impact of different traffic directions and intensities on bandwidth availability, improving the precision of bandwidth estimation and reservation.

Priority-Awareness and SLA Compliance
A key innovation of this work is its integration of migration urgency and SLA bounds into the bandwidth reservation process. The algorithm ensures that high-priority migrations are not delayed due to limited bandwidth, while medium- and low-priority migrations are only permitted to use bandwidth in a manner that does not violate the SLA guarantees of co-located VMs. This hierarchical treatment of priorities introduces a principled way to balance fairness, efficiency, and service quality, particularly in multi-tenant environments where resource sharing is inevitable.

By embedding SLA constraints into decision-making, the approach also supports more reliable QoS provisioning. The system effectively avoids overcommitting network resources in a way that would degrade application responsiveness or throughput, thereby enhancing user satisfaction and reducing the risk of SLA penalties for cloud providers.

Comparative Advantage Over Existing Techniques
Compared to traditional pre-copy, post-copy, and hybrid-copy migration techniques, the traffic-aware algorithm offers two distinct advantages:

- Context-Sensitive Strategy Selection: Unlike existing methods that apply a uniform migration technique regardless of context, the proposed system selects between pre-copy and post-copy based on real-time traffic conditions at both source and destination hosts. This ensures that the chosen technique is always the most suitable for the prevailing conditions.
- Adaptability to Multi-VM Scenarios: In contrast to many prior works, which assume uniform urgency and behaviour across VMs, this work accounts for heterogeneous workloads and urgency levels. The multiple VM migration algorithm incorporates traffic-aware decisions at a per-VM level, making it more scalable and suitable for realistic cloud workloads.

Practical Implications and Deployment Feasibility
The system was implemented using widely available tools such as `tc` and `ifstat`, ensuring that it is deployable in real-world data center environments without requiring specialized hardware or proprietary software. This ease of adoption is

crucial for cloud providers who seek performance improvements without major architectural overhauls.

Moreover, the model-based bandwidth estimation allows the algorithm to react quickly to changes in traffic load, making it suitable for dynamic cloud environments where traffic patterns can shift unpredictably. The integration with SLA policies also supports enterprise-grade service management, which is increasingly important as cloud services cater to mission-critical workloads.

While the proposed approach demonstrates strong performance in LAN-based environments, it does not currently account for wide-area migrations where network latencies and variances are more pronounced. Extending the algorithm to support WAN migrations, possibly by incorporating latency-sensitive SLA models, represents a promising area for future research.

Additionally, while the algorithm accounts for SLA constraints at the source host, it does not evaluate potential contention at the destination beyond bandwidth availability. Future enhancements could incorporate destination-level SLA considerations and VM interdependencies post-migration, which would help prevent performance degradation after placement.

Another potential direction is the use of predictive models or machine learning to improve traffic classification and bandwidth estimation. Such models could learn traffic behaviour over time and proactively optimize migrations before contention becomes critical.

VIII. CONCLUSION

This paper 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

[1] M. Bahrami, A. T. Haghighat, and M. Gholipoor, *A review of virtual machine migration techniques in data center*. [Online]. Available: https://www.researchgate.net/publication/372409749.

[2] U. Deshpande, X. Wang, and K. Gopalan, "Live gang migration of virtual machines," in *Proceedings of the 20th international symposium on High performance distributed computing*, 2011, pp. 135–146.

[3] T. Le, *A survey of live virtual machine migration techniques*, Nov. 2020. DOI: 10.1016/j.cosrev.2020.100304.

[4] C. Jo, E. Gustafsson, J. Son, and B. Egger, "Efficient live migration of virtual machines using shared storage," *ACM Sigplan Notices*, vol. 48, no. 7, pp. 41–50, 2013.

[5] T. Wood, K. Ramakrishnan, P. Shenoy, *et al.*, "Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines," *IEEE/ACM Transactions On Networking*, vol. 23, no. 5, pp. 1568–1583, 2014.

[6] P. Padala, K. G. Shin, X. Zhu, *et al.*, "Adaptive control of virtualized resources in utility computing environments," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, 2007, pp. 289–302.

[7] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif, *et al.*, "Black-box and gray-box strategies for virtual machine migration.," in *NSDI*, vol. 7, 2007, pp. 17–17.

[8] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott, "Proactive fault tolerance for hpc with xen virtualization," in *Proceedings of the 21st annual international conference on Supercomputing*, 2007, pp. 23–32.

[9] L. Y. Devi, P. Aruna, N. Priya, *et al.*, "Security in virtual machine live migration for kvm," in *2011 International Conference on Process Automation, Control and Computing*, IEEE, 2011, pp. 1–6.

[10] R. Nathuji and K. Schwan, "Virtualpower: Coordinated power management in virtualized enterprise systems," *ACM SIGOPS operating systems review*, vol. 41, no. 6, pp. 265–278, 2007.

[11] L. Hu, H. Jin, X. Liao, X. Xiong, and H. Liu, "Magnet: A novel scheduling policy for power reduction in cluster with virtual machines," in *2008 IEEE International Conference on Cluster Computing*, IEEE, 2008, pp. 13–22.

[12] C. Clark, K. Fraser, S. Hand, *et al.*, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, 2005, pp. 273–286.

[13] M. Nelson, B.-H. Lim, G. Hutchins, *et al.*, "Fast transparent migration for virtual machines.," in *USENIX Annual technical conference, general track*, 2005, pp. 391–394.
[14] M. R. Hines and K. Gopalan, "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*, 2009, pp. 51–60.
[15] M. R. Hines, U. Deshpande, and K. Gopalan, *Post-copy live migration of virtual machines*.
[16] S. Sahni and V. Varma, "A hybrid approach to live migration of virtual machines," in *2012 IEEE international conference on cloud computing in emerging markets (CCEM)*, IEEE, 2012, pp. 1–5.
[17] U. Deshpande and K. Keahey, "Traffic-sensitive live migration of virtual machines," *Future Generation Computer Systems*, vol. 72, pp. 118–128, Jul. 2017, ISSN: 0167739X. DOI: 10.1016/j.future.2016.05.003.
[18] F. Callegati and W. Cerroni, "Live migration of virtualized edge networks: Analytical modeling and performance evaluation," in *2013 IEEE SDN for future networks and services (SDN4FNS)*, IEEE, 2013, pp. 1–6.
[19] H. A. Nadeem, H. Elazhary, and M. A. Fadel, "Priority-aware virtual machine selection algorithm in dynamic consolidation," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 11, 2018.
[20] W. Huang, M. J. Koop, Q. Gao, and D. K. Panda, "Virtual machine aware communication libraries for high performance computing," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, 2007, pp. 1–12.
[21] G. Sun, D. Liao, D. Zhao, Z. Xu, and H. Yu, "Live migration for multiple correlated virtual machines in cloud-based data centers," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 279–291, 2015.
[22] V. Chang, R. J. Walters, and G. Wills, "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, 2013, pp. 245–264.
[23] G. Sun, D. Liao, V. Anand, D. Zhao, and H. Yu, "A new technique for efficient live migration of multiple virtual machines," *Future Generation Computer Systems*, vol. 55, pp. 74–86, 2016.
[24] D. Fernando, P. Yang, and H. Lu, "Sdn-based order-aware live migration of virtual machines," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1818–1827. DOI: 10.1109/INFOCOM41043.2020.9155415.
[25] K. Tsakalozos, V. Verroios, M. Roussopoulos, and A. Delis, "Live vm migration under time-constraints in share-nothing iaas-clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2285–2298, 2017.
[26] K. Lu, R. Yahyapour, P. Wieder, C. Kotsokalis, E. Yaqub, and A. I. Jehangiri, "Qos-aware vm placement in multi-domain service level agreements scenarios," in *2013 IEEE Sixth International Conference on Cloud Computing*, IEEE, 2013, pp. 661–668.
[27] I. Odun-Ayo, B. Udemezue, and A. Kilanko, "Cloud service level agreements and resource management," *Adv. Sci. Technol. Eng. Syst.*, vol. 4, no. 2, pp. 228–236, 2019.
[28] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu, "Service level agreement based energy-efficient resource management in cloud data centers," *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1621–1633, 2014.
[29] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, *Cost of virtual machine live migration in clouds: A performance evaluation*, 2009. [Online]. Available: http://www.cloudbus.org.
[30] M. A. Altahat, A. Agarwal, N. Goel, and J. Kozlowski, "Dynamic hybrid-copy live virtual machine migration: Analysis and comparison," vol. 171, 2020. DOI: 10.1016/j.procs.2020.04.156.
[31] G. Soni and M. Kalra, "Comparative study of live virtual machine migration techniques in cloud," *International Journal of Computer Applications*, vol. 84, pp. 19–25, 14 Dec. 2013. DOI: 10.5120/14643-2919.
[32] Y. Kuno, K. Nii, and S. Yamaguchi, "A study on performance of processes in migrating virtual machines," in *2011 Tenth International Symposium on Autonomous Decentralized Systems*, IEEE, 2011, pp. 567–572.
[33] Y. Cui, L. Zhu, Z. Cai, and Y. Hu, "An adaptive traffic-aware migration algorithm selection framework in live migration of multiple virtual machines," *International Journal of Performability Engineering*, vol. 16, pp. 314–324, 2 2020, ISSN: 09731318. DOI: 10.23940/ijpe.20.02.p14.314324.
[34] "Linux tc." (), [Online]. Available: https://man7.org/linux/man-pages/man8/tc.8.html.
[35] "Linux ifstat." (), [Online]. Available: https://www.man7.org/linux/man-pages/man8/ifstat.8.html.
[36] D. Fernando, J. Terner, K. Gopalan, and P. Yang, "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, 2019, pp. 343–351.
[37] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *2011 Proceedings IEEE INFOCOM*, IEEE, 2011, pp. 66–70.
[38] Y. Cui, Z. Yang, S. Xiao, X. Wang, and S. Yan, "Traffic-aware virtual machine migration in topology-

adaptive dcn," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3427–3440, 2017.

[39] F. P. Tso, K. Oikonomou, E. Kavvadia, and D. P. Pezaros, "Scalable traffic-aware virtual machine management for cloud data centers," in *2014 IEEE 34th International Conference on Distributed Computing Systems*, IEEE, 2014, pp. 238–247.

[40] R. Kanniga Devi, G. Murugaboopathi, and M. Muthukannan, "Load monitoring and system-traffic-aware live vm migration-based load balancing in cloud data center using graph theoretic solutions," *Cluster Computing*, vol. 21, no. 3, pp. 1623–1638, 2018.

[41] X. Fu, C. Zhang, J. Chen, L. Zhang, and L. Qiao, "Network traffic based virtual machine migration in cloud computing environment," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, IEEE, 2019, pp. 818–821.

[42] J. Liu, Y. Li, D. Jin, L. Su, and L. Zeng, "Traffic aware cross-site virtual machine migration in future mobile cloud computing," *Mobile Networks and Applications*, vol. 20, pp. 62–71, 2015.

[43] R. Nasim and A. J. Kassler, "Network-centric performance improvement for live vm migration," in *2015 IEEE 8th International Conference on Cloud Computing*, IEEE, 2015, pp. 106–113.

[44] S. Maheshwari, S. Choudhury, I. Seskar, and D. Raychaudhuri, "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, 2018, pp. 1–6.

[45] A. Dalvandi, M. Gurusamy, and K. C. Chua, "Time-aware vmflow placement, routing, and migration for power efficiency in data centers," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 349–362, 2015.

[46] R. A. Haidri, M. Alam, M. Shahid, S. Prakash, and M. Sajid, "A deadline aware load balancing strategy for cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 1, e6496, 2022.

[47] J. Son and R. Buyya, "Priority-aware vm allocation and network bandwidth provisioning in software-defined networking (sdn)-enabled clouds," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 17–28, 2018.

[48] V. Kherbache, E. Madelaine, and F. Hermenier, "Scheduling live migration of virtual machines," *IEEE transactions on cloud computing*, vol. 8, no. 1, pp. 282–296, 2017.

[49] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni, and A. Campi, "Clouds of virtual machines in edge networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 63–70, 2013.

[50] W. Cerroni and F. Esposito, "Optimizing live migration of multiple virtual machines," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1096–1109, 2016.

[51] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, E. S. Pilli, and D. Kapil, "A critical survey of live virtual machine migration techniques," *Journal of Cloud Computing*, vol. 6, no. 1, pp. 1–41, 2017.