Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Towards smarter live migration: Minimizing SLO violations and costs

Youngsu Cho [a], Changyeon Jo [a], Reza Entezari-Maleki [b], Jörn Altmann [a], Bernhard Egger [a,*]

[a] Seoul National University, 1, Gwanak-ro Gwanak-gu, 08826, Seoul, South Korea
[b] School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran

ARTICLE INFO

Keywords:
Live migration
Data center management
Service-level objective
Machine learning
Performance estimation.

ABSTRACT

Data centers employ live virtual machine (VM) migration to optimize resource usage while ensuring continuous execution of guest operating systems. Given the current resource utilization, sophisticated algorithms determine when and where to migrate which VMs. Surprisingly little attention, however, is given to selecting the appropriate migration technique based on specific host and guest workload characteristics. This work first shows that relying on a single live migration algorithm leads to significantly more Service-Level Objective (SLO) violations and higher resource usage than adaptively selecting the most suitable migration algorithm. Building on this observation, we then present an intelligent live migration framework that selects the most appropriate live migration algorithm based on SLOs and operational cost factors, using a multi-objective optimization approach. Through a comprehensive evaluation across diverse hotspot and consolidation scenarios, we demonstrate that the presented framework is able to substantially reduce SLO violation while optimizing key operational metrics. The framework reduces the total migration time by a factor of 1.5 and decreases SLO violations by nearly an order of magnitude compared to the predominantly used pre-copy method. Moreover, it achieves near-optimal VM migration technique selection compared to an Oracle under varying workload conditions. The results indicate that intelligent selection of live migration algorithms can significantly enhance both application performance and resource efficiency in virtualized environments.

1. Introduction

Virtualization has become integral to cloud computing, enabling data centers to efficiently manage resources while ensuring tenant isolation. Major cloud service providers, including Amazon Web Services [1], Microsoft Azure [2], Google Cloud [3], IBM Cloud [4], or Alibaba Cloud [5], rely on virtual machines (VMs) to deliver scalable and flexible Infrastructure as a Service (IaaS) offerings.

Live migration refers to the process of transparently transferring active VMs between physical hosts without interrupting the execution of the VM. Live migration plays a crucial role in data center operations for routine maintenance, load balancing, and resource consolidation to reduce the Total Cost of Ownership and energy consumption of a data center [6–9]. For instance, Google reports migrating over a million VMs monthly for hardware maintenance and software updates [10,11].

Despite its advantages, live migration is predominantly used for maintenance in commercial data centers [12]. A broader adoption for load balancing and energy optimization remains limited due to several challenges. Firstly, VM performance can be temporarily impacted during migration, potentially violating Service-Level Agreements (SLAs) such as throughput and availability guarantees [13]. Secondly, migrations generate additional network traffic and CPU load in the data center, incurring short-term financial costs for operators despite long-term energy savings and resource utilization benefits [14,15]. Consequently, the initial VM placement becomes critical [16–20], yet static placement strategies often fail to adapt to fluctuating workloads, resulting in underutilized resources. Studies have shown that CPU utilization in data centers can range between 10 and 50 percent over extended periods [21].

Existing VM management systems typically employ default migration techniques such as pre-copy [22] or post-copy [23], without considering the specific workload characteristics or system conditions. Recent research suggest that dynamically selecting migration techniques based on workload characteristics can significantly enhance live migration efficiency [24–31]. Several guidelines have been proposed to select the best-suited live migration technique considering the VM workload and resource utilization of the host [10,32–37]. However, current solutions often lack comprehensive automation, failing to integrate workload

* Corresponding author.
E-mail addresses: youngsu@csap.snu.ac.kr (Y. Cho), changyeon@csap.snu.ac.kr (C. Jo), entezari@iust.ac.ir (R. Entezari-Maleki), jorn.altmann@acm.org (J. Altmann), bernhard@csap.snu.ac.kr (B. Egger).

https://doi.org/10.1016/j.future.2025.108085

detection, SLO impact prediction, and migration execution into a unified framework.

To address these limitations, we present an innovative VM management framework that intelligently selects the most suitable live migration algorithm by analyzing the VM workload demands, SLO constraints, and overall data center conditions. Our approach employs machine learning models trained on extensive migration data to predict the optimal migration strategy that maximizes the economic benefits for both data centers and customers through improved VM efficiency and fewer SLO violations. Our approach is based on and extends state-of-the-art live migration models [35,38], which predict various migration performance metrics for all supported live migration techniques in the QEMU/KVM hypervisor, and further introduces a novel weighted combined score metric to facilitate multi-objective optimization.

In this work, we demonstrate and evaluate our method using the QEMU/KVM hypervisor. QEMU/KVM was selected due to its open-source nature, wide adoption in both academia and industry, and its flexible support for a variety of live migration techniques. However, we emphasize that the proposed framework is not limited to QEMU/KVM and can be implemented on other hypervisors such as VMware vSphere, OpenStack or Xen, as it relies on standard migration metrics and interfaces that are available in most modern virtualization platforms. Compared to our prior work [35], we introduce four significant advancements: (1) expanded support for migration techniques including the hybrid PRE-POST method; (2) enhanced framework capabilities addressing both hotspot resolution and consolidation scenarios; (3) improved prediction models trained on an expanded dataset of 55,000+ migrations; and (4) comprehensive evaluation across all migration techniques under diverse operational conditions.

To demonstrate the efficacy of our framework, we implement a cluster-wide VM scheduler that responds to load-balancing and consolidation triggers. Unlike static approaches relying on a single, fixed migration technique, our scheduler dynamically selects the most suitable method based on real-time workload and system characteristics. An evaluation across diverse workloads and SLO constraints shows that our approach significantly reduces SLO violations and improves overall resource utilization.

In summary, the contributions of this work are as follows:

- The development of a live migration performance model capable of predicting optimal migration techniques based on multiple objectives.
- The implementation of a management framework that utilizes the performance model to automate VM migrations for load balancing and server consolidation.
- A comprehensive evaluation that demonstrates significant reductions in SLO violations and resource usage, achieving near-optimal performance compared to traditional single-technique approaches.

The remainder of this paper is organized as follows. Section 2 provides background on live migration and summarizes relevant related work. Section 3 outlines the motivation behind our approach. Section 5 describes the design and implementation of the presented VM live migration orchestration framework. Section 6 presents and discusses the experimental evaluation. Finally, Section 7 concludes this paper.

2. Background and related work

2.1. Live migration

Live migration refers to the process of transferring a virtual machine (VM) from one physical host to another while keeping the guest OS operational. Live migration requires moving the VM's volatile memory data, vCPU execution context, and other necessary runtime states without halting its operation. As modern VMs often have memory footprints of several tens of gigabytes, the memory transfer constitutes the primary overhead during live migration [39].

The process of live migrating a VM from a source node to destination node can be separated into three distinct phases (Fig. 1): *prepare*, *stop-and-copy*, and *resume*. Depending on the selected migration technique, the actions and the length of the phases vary considerably.

In the *prepare* phase, a new VM is created on the destination host. Many techniques transfer the VM's memory partially or entirely to the destination node while the VM is still executing on the source node. During the *stop-and-copy* phase, the VM is stopped on the source node, the necessary execution context – such as CPU registers and device contexts – is transferred to the destination node, and then the VM is restarted on the destination node. In the *resume* phase, the remaining parts of the VM's volatile execution state – such as main memory that is still residing on the source node – are transferred to the destination node. The *resume* phase ends with the destruction of the VM on the source node.

2.2. Live migration techniques

This section outlines the key live migration techniques implemented in widely used hypervisors such as QEMU/-KVM [40] and Xen [41],

2.2.1. Basic transfer algorithms

Live migration techniques are commonly classified by the timing and strategy of the memory transfer. The most fundamental approach is *pre-copy* (**PRE**), where all VM memory is transferred to the destination before finalizing the migration [22]. During this process, dirty memory pages are copied iteratively while the VM continues executing on the source. The VM is paused once the dirtying rate drops below a threshold – typically when the network bandwidth exceeds the page update rate – or after a fixed number of iterations. The remaining dirty pages and the processor and device states are transferred, then the VM is resumed on the destination host. Since the entire VM state is already in place, the *resume* phase is short.

In contrast, *post-copy* (**POST**) migration begins by transferring only the minimal execution context and immediately resumes the VM on

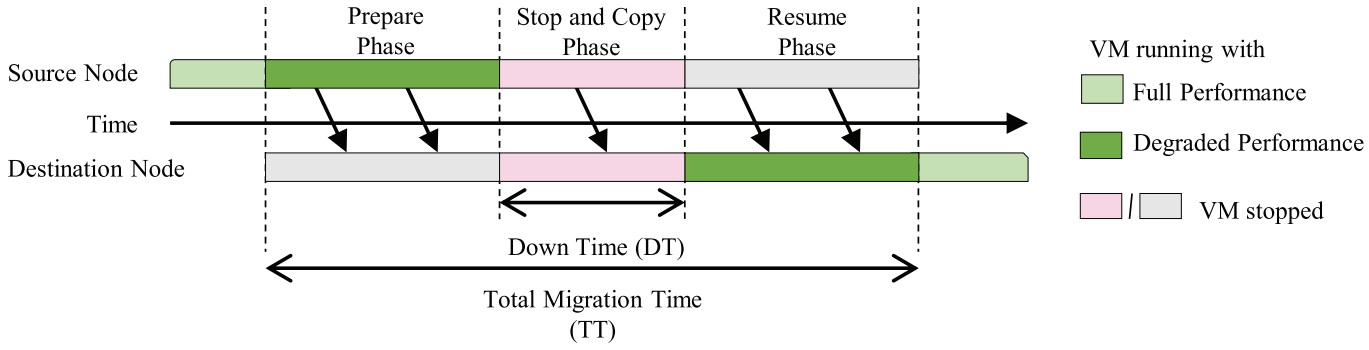

Fig. 1. Live migration phases.

Table 1
Overview of strengths and weaknesses of the live migration techniques across six performance metrics. A ✓ indicates strong performance for a given metric, while a ✗ indicates performance metrics where the technique typically struggles.

	Total migration Time (TT)	Downtime (DT)	Transferred data volume (TD)	Performance degra- dation (PERF)	Additional CPU usage (CPU)	Additional memory usage (MEM)
Pre-Copy (PRE)						
Post-Copy (POST)	✓	✓	✓	✗		
PRE-POST	✓	✓	✓	✗		
CPU Throttling (THR)				✗		
Delta Compression (DCdlt)			✓			✗
Zstd Compression (DCZstd)			✓		✗	
DCdlt-DCZlib			✓		✗	✗
DCZlib-Async			✓		✗	
DCdlt-DCZlib-Async			✓		✗	✗

the destination host [42]. During the *resume* phase, the VM memory is fetched from the source on demand as pages are accessed. Because each page is transferred only once, *post-copy* can reduce the total migration time, but page faults during execution can cause severe performance degradation.

To combine the strengths of both methods, **PRE-POST** performs a single iteration of pre-copy during which most of the memory is transferred. The VM is then resumed on the destination, any remaining memory pages are retrieved on demand. This hybrid strategy reduces performance degradation and resume time relative to *post-copy*, while also avoiding the overhead of multiple pre-copy rounds required by traditional *pre-copy* migration.

2.2.2. Optimizations

The presented framework builds on the KVM/QEMU virtualization machine manager (VMM) [43], which supports several orthogonal live migration optimizations aimed at improving performance or reducing the volume of transferred data. These include CPU throttling and various forms of memory compression. While effective, such techniques often require additional CPU or memory resources.

CPU throttling (THR) reduces the virtual CPU speed to slow down the rate at which memory is modified, thereby decreasing the amount of data transferred during each iteration in the *prepare* phase. Throttling is typically only activated after several iterations if the number of dirty pages remains high. This optimization can reduce both total migration time and downtime but may cause significant performance degradation within the VM. As a result, it is typically avoided in scenarios with strict performance SLOs.

Compression-based optimizations apply various algorithms to reduce memory volume before transfer. **Delta compression (DCdlt)** tracks changes to memory pages between pre-copy iterations and transfers only the differences. It is computationally light but memory-intensive, as it requires maintaining copies of modified pages. **Zlib compression (DCZlib)** and **Zstd compression (DCZstd)** [44] use general-purpose compression algorithms to shrink the memory footprint. These techniques require considerable CPU resources on the source host and may be impractical when the migration is triggered by resource pressure.

Additional variants refine or combine these techniques. **DCdlt-DCZlib** applies Zlib compression to memory pages that have first been delta-compressed. **DCZlib-Async** and **DCdlt-DCZlib-Async** eliminate synchronization between the compression thread and pre-copy iterations, allowing the compression process to operate asynchronously. As a result, the compression thread can begin compressing re-dirtied pages without waiting for the current pre-copy round to complete.

2.3. Live migration metrics

Throughout this paper, we utilize the following six metrics to evaluate the performance of VM live migration:

1. **Total migration time (TT)**: The elapsed time from the start of the migration until the VM is fully operations on the destination host.
2. **Downtime (DT)**: The duration of the *stop-and-copy* phase, i.e., the period during which the VM is paused.
3. **Transferred data volume (TD)**: The total amount of data transferred from the source to the destination during migration.
4. **Performance degradation (PERF)**: The impact of migration on VM performance, measured as the deviation from a guaranteed instructions-per-second (IPS) baseline.
5. **Additional host CPU utilization (CPU)**: The increase in CPU usage on the source host caused by the migration process.
6. **Additional memory utilization (MEM)**: The amount of memory consumed on the source by the live migration algorithm.

Table 1 summarizes the nine combinations of transfer algorithms and optimizations evaluated in this work, highlighting their strengths and weaknesses with respect to the six performance metrics. While additional migration optimizations have been proposed in prior work [23–30,32], they are not supported by current mainstream hypervisors and are thus excluded from our evaluation.

2.4. Related work

Live migration techniques have evolved significantly over the past decades, creating the challenge of selecting appropriate approaches for specific operational scenarios. Several studies [33,34,45] have proposed heuristic-based selection guidelines, but these lack systematic implementation frameworks for automated decision-making in production environments. Concurrently, researchers have developed analytical and machine learning models to predict migration performance. While analytical approaches provide theoretical insights, they require complex calibration and often fail to adapt to dynamic data center conditions.

Jo et al. [38] presented a significant advancement with their machine learning-based prediction model trained on 40,000 migrations across diverse workloads. Their model predicts six performance metrics for five migration techniques, demonstrating that appropriate technique selection reduces SLO violations. Our work extends this foundation by improving prediction accuracy and supporting additional migration techniques.

Sandpiper [6], one of the earliest dynamic VM migration systems, implements hotspot mitigation through periodic resource utilization profiling and time series prediction. It selects VMs for migration using a heuristic algorithm that prioritizes based on resource footprints (CPU, memory, network utilization). However, Sandpiper relies exclusively on pre-copy migration and estimates costs primarily on VM memory size, limiting optimization across diverse workloads.

CloudScale [7] offers a framework for dynamic VM scaling and resource conflict resolution through migration. It incorporates predictive resource demand modeling to prevent SLO violations proactively. While considering migration overhead to minimize disruptions, CloudScale employs a simplistic linear regression model for cost estimation, which inadequately captures the non-linear relationships between VM

characteristics and migration performance, particularly for memory-intensive workloads [38,46,47].

CAMIG [48] addresses concurrent migration challenges by considering resource dependencies among multiple VMs, using maximal cliques and independent sets to minimize interference. Unlike our approach that focuses on technique selection for individual migrations, CAMIG optimizes the scheduling order of multiple simultaneous migrations without considering the specific migration technique used for each VM.

Belgacem et al. [49] propose a model that applies machine learning to optimize both VM selection and the migration process itself. Their approach focuses primarily on reducing migration frequency and energy consumption by considering various cloud environment attributes. While their work emphasizes when to migrate and which VMs to select, our framework concentrates on how migrations should be performed through optimal technique selection across multiple available methods.

Haris et al. [50] employ machine learning to predict downtime in pre-copy live migration, dynamically terminating iterations when predicted downtime falls below a threshold. While their approach optimizes a single migration technique, our framework selects from multiple techniques and considers a broader range of metrics beyond just downtime.

Gong et al. [51] explore deep reinforcement learning for optimizing VM migration decisions in dynamic resource allocation scenarios. Their theoretical investigation focuses primarily on when to migrate VMs, whereas our work implements and evaluates a practical framework that determines how to perform migrations through optimal technique selection.

3. Challenges and opportunities in live migration

The analysis of the Google cluster trace datasets [52–54] provides detailed insights into resource usage in large-scale cloud environments. A key observation from these datasets is the highly dynamic nature of resource utilization, caused by a diverse mix of short- and long-running jobs with varying demands. This workload volatility leads to frequent transitions between underutilized and overutilized states across nodes.

Fig. 2 shows the results of our simulation based on the publicly available Google cluster trace data, which covers a large-scale production environment.

To avoid performance degradation during peak demand, data centers often apply conservative policies that prevent overcommitting resources, resulting in frequent periods of underutilization during normal operation. Fig. 2 illustrates the resource usage of consolidating four random nodes into one and classifying them as overutilized (resource usage over 70 %) or underutilized (below 50 %). The number of over- and underutilized nodes fluctuates significantly over short intervals. These results demonstrate the need for live migration to enable continuous and adaptive VM placement.

Despite its potential, live migration is often used with a fixed migration technique, without considering the workload characteristics of the VM or the state of the host. This is suboptimal, as no single method performs best under all conditions. For example, compression-based approaches are ineffective for incompressible memory, and different VM workloads or host states – such as CPU load or network bandwidth – require different trade-offs. Recent work [33,38,45] shows that selecting the appropriate migration technique can significantly reduce the migration overhead.

Modern orchestration frameworks do not support dynamic selection of migration techniques. Even Google, which supports both `pre-copy` and `post-copy` [10], does not automatically choose between them based on workload or system characteristics, mostly because the conditions under which one technique outperforms another are not well understood.

Fig. 3 demonstrates the impact a chosen technique can have on VM performance using a moderately CPU- and memory-intensive workload: Pre-copy (`PRE`) sustains stable performance but requires significantly more time to complete the migration (42s). With post-copy (`POST`), the migration completes after 25s; however, the VM experiences a severe performance degradation, particularly immediately after being restarted on the destination node (timestamp 6–13s).

To fully exploit the benefits of live migration, data centers require an intelligent framework that dynamically selects the most suitable technique based on current workload behavior, system state, and SLO constraints.

In addition, modern data centers are often composed of heterogeneous nodes with varying hardware capabilities (e.g., CPU architecture, core count, memory bandwidth). This heterogeneity further complicates live migration, as the performance of a migrated VM may differ significantly depending on the destination node's characteristics. Accurately predicting the impact of migration in such environments requires models that can capture and utilize detailed performance metrics from both the source and destination nodes.

4. Predicting live migration performance

Selecting the optimal migration technique is challenging due to the large number of available techniques, the diversity of VM workloads, and the variability in system and network conditions. Constructing analytical models is impractical, especially at the scale required for modern data centers.

To address this, we employ machine learning to predict the performance of different migration techniques. Building on our prior work [38], we extend the model to support nine live migration techniques and six key performance metrics. These predictions form the basis of our dynamic migration strategy, enabling accurate, real-time selection of the most suitable technique for each migration event.

4.1. Problem formulation

We formulate technique selection as a multi-output regression problem. Given a set of features characterizing the VM workload and system state, the model predicts the expected performance of each migration technique across six metrics. These predictions allow the framework to select the technique that best satisfies the current SLOs.

The model inputs (features) fall into four categories:

- VM configuration: memory size, vCPU count
- Memory access behavior: page dirty rate, spatial/temporal locality
- Host resource utilization: CPU and memory load on source and destination
- Network conditions: available bandwidth, latency

These features are collected through lightweight profiling immediately before migration. The model outputs are the six performance metrics defined in Section 2.3: `TT`, `DT`, `TD`, `PERF`, `CPU`, and `MEM`.

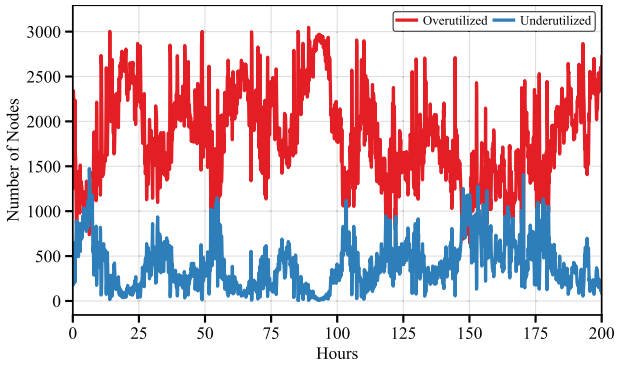

Fig. 2. Simulation of node consolidation using Google trace data [52]. Four nodes are consolidated into one node and classified as over- (>70 %) or underutilized (<50 % resource utilization) based on the trace, which spans 29 days of production workloads.

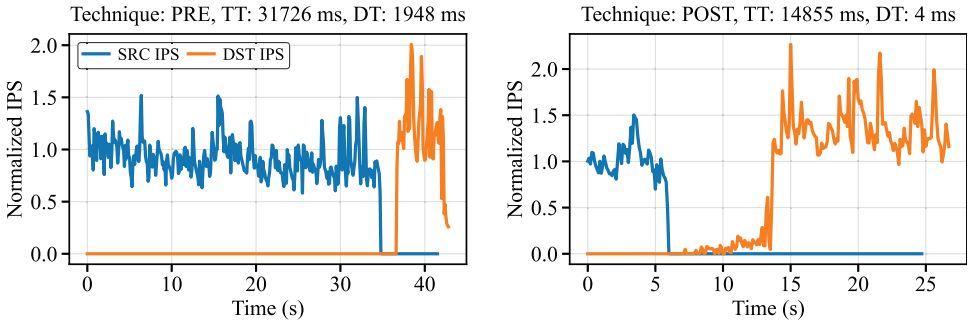

Fig. 3. Effect of live migration technique selection on VM performance, in instructions per second (IPS). The figures plot the results for a workload migrated with the PRE and POST technique, respectively. Migration is initiated at timestamp $0s$ in both figures. With PRE, the VM is able to sustain relatively stable performance, both during and after migration. With POST, on the other hand, performance collapses almost completely immediately after the VM is restarted on the destination host (timestamp $6 - 13s$). Both experiments were performed by migrating the same VM under identical conditions on our testbed.

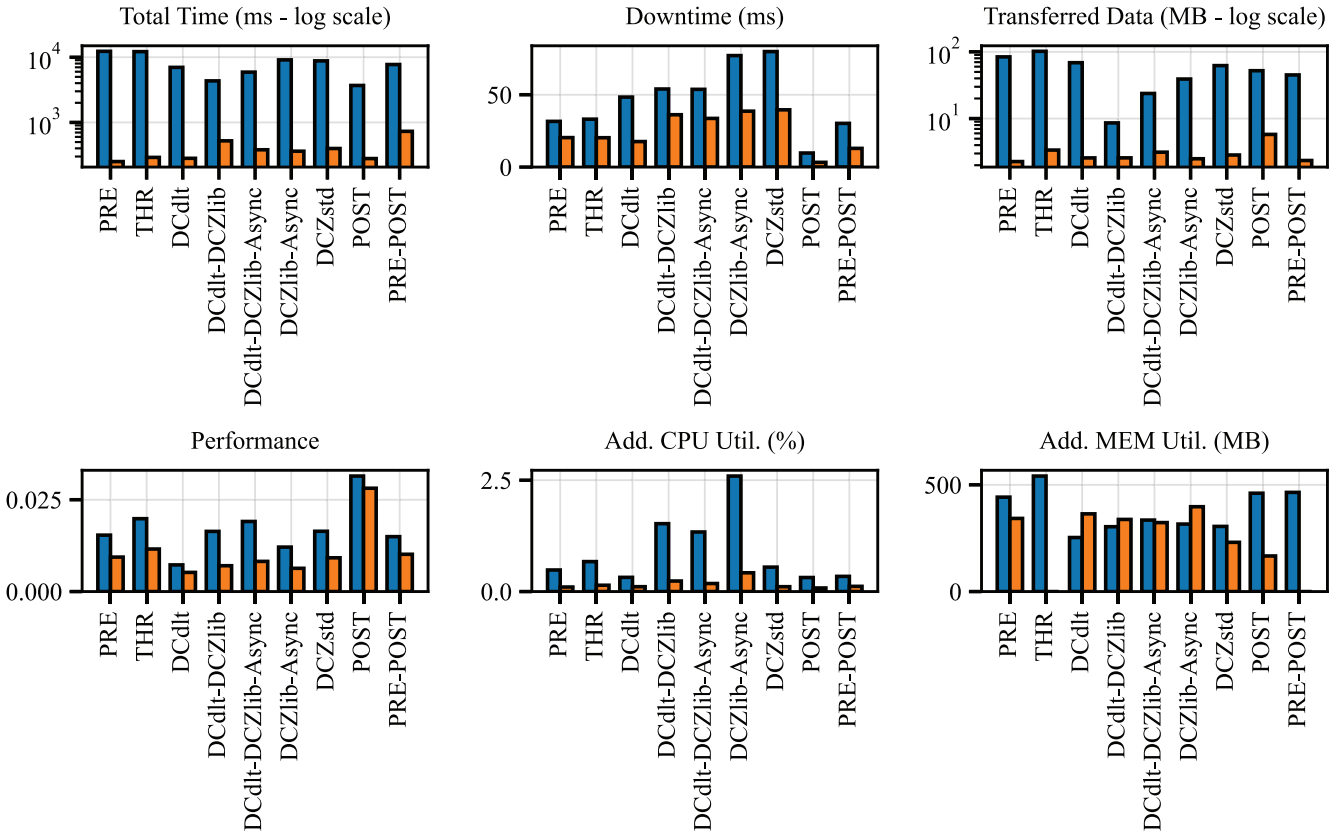

Fig. 4. 10-fold cross validation of the SVR. Bagg and ExtraTrees model (Geo-mean absolute error).

4.2. Model evaluation and selection

We extend our prior work [38], which supports five techniques using SVR with bagging, by scaling the model to support nine techniques and evaluating several machine learning algorithms: Support Vector Regression (SVR), Random Forests, Gradient Boosting, and Extremely Randomized Trees (ExtraTrees). Our evaluation uses 10-fold cross-validation over a dataset of 55,000 migrations and compares models using normalized Root Mean Squared Error (RMSE), training time, and inference time.

ExtraTrees outperforms the alternatives and is selected for deployment. Compared to SVR with bagging, ExtraTrees excel in

- Prediction accuracy: ExtraTrees achieves the lowest average RMSE across all metrics, with improvements of up to 15.4 % over SVR at a 95 % confidence level [55]. The gains are most pronounced for PERF and DT.
- Training time: ExtraTrees has a training complexity of $O(N\,D\log N)$ versus $O(N^3)$ for SVR [56], and is empirically about twice as fast to train.
- Inference speed: ExtraTrees achieves average inference times 11x faster than SVR, with $O(\log N)$ complexity compared to SVR's $O(K\,D)$ [57].

Fig. 4 shows cross-validation results for SVR and ExtraTrees on the training dataset. Each record in the dataset includes VM characteristics obtained through black-box profiling, the state of involved hosts, the applied migration technique, and observed performance metrics. The dataset includes both synthetic and real-world workloads covering a broad parameter space.

Each migration technique and metric combination is modeled independently to capture technique-specific relationships between features and outcomes. This results in improved prediction accuracy over a unified model.

4.3. Model robustness and adaptability

The training dataset includes a wide range of synthetic and real-world benchmarks, enabling the model to make consistently accurate predictions – especially with respect to *relative performance differences* between migration techniques. This level of accuracy is sufficient for selecting the technique that minimizes SLO violations and maximizes resource efficiency, as shown in Section 6.

While the absolute prediction accuracy may degrade for edge cases or unusual workload behaviors not represented in the training set, in practice, this limitation is not critical. VM workloads in data centers tend to exhibit relatively stable characteristics. When new workload types emerge, they can be incorporated into the training set to improve prediction accuracy over time. Standard machine learning approaches, such as transfer learning, can be employed to reduce the overhead of adapting the model to new workload domains, ensuring continued accuracy with minimal retraining effort.

By collecting these metrics from both the source and destination nodes, the model can learn to predict the performance impact of migration even when the underlying hardware differs. This enables accurate estimation of SLO violations and migration overheads in heterogeneous environments, as demonstrated in our evaluation.

In particular, compression is applied during the prepare phase on the source node, and decompression on the destination node incurs negligible overhead. This ensures that the destination node's performance is not adversely affected during migration.

5. Framework design and implementation

This section presents the design and key components of our framework. The system consists of a single orchestrator managing multiple worker nodes within a co-located rack. Each worker runs an agent that enables coordinated VM management and reports local resource metrics.

5.1. Framework architecture

Fig. 5 illustrates the system architecture. The framework runs on a QEMU/KVM hypervisor and consists of two main components: the orchestrator and the node-level agents. The orchestrator includes a cluster manager, two schedulers, a migration technique selector, and performance prediction models. Agents operate as background daemons on each worker node, handling VM lifecycle events and reporting local resource usage to the orchestrator.

5.1.1. Components

The **workload scheduler** manages VM deployment based on incoming workload requests. It uses cached resource availability data that is updated continuously during provisioning to place VMs on nodes with

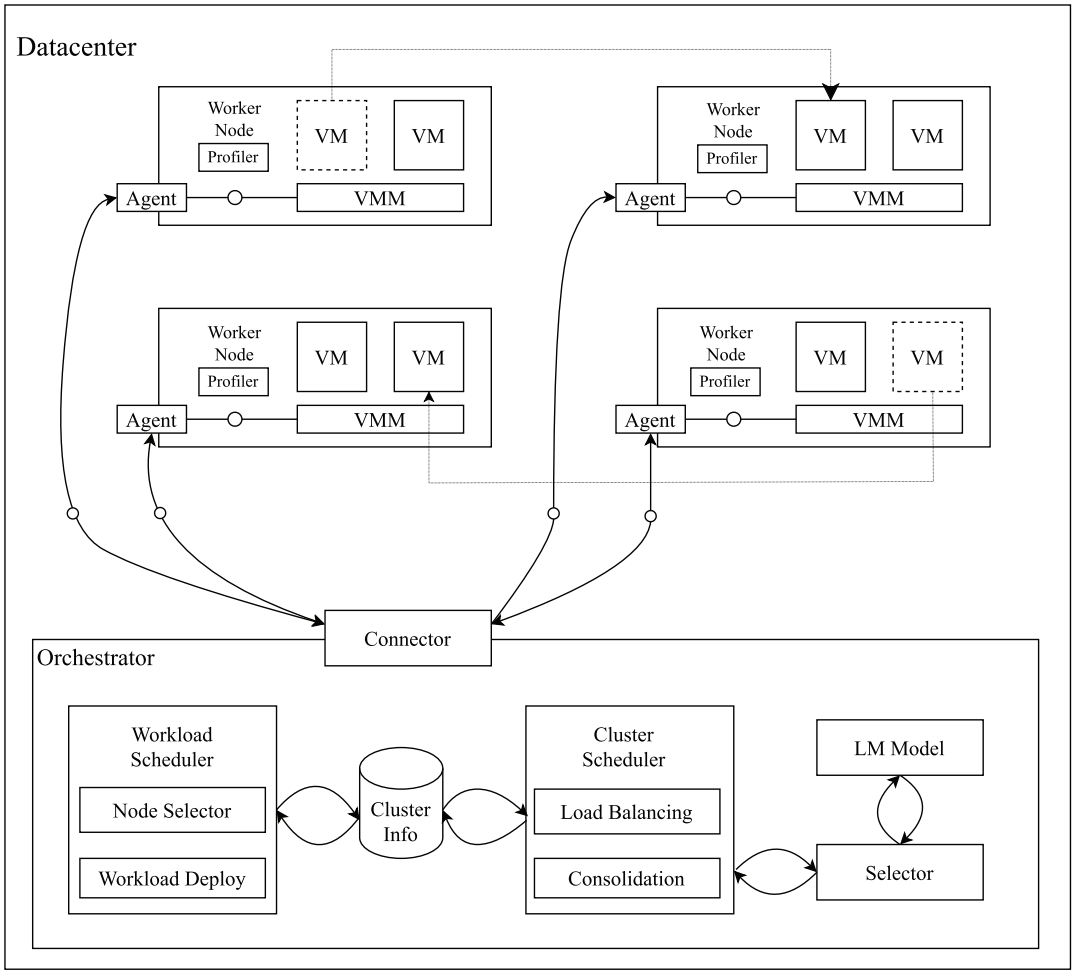

Fig. 5. High-level organization of the framework.

sufficient capacity. After selecting a target node, the scheduler instructs the cluster manager to create the VM and launch the specified workload. The **cluster scheduler** ensures balanced resource usage across nodes by periodically polling resource metrics. It identifies overloaded nodes (*hotspots*) and underutilized nodes, and triggers migrations to redistribute workloads. For hotspot mitigation, the scheduler selects the node with the lowest resource usage as the migration destination. **Node selection** for migration is based on a configurable policy, which may use random selection, round-robin scheduling, or dedicated fallback nodes. In hotspot scenarios, the default policy typically selects the least-utilized node to maximize balancing efficiency. The **migration event monitor** is configured with thresholds for CPU and memory usage, profiling intervals and durations, and selection metrics for candidate VMs. These include memory write rates, CPU utilization, and network usage. Profiling ensures that the VM with the largest resource footprint is selected as the migration source.

Following the approach introduced in Sandpiper [6], our framework uses a resource volume metric to select VMs for migration. Algorithm 1 shows how the cluster scheduler generates migration plans. The function `getTheLargestVM()` (line 11) selects the VM with the largest resource footprint, using the definition from [6]. The time complexity of Algorithm 1 is $O(n + v * t)$, where n is the number of nodes, v the number of VMs on the hottest node, and t the number of available migration techniques. The algorithm first performs a linear scans across nodes to detect the hottest and the coldest node (lines 2–8). The largest VM on the hottest node is selected for migration, and the best migration technique determined (Section 5.2.3). For consolidation, the scheduler determines whether the current workloads can be packed onto fewer physical nodes. It uses a modified First-Fit Decreasing bin-packing heuristic [58–60]. If consolidation is feasible, a migration plan is generated that minimizes the number of migrations while preventing overloads. This allows idle nodes to be shut down, improving energy efficiency. One orchestrator can handle up to several racks of nodes; for larger deployments, a hierarchy of orchestrators that combine local and global load balancing is required.

Algorithm 1 Load balancing algorithm.

```
1:  function LOAD BALANCING
2:      hottest ← null
3:      coldest ← null
4:      for all node ∈ cluster/rack do
5:          if node is hotter than hottest then
6:              hottest ← node
7:          else if node is colder than coldest then
8:              coldest ← node
9:      if hottest == null then
10:         return null
11:     vm ← getLargestVM(hottest)
12:     technique ← getBestLMAlgo(hottest, coldest, vm)
13:     return (vm, technique, hottest, coldest)
```

5.1.2. Implementation

The framework consists of approximately 4000 lines of Python code. The cluster orchestrator communicates with worker node agents using gRPC [61] for efficient remote procedure calls. Agents run as daemon processes and interact with QEMU VMs via the QEMU Machine Protocol (QMP) [43].

Both the orchestrator and the agents are configurable via dedicated configuration files. We use a modified QEMU/KVM monitor that supports detailed VM profiling, including the analysis of memory access pattern. This monitor leverages Intel's NPT memory management hardware to track memory page modifications with minimal overhead.

5.2. Migration policy

This subsection outlines how the framework integrates migration-aware Service Level Objectives (SLOs) and supports customizable management policies for different operational contexts.

5.2.1. Service level objectives

The framework supports a set of migration-related SLOs that address both user-facing and operator-facing priorities. Each SLO can define either an upper or lower threshold: upper thresholds specify a maximum value beyond which an SLO violation occurs, while lower thresholds define a minimum acceptable value that must not be breached. The relative importance of SLOs can be specified by a weight for each SLO.

User SLOs focus on application availability and performance. The framework currently supports the following metrics, capturing the impact of migration on VM execution.

- Downtime (`DT`) is a upper-bound SLO to limit and quantify service interruption during migration, and
- Performance preservation (`PERF`), a lower-bound SLO measured in instructions per second (IPS) to ensure a certain level of performance during migration.

Operator SLOs address system-level efficiency and resource usage. All operator SLOs specify upper thresholds.

- Total migration time (`TT`),
- Transferred data volume (`TD`), and
- Additional resource utilization on the source host, such as CPU and memory usage (`CPU`, `MEM`).

These SLOs allow operators to define policies that focus on various aspects by specifying threshold values, assigning relative importance through weights, and selecting optimization targets. These policies can be adapted at runtime to respond to changing workload conditions or business priorities, without requiring changes to the system architecture.

5.2.2. Policy support

The presented framework supports flexible adaptation to diverse operational scenarios through **migration policies**, enabling data center operators to shape VM migration accordingly. A migration policy guides the migration selection process and includes a set of user-facing and operator-oriented weighted SLOs, along with an optimization target. The migration selector predicts the SLOs and migration metrics for all available techniques and selects the one expected to minimize the weighted SLO violations and perform best with respect to the optimization target.

To illustrate the concept, this work evaluates two policies: hotspot mitigation and consolidation. In **hotspot mitigation**, the goal is to alleviate overload on selected nodes while maintaining application responsiveness. Policies in this category typically assign high weights (0.6–0.8) to performance preservation, enforce strict downtime limits, and prioritize minimizing total migration time. In contrast, **consolidation** aims to reduce the number of active nodes by migrating workloads onto fewer machines. These policies emphasize operational concerns such as transferred data volume and additional resource usage and tend to assign moderate weights to performance metrics.

5.2.3. Migration algorithm selection

The selection of the migration algorithm is based on a multi-objective optimization function. Given are N live migration algorithms $T = \{t_1, t_2, \ldots, t_N\}$ and K metrics $M = \{m_1, m_2, \ldots, m_K\}$. The SLOs $S = \{s_1, s_2, \ldots, s_K\}$ are defined over the same space as the metrics (i.e., there can be one SLO per metric). Each SLO s_k has an associated threshold value θ_k that, if violated, constitutes an SLO violation. Additionally,

each SLO is assigned a weight w_k that reflect the SLO's significance – the larger the weight, the more desirable is the optimization of the SLO. For each available migration techniques $T = \{t_1, t_2, \ldots, t_n\}$, the machine learning models predict the six key metrics defined in Section 2.2.

For a metric m_k with SLO s_k and a prediction p_k, we define an *absolute* and *weighted* SLO violation score. The absolute SLO violation score V_{abs} assumes a value of 1 if the SLO is expected to be violated and 0 otherwise:

$$V_{abs}(p_k, \theta_k) = \begin{cases} 1 & \text{if upper-bound limit and } p_k > \theta_k \\ 1 & \text{if lower-bound limit and } p_k < \theta_k \\ 0 & \text{otherwise (no violation)} \end{cases} \quad (1)$$

The weighted SLO violation score captures the weighted relative extend of an SLO violation by multiplying the normalized SLO violation with its weight:

$$V_{rel}(p_k, \theta_k, w_k) = V_{abs}(p_k, \theta_k) \cdot w_k \cdot \frac{|p_k - \theta_k|}{\theta_k} \quad (2)$$

As an example, consider an upper-bound threshold of 10 and a predicted value of 15 for a given metric and migration technique. Then, $V_{abs}(15, 10) = 1$ because the predicted value of the metric exceeds the its SLO threshold. Assuming a weight of 0.2, the relative SLO violation score $V_{res}(15, 10, 0.2) = 0.1$, capturing the fact that the SLO metric is violated by a factor of 0.5 ($\frac{|15-10|}{10}$) multiplied by the weight of 0.2.

The total absolute and weighted SLO violation scores for a technique t_n are obtained by summing up the absolute/weighted SLO violation scores for each metric:

$$V^{tot}_{abs}(t_n) = \sum_{k=1}^{K} V_{abs}(p_k, \theta_k) \quad (3)$$

$$V^{tot}_{rel}(t_n) = \sum_{k=1}^{K} V_{rel}(p_k, \theta_k, w_k) \quad (4)$$

The chosen technique t^* is selected from the set of all techniques that attain the minimum of V^{tot}_{rel}. If this set contains multiple techniques, the one that maximizes the objective function $O(t_n)$ is selected. The objective function is defined analogously to V^{tot}_{rel} and quantifies the distance of a metric from its threshold. Note that the objective weights are defined independently of the SLO weights.

$$O_{rel}(p_k, \theta_k, w_k) = \begin{cases} w_k \cdot \left(1 - \frac{p_k}{\theta_k}\right) & \text{if upper-bound limit} \\ w_k \cdot \left(\frac{p_k}{\theta_k} - 1\right) & \text{if lower-bound limit} \end{cases} \quad (5)$$

$$O^{tot}_{rel}(t_n) = \sum_{k=1}^{K} O_{rel}(p_k, \theta_k, w_k) \quad (6)$$

As an example, consider one SLO with an lower-bound threshold of 10, a predicted value of 15, and a weight of 0.5 for a given migration technique and metric. Then, $O_{rel}(15, 10, 0.5) = 0.5 \cdot (\frac{15}{10} - 1) = 0.25$. For a second SLO, the upper-bound threshold is 50, the predicted value 40, and the weight 0.5, yielding $O_{rel}(40, 50, 0.5) = 0.5 \cdot (1 - \frac{40}{50}) = 0.1$ and a total objective score of $O^{tot}_{rel} = 0.25 + 0.1 = 0.35$.

The set of optimal techniques T^* is thus the set of techniques that minimize V^{tot}_{rel}:

$$T^* = \arg\min_{t_n \in T} V^{tot}_{rel}(t_n) \quad (7)$$

If exactly one technique satisfies all SLOs, it is chosen. Otherwise, we select the technique that maximizes the objective score:

$$t^* = \begin{cases} t_n \in T^* & |T^*| = 1 \\ \arg\max_{t_n \in T^*} O^{tot}_{rel}(t_n) & \text{otherwise} \end{cases} \quad (8)$$

5.2.4. Policy determination guidelines

To define effective migration policies, we recommend the following systematic approach:

1. **Identify business priorities**: Determine which aspects of migration have the greatest impact on service delivery and operational cost. Customer-facing services often prioritize performance preservation and minimal downtime.
2. **Translate requirements into SLO thresholds**: Map business requirements to measurable metrics. For example, tail-latency related Service Level Agreements (SLAs) can be translated into SLOs that guarantee a minimum level of performance (`PERF`).
3. **Assign weights based on criticality**: Allocate higher weights to metrics that are closely tied to customer SLAs or essential operations.
   - *Quantitative approach*: Perform a sensitivity analysis to measure application impact under varying degrees of SLO violation. Weights are assigned in proportion to the observed performance degradation.
   - *Risk-based weighting*: Assess the operational risk, such as financial loss or reduced customer satisfaction, associated with violating each SLO. Assign higher weights to higher-risk metrics.
   - *Historical analysis*: Review past migration events to identify metrics that correlated with service disruptions or performance regressions.
4. **Select an optimization target**: When multiple live migration techniques satisfy all SLOs, the chosen technique is selected based on optimization metric. This metric typically reflects the greatest operational benefit for the given context.
5. **Customize by context**: Adapt thresholds, weights, and optimization targets to suit specific workloads or changing conditions. Different application classes may require distinct policy profiles.
6. **Validate and refine**: Test derived policies in a non-production environment and refine them based on observed outcomes. Iterative tuning improves accuracy and resilience.

The weights used in the evaluation (Table 3) were derived using this methodology, with emphasis on the quantitative approach. The higher weights assigned to performance preservation in hotspot mitigation reflect the critical need of maintaining application responsiveness during periods of resource contention. In contrast, consolidation policies emphasize network efficiency to minimize the migration overhead during non-urgent workload redistribution.

6. Evaluation

This section presents a comprehensive evaluation of the framework across a range of data center scenarios. The evaluation begins with an assessment of individual live migration techniques, focusing on their performance and SLO violation scores in a controlled cluster environment. Then, the machine learning model is employed to select appropriate migration techniques under defined policy constraints. We compare the presented framework, labeled `GUIDED` against the nine static migration policies (`PRE`, `POST`, etc.) as well as an oracle (`ORACLE`) that always selects the best migration policy for the given scenario. The oracle technique represents an idealized baseline in which all migration outcomes are known in advance. Any deviation between `GUIDED` and `ORACLE` in terms of the SLO violation count and/or score is attributable to inaccuracies in performance prediction produced by the model.

6.1. Environment

The evaluation is conducted on an internal research cluster consisting of six heterogeneous physical nodes. Four nodes are equipped with octa-core Intel Skylake i7-6700 processors and 32GB of physical memory. The other two nodes use Intel(R) Xeon(R) Silver 4114 2.20GHz processors (10-core, 20-thread) with 64GB of memory. All nodes run Ubuntu 20.04 with Linux kernel version 5.3.7. The cluster is interconnected via a 1 Gbit/s local network. QEMU/KVM version 4.5 provides hypervisor functionality, with each VM running Ubuntu 18.04

Table 2
VM configurations for hotspot and consolidation scenarios.

Index	Hotspot			Consolidation		
	Workload	Cpus	Memory	Workload	Cpus	Memory
1	parsec.facesim	4	8 GB	NPB.bt.A	2	2 GB
2	parsec.vips	1	8 GB	tensorflow.CIFAR10	1	8 GB
3	parsec.vips	2	8 GB	NPB.dc.A	2	8 GB
4	parsec.vips	2	4 GB	parsec.vips	2	2 GB
5	parsec.vips	4	8 GB	parsec.raytrace	1	8 GB
6	parsec.blackscholes	4	2 GB	parsec.canneal	1	4 GB
7	parsec.x264	4	4 GB	tensorflow.CIFAR10	1	4 GB
8	tensorflow.CIFAR10	4	2 GB	tensorflow.CIFAR10	1	4 GB
9	parsec.vips	2	8 GB	parsec.bodytrack	1	8 GB
10	parsec.blackscholes	2	2 GB	parsec.x264	4	8 GB
11	parsec.vips	2	4 GB	NPB.cg.B	2	2 GB
12	NPB.dc.A	4	8 GB	parsec.raytrace	1	8 GB
13	parsec.bodytrack	4	2 GB	NPB.bt.A	2	8 GB
14	tensorflow.CIFAR10	4	8 GB	NPB.bt.C	1	4 GB
15	tensorflow.CIFAR10	4	2 GB	NPB.cg.B	4	2 GB
16	parsec.blackscholes	4	8 GB	NPB.bt.A	4	4 GB
17	parsec.vips	2	8 GB	parsec.bodytrack	2	2 GB
18	parsec.vips	4	8 GB	NPB.dc.A	4	2 GB
19	tensorflow.CIFAR10	4	8 GB	parsec.blackscholes	2	2 GB
20	parsec.bodytrack	4	8 GB	tensorflow.CIFAR10	4	4 GB
21	NPB.cg.B	4	2 GB	NPB.ft.B	1	8 GB
22	tensorflow.CIFAR10	4	8 GB	tensorflow.CIFAR10	4	4 GB
23	parsec.bodytrack	4	2 GB	NPB.ft.B	1	4 GB
24	NPB.cg.B	4	8 GB	NPB.cg.B	4	4 GB
25	parsec.vips	2	8 GB	NPB.cg.B	1	8 GB

as its guest operating system. Disk images are accessed via network-attached storage, with VM disk I/O traffic routed through the VM network, thereby potentially introducing interference with migration operations.

The VMs are configured with 1–4 vCPUs and 2–8 GB of memory. The migration metrics `TT`, `DT` and `TD` are measured directly at the hypervisor, while `PERF`, `CPU`, and `MEM` are collected using Linux system tools such as **perf** and **pidstats**. All results show the average of eight valid runs with the same predefined random seeds to create random, yet reproducible initial configurations.

6.2. Workloads

We select a diverse set of applications that reflect various workload characteristics commonly observed in modern data centers. These workloads vary in resource usage, memory access behaviors, and computational intensity, capturing a broad spectrum of real-world application scenarios.

The workloads were carefully selected to ensure comprehensive coverage of both overload and consolidation scenarios. For overload scenarios, we include applications with high CPU and memory demands, simulating conditions with significant resource contention. For consolidation, we use workloads with low to moderate resource usage, representing typical situations where underutilized nodes can be consolidated. This design ensures that the evaluation addresses a wide range of operational challenges, from severe contention to energy-efficient optimization opportunities.

Our evaluation includes the following three categories:

1. **Computation-intensive parallel workloads**: Eight applications from the PARSEC benchmark suite [62] (Blackscholes, Bodytrack, Canneal, Dedup, Facesim, Raytrace, Vips, and X264). These applications represent diverse domains such as financial modeling (Blackscholes), computer vision (Bodytrack), engineering simulations (Canneal, Facesim), media processing (Vips, X264), and data compression (Dedup). They demonstrate varying levels of parallelism, memory access patterns, and CPU usage.
2. **Scientific computing workloads**: Six benchmarks from the NAS Parallel Benchmarks (NPB) [63]: bt.A, bt.C, cg.B, dc.A, ft.B, and is.C. NPB is a widely used benchmark suite designed by NASA to represent the computation and data movement patterns of large-scale computational fluid dynamics (CFD) and scientific applications. Each NPB kernel models a different class of scientific computation, and the selected input sizes (problem classes A, B, and C) reflect a range of computational intensities and memory requirements, following the standard NPB specification[1]. This selection ensures that our evaluation covers both moderate and large-scale scientific workloads, and the diversity of NPB kernels allows us to assess the migration framework under a variety of memory access and communication patterns.
3. **Machine learning workloads**: CIFAR-10 inference using TensorFlow (`Tensorflow.CIFAR10`), representing modern AI workloads. This application features distinct memory access patterns due to large matrix operations and exhibits unique resource demands during inference.

The complete list of workloads and their VM configurations is shown in Table 2. All applications are executed using all available vCPUs of the VM. To ensure reproducibility, workloads are selected using a fixed random seed for each experiment.

This workload diversity enables evaluation across applications with:

- Varying memory modification rates
- Different degrees of spatial and temporal memory access locality
- A range of CPU utilization profiles (compute-bound vs. memory-bound)
- Differing sensitivities to performance degradation during migration

6.3. Maintenance migration

The first scenario evaluates the performance of static migration techniques with `GUIDED` in a migration scenario. The goal is to migrate away all VMs of a node to allow maintenance due to, for example, a failing hardware component. We place several VMs executing random workloads on a source node and serially migrate all VMs to other nodes until the source node is empty. Fig. 6 shows the average from initiation to

[1] https://www.nas.nasa.gov/software/npb_problem_sizes.html

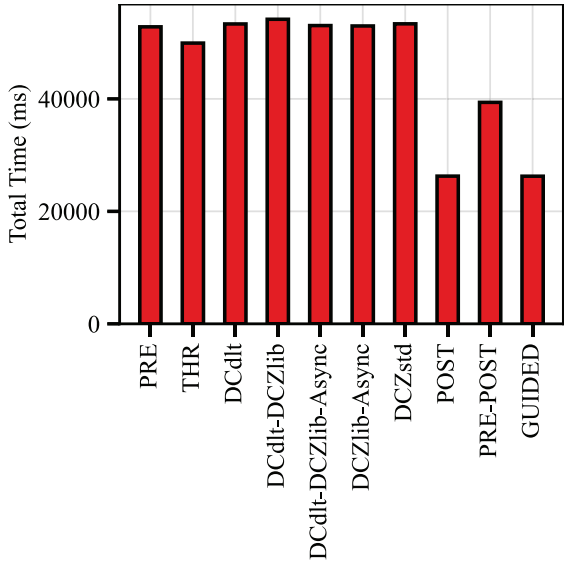

Fig. 6. Migration results for an urgent case.

completion of the migrations. We begin by evaluating the framework in a time-critical scenario designed to validate the accuracy of the live migration performance prediction model. We observe that `POST` completes migrations approximately in half the time as the standard `PRE` technique and outperforms all other static techniques. With the total migration time (`TT`) being the only concern, our framework `GUIDED` consistently selects the `POST` technique for all migrations events, confirming that the model is able to identify the optimal technique under a simple, single-objective condition.

6.4. Policies for hotspot and consolidation scenarios

We now evaluate the framework under a range of hotspot and consolidation scenarios using complex, multi-metric policies. Table 3 lists the six policies used in our experiments. These policies were designed to reflect realistic operational priorities in modern data centers. Each policy defines a set of SLO constraints (with associated weights) and an optimization goal. The following paragraphs describe each policy in more detail.

We first consider the three policies designed for **hotspot mitigation scenarios**:

- **Ensure performance**: Designed for business-critical applications with high performance preservation weight and moderate downtime constraint. The optimization goal `TT` minimizes total migration time when there are several viable migration techniques. This policy is particularly relevant for high-value, customer-facing services where performance preservation is essential.
- **Traffic overload**: Created for environments with network congestion, this policy minimizes the transferred data volume while balancing performance and downtime. This policy is a good fit for situations with network bandwidth constraints or when the source nodes are experiencing network congestion.
- **High performance**: This comprehensive policy targets hotspot mitigation while considering both application and infrastructure constraints. It balances application metrics with operational resource limits. The policy aims to minimize the total migration time while considering the broadest range of constraints.

Three policies represent **consolidation scenarios**:

- **Save traffic**: Emphasizes network efficiency by minimizing the transferred data volume while maintaining moderate performance requirements. The lower performance threshold acknowledges that consolidation operations are often scheduled during off-peak hours when a moderate performance impact is acceptable.
- **Reduced downtime**: Targeting services where availability is critical even during consolidation, this policy minimizes downtime as its primary objective. It enforces a moderate downtime threshold, maintains reasonable performance, and limits total migration time.
- **Lowest operation cost**: This resource-efficient policy minimizes total migration time while maintaining acceptable performance, moderate downtime, and restricting memory overhead. It is designed for large-scale consolidation operations where minimizing total migration duration is valuable for predictable operations planning.

The policies intentionally set different `PERF` thresholds for hotspot (75-80 %) and consolidation (50-70 %) scenarios to reflect their distinct operational priorities. VMs on overloaded nodes are already experiencing performance interference, making performance preservation during hotspot mitigation especially critical. In contrast, VMs selected for consolidation typically run on underutilized nodes and can tolerate moderate performance degradation in exchange for infrastructure efficiency gains.

6.5. Hotspot scenario

Load balancing is a critical operational strategy in data centers for optimizing resource utilization and ensuring workload performance. A node is classified as a *hotspot* when the resource demand exceeds a defined threshold. Hotspots frequently occur when VM workloads exhibit high dynamicity and are densely consolidated on a physical server. Effective hotspot management is essential for VM orchestration systems to maintain high resource utilization while guaranteeing application performance.

To evaluate the hotspot mitigation of the presented framework, we generate hotspots by strategically placing VMs on selected physical nodes. Each node is assigned three to four VMs executing randomly selected workloads. Following Sandpiper's approach [6], a node is classified as a hotspot if CPU utilization exceeds 75 % in three out of five measurements during a five-minute window with measurements every 60 seconds.

To compare the intelligent migration technique selection (`GUIDED`) against static migration strategies, we first perform migrations using

Table 3
Policies defining an objective and SLOs.

Scenario	Policy Name	User SLOs	Operator SLOs	Optimize
Hotspot	Ensure Performance	PERF: 75 %, Weight: 0.7 DT: 1000ms, Weight:0.3		TT
	Traffic Overload	DT: 2000ms, Weight:0.3 PERF: 80 %, Weight: 0.7		TD
	High Performance	DT: 2000ms, Weight: 0.2 PERF: 80 %, Weight: 0.6	CPU: 100 %, Weight: 0.1 MEM: 512MB, Weight: 0.1	TT
Consolidation	Save Traffic	PERF: 50 %, Weight: 1		TD
	Reduced Downtime	DT: 2000ms, Weight: 0.5 PERF: 70 %, Weight: 0.3	TT: 60000ms, Weight: 0.2	DT
	Lowest Operation Cost	DT: 2000ms, Weight: 0.2 PERF: 70 %, Weight: 0.7	MEM: 512MB, Weight: 0.1	TT

GUIDED, then replay the exact same migration sequence using each static migration technique. The ORACLE result is obtained by selecting the best-performing migration technique for each event, based on the recorded migration metrics.

The results are shown in Figs. 7 and 8. Fig. 7 presents the absolute SLO violation scores (V_{abs}^{tot}) in the top row and the weighted SLO violation score (V_{rel}^{tot}) in the bottom row for all nine static technique selections, the presented framework (GUIDED), and the oracle across the three hotspot policies defined in Table 3. Fig. 8 plots the absolute value of the optimization metric for each policy. Across all three hotspot scenarios, the presented framework achieves results that only slightly behind ORACLE, validating the effectiveness of dynamic migration technique selection based on machine-learned performance prediction models. In two of the three scenarios, GUIDED produces the lowest number of SLO violations among all static technique selections. In the *Traffic Overload* scenario, DCdlt and DCZstd show slightly better SLO violation counts and/or scores; however, when also including the optimization metric (TD - transferred data volume), GUIDED outperforms both static techniques. We also observe that the GUIDED only performs marginally worse than ORACLE. This further demonstrates the high accuracy of the migration performance prediction models. As a reminder, any deviation from ORACLE is due to prediction inaccuracies.

6.6. Consolidation scenario

Consolidation aims to concentrate VMs on fewer nodes to allow idle nodes to be powered down, thereby conserving energy. The primary distinction from load balancing lies in availability of resources:

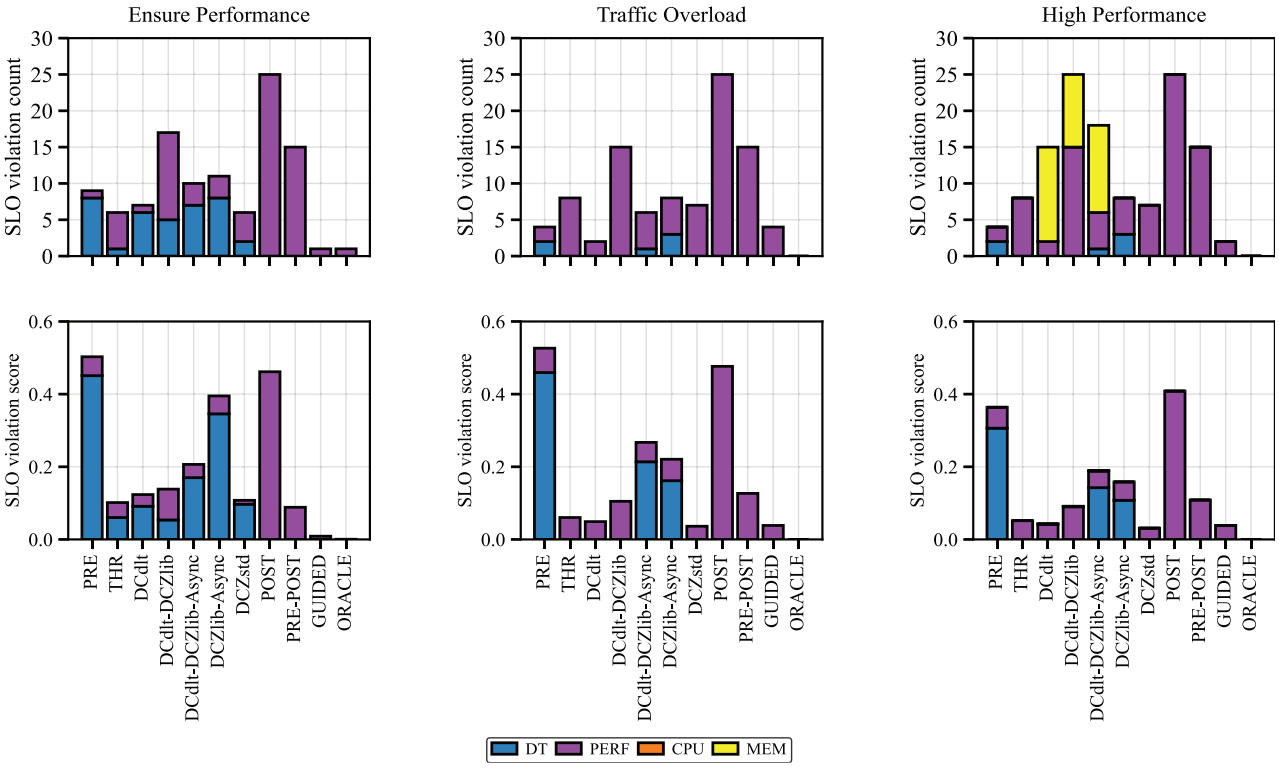

Fig. 7. SLO violation counts and scores of hotspot mitigation scenarios.

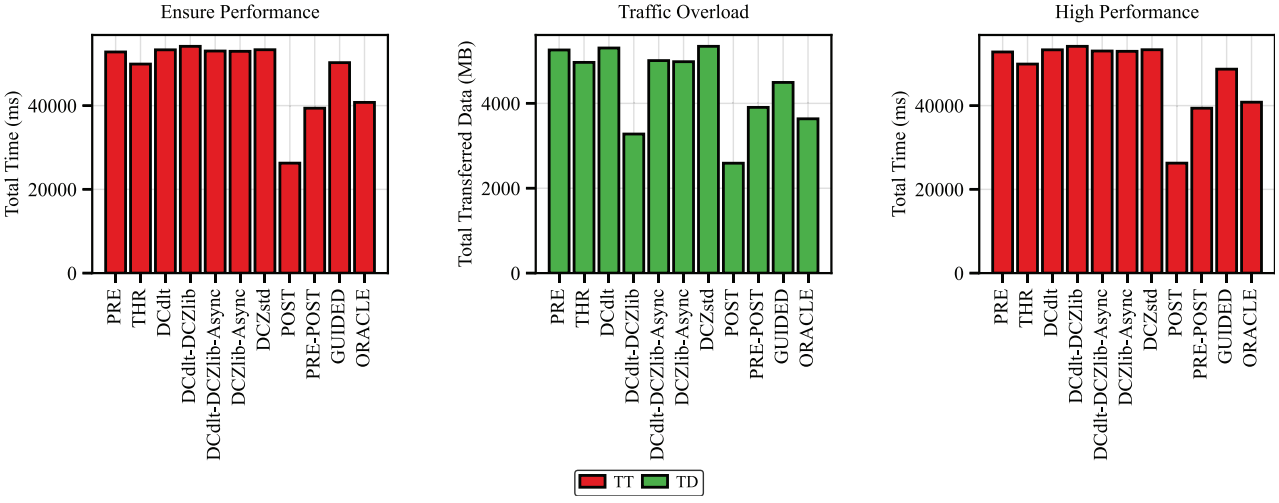

Fig. 8. Optimization goal of hotspot mitigation scenarios.

in consolidation scenarios, source nodes are typically underutilized, which enables the use of more resource-intensive migration techniques such as DCZlib-Async, DCdlt, DCdltDCZlib, DCdlt-DCZlib-Async, and DCZstd.

To evaluate this scenario, the cluster is initialized by randomly placing 20 VMs with diverse workloads (Table 2). We then perform 25 migrations using the consolidation policies defined in Table 3.

Figs. 9 and 10 show the results across the three consolidation scenarios. In the first scenario, which focuses on minimizing the transferred data volume as its optimization target, most techniques, except DCdlt-DCZlib and POST, achieve zero SLO violations. The GUIDED framework outperforms all static techniques in optimizing the objective, even surpassing post-copy based techniques and approaching ORACLE performance.

In the second scenario, which includes three SLO constraints and prioritizes downtime minimization, the GUIDED selection records four violations – only to be outperformed by THR with three violation – and achieves the lowest weighted SLO violation score apart from the theoretical optimum ORACLE. While GUIDED does not match the downtime of POST (91ms) or ORACLE (10ms), it maintains a competitive average downtime of 160ms with fewer SLO violations.

In the final consolidation scenario, GUIDED again demonstrates the best performance in terms of SLO violation count and violation score across all static techniques. It achieves total migration times comparable to PRE-POST, which closely follows POST in optimization performance.

These results confirm that model-based, dynamic migration technique selection effectively reduces the Total Cost of Ownership (TCO) during VM consolidation by using fewer resources and completing

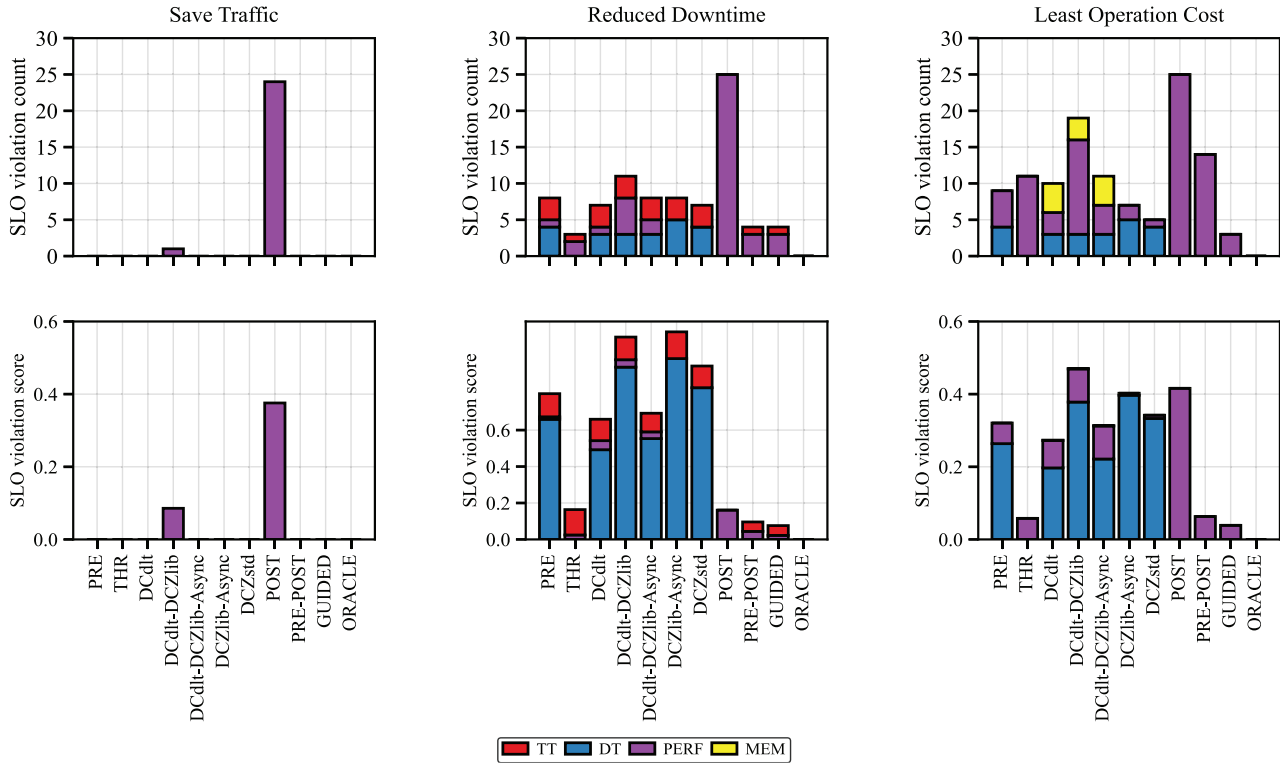

Fig. 9. SLO violation counts and scores of consolidation scenarios.

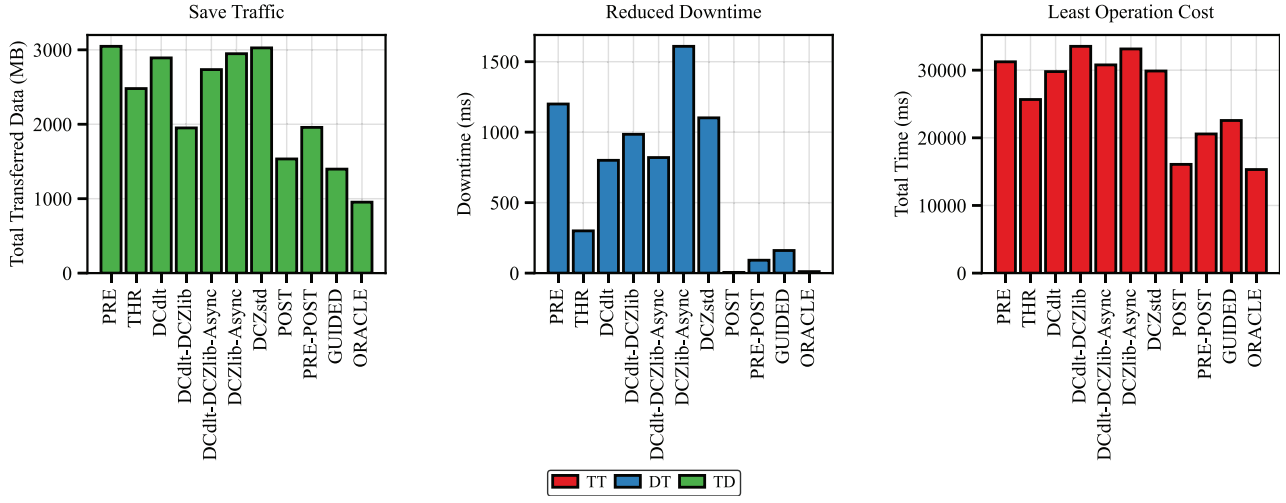

Fig. 10. Optimization goal of consolidation scenarios.

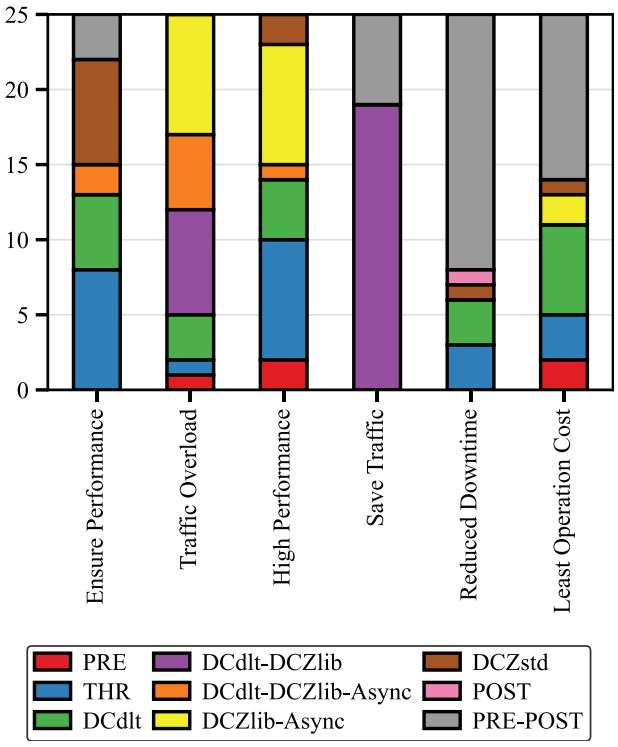

Fig. 11. Selected migration techniques.

migrations more efficiently than any single-technique approach. Considering the combined metrics violation count, violation score, and optimization performance, `GUIDED` closely approximates the theoretical optimum represented by `ORACLE`.

6.7. Analysis of technique selection

Fig. 11 shows the migration techniques selected by `GUIDED` across the six policies: three hotspot scenarios (left bars) and three consolidation scenarios (right bars).

In hotspot scenarios, where resources are scarce and constraints are tight, we observe a wide range of selected techniques. Selecting an appropriate migration strategy in this context is challenging, as it must account for VM workload characteristics, the local resource situation, and strict SLO requirements. The model handles this complexity effectively. For example, under the *Ensure Performance* policy (left-most bar), throttling and compression-based techniques are frequently selected, as they offer a favorable trade-off between performance preservation and total migration time.

In contrast, consolidation scenarios show less variation in technique selection. Since the source nodes are underutilized and the SLO thresholds are more relaxed, most techniques avoid violations. As a result, the model focuses primarily on optimizing the target metric. Under the *Save Traffic* policy, `DCdlt-DCZlib` is selected for 19 out of 25 migrations. This reflects the model's prediction that delta compression combined with Zlib achieves the greatest data reduction for the observed memory access patterns, while staying within acceptable performance bounds.

The `PRE-POST` technique is frequently chosen across all three consolidation scenarios. It offers fast migration completion with a moderate impact on VM performance during the recovery phase. Given the lower `PERF` thresholds (50-70 %) in these policies, the model correctly identifies `PRE-POST` as a viable choice that completes migrations quickly without violating constraints.

Interestingly, in hotspot scenarios, techniques such as `DCZlib` and `DCdlt` are selected more frequently than expected. One might anticipate that faster techniques like `POST` or `PRE-POST` would dominate. However, the model accurately predicts that post-copy methods would cause significant performance degradation for the given workloads, which is unacceptable under stricter performance SLOs. Consequently, the model favors compression-based approaches that better preserve performance while maintaining reasonable migration durations.

The results demonstrate the model's ability to navigate the complex relationship between workload characteristics, infrastructure conditions, and technique behavior. Rather than applying static heuristics, the model dynamically adapts its recommendations to the specific constraints and objectives of each migration scenario to strike a balance between SLO compliance and operational efficiency.

6.8. Prediction accuracy and SLO violation analysis

The prediction model shows varying accuracy across the six migration metrics. As seen in Table 4, downtime (`DT`) exhibits the highest relative error (GMRE = 0.39). This is primarily due to the substantial variance between techniques: post-copy methods yield consistently low downtimes (around 10ms) by resuming the VM early, whereas pre-copy techniques often incur much longer and less predictable downtimes, particularly under memory-intensive workloads. During the stop-and-copy phase, the remaining dirty pages must be transferred while the VM is paused, resulting in downtimes that can differ by orders of magnitude depending on instantaneous memory activity.

Downtime prediction is further complicated by its sensitivity to transient factors such as network congestion and host CPU scheduling during the final migration phase. Despite these challenges, the model based

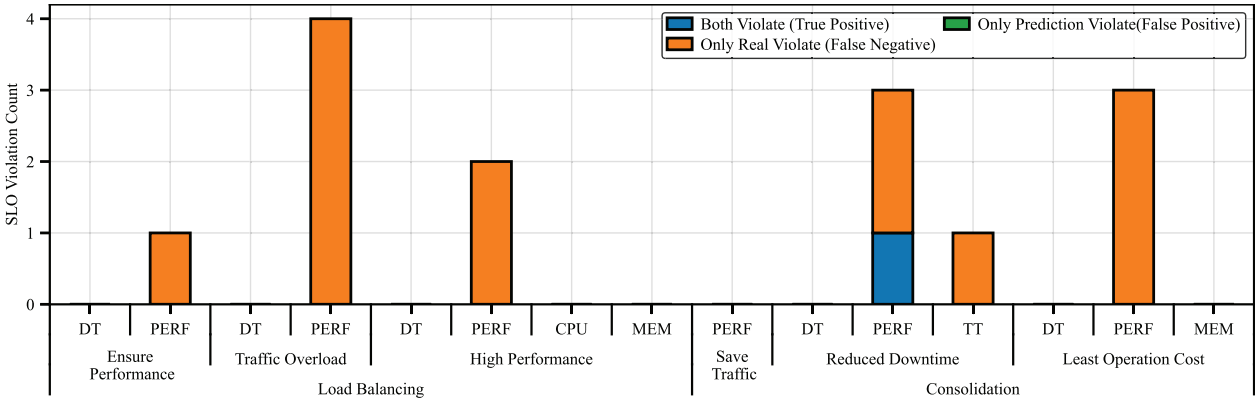

Fig. 12. Classification of SLO violations: blue bars indicate correctly predicted violations (true positives), orange bars indicate missed violations (false negatives).

Table 4
Model prediction Generalized Mean Absolute Error (GMAE) and Generalized Mean Relative Error (GMRE) across all evaluated metrics.

Metric	GMAE	GMRE
Total migration time (TT)	4161.02	0.14
Downtime (DT)	84.76	0.39
Transferred data volume (TD)	338.68	0.13
Performance degradation (PERF)	0.08	0.09
Add. host CPU utilization (CPU)	2.12	0.08
Add. memory utilization (MEM)	57.47	0.52

on ExtraTrees regression (see Section 5) accurately captures the *relative* performance differences between techniques. Relative accuracy is sufficient for guiding technique selection, as the framework requires only the correct ranking of candidate techniques to avoid SLO violations in most cases.

Fig. 12 quantifies the effect of prediction errors on actual SLO compliance. Blue bars represent correctly predicted violations (true positives), while orange bars indicate unexpected violations (false negatives). While some violations remain undetected due to prediction inaccuracies, the overall classification accuracy remains high, confirming the model's practical effectiveness.

On average, the model achieves a relative prediction error of approximately 22.5 % across all metrics. Notably, the relative error is higher for downtime (39 %) and memory overhead (52 %), primarily due to the high variability and sensitivity of these metrics to transient system conditions. While the model achieves low prediction errors for most metrics, these results highlight the importance of continued refinement, especially for metrics such as downtime and memory overhead, where prediction errors are most pronounced. Future work may incorporate uncertainty estimation or online adaptation mechanisms to further improve reliability under highly variable conditions.

7. Conclusion

This paper presents a machine-learning-based framework for intelligent VM live migration in data centers. Unlike traditional approaches that rely on static technique selection, our framework dynamically chooses the most suitable migration strategy for each event based on workload characteristics, system conditions, and policy-driven Service Level Objectives (SLOs). The core of the framework is a set of performance prediction models that estimate key migration metrics for all supported techniques, enabling multi-objective optimization under real-world constraints.

The framework is efficient, lightweight, and suitable for rack-scale environments. It integrates seamlessly with QEMU/KVM via gRPC and QMP, and supports a broad range of migration techniques, including pre-copy, post-copy, and compression-based optimizations.

We evaluated the framework across diverse hotspot mitigation and consolidation scenarios using workloads from PARSEC, NPB, and TensorFlow. The results show that the presented approach consistently outperforms static technique selection, often matching or closely approximating an oracle that has perfect foresight. The framework reduces SLO violations, improves migration efficiency, and adapts to both performance-critical and cost-driven operational goals.

The prediction model achieves a relative error of approximately 15 % across all metrics. While absolute accuracy varies – particularly for downtime due to its sensitivity to transient conditions – the model reliably predicts relative performance differences between techniques, which is sufficient for effective decision-making in most scenarios. Errors that do occur tend to result in minor deviations from the oracle baseline.

A limitation of the current approach lies in generalization: while the model performs well on a diverse dataset of synthetic and real workloads, its accuracy on entirely novel workload types remains unverified. Future work will explore online learning and transfer learning to extend adaptability, as well as the incorporation of confidence estimation to guide fallback decisions under uncertainty.

Overall, this work demonstrates that intelligent, workload-aware migration technique selection is both feasible and beneficial. By combining predictive modeling with flexible policy definition, data centers can improve application performance, reduce operational overhead, and better meet business objectives under dynamic workloads.

CRediT authorship contribution statement

Youngsu Cho: Writing – original draft, Visualization, Software, Methodology, Data curation, Conceptualization; **Changyeon Jo:** Writing – original draft, Methodology, Formal analysis, Data curation, Conceptualization; **Reza Entezari-Maleki:** Writing – review & editing; **Jörn Altmann:** Writing – review & editing; **Bernhard Egger:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Jörn Altmann und Bernhard Egger report financial support was provided by the National Research Foundation of Korea, funded by the Korea government, Ministry of Science and ICT (MSIT), with project No. NRF-RS-2023-00302083 (as part of the EC-funded EU Horizon Swarmchestrate Project), project No. 21A20151113068 (BK21 Plus for Pioneers in Innovative Computing - Dept. of Computer Science & Engineering, SNU), the Institute of Engineering Research, and Seoul National University. ICT at Seoul National University provided research facilities for this study. The other authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Amazon EC2, 2024. https://aws.amazon.com/ec2/.
[2] Azure virtual machines, 2024. https://azure.microsoft.com/en-us/services/virtual-machines/.
[3] Google cloud compute engine, 2024. https://cloud.google.com/compute.
[4] IBM cloud virtual server for VPC, 2024. https://www.ibm.com/products/virtual-servers.
[5] Alibaba cloud 2024. https://www.alibabacloud.com/.
[6] T. Wood, P. Shenoy, A. Venkataramani, M. Yousif, Black-box and gray-box strategies for virtual machine migration, in: USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2007. http://faculty.cs.gwu.edu/timwood/papers/NSDI07-sandpiper.pdf.
[7] Z. Shen, S. Subbiah, X. Gu, J. Wilkes, Cloudscale: elastic resource scaling for multi-tenant cloud systems, 5 of the 2nd ACM Symposium on Cloud Computing, SoCC '11 New York, NY, USA, ACM, 2011. https://doi.org/10.1145/2038916.2038921
[8] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, J. Lawall, Entropy: a consolidation manager for clusters, in: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '09, New York, NY, USA, ACM, 2009, pp. 41–50. https://doi.org/10.1145/1508293.1508300
[9] D. Novaković, N. Vasić, S. Novaković, D. Kostić, R. Bianchini, Deepdive: transparently identifying and managing performance interference in virtualized environments, in: Proceedings of the 2013 USENIX Conference on Annual Technical Conference, USENIX ATC'13, USENIX Association, Berkeley, CA, USA, 2013, pp. 219–230. http://dl.acm.org/citation.cfm?id=2535461.2535489.
[10] A. Ruprecht, D. Jones, D. Shiraev, G. Harmon, M. Spivak, M. Krebs, M. Baker-Harvey, T. Sanderson, Vm live migration at scale, in: Proceedings of the 14th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '18, New York, NY, USA, ACM, 2018, pp. 45–56. https://doi.org/10.1145/3186411.3186415
[11] Google cloud documentation, 2024. https://cloud.google.com/compute/docs/instances/live-migration-process.
[12] Google cloud platform blog, 2024. https://goo.gl/Ui3HFd.

[13] R. Eswaran, M. Yan, K. Gopalan, Tackling memory footprint expansion during live migration of virtual machines, in: 2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2024, pp. 158–167. https://doi.org/10.1109/CCGrid59990.2024.00027

[14] W. Voorsluys, J. Broberg, S. Venugopal, R. Buyya, Cost of virtual machine live migration in clouds: a performance evaluation, in: M.G. Jaatun, G. Zhao, C. Rong (Eds.), Cloud Computing, Berlin Heidelberg, Springer, 2009, pp. 254–265.

[15] R.M. Haris, K.M. Khan, A. Nhlabatsi, Live migration of virtual machine memory content in networked systems, Comput. Netw. 209 (2022). https://doi.org/10.1016/j.comnet.2022.108898

[16] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID '10, Washington, DC, USA, IEEE Computer Society, 2010, pp. 826–831. https://doi.org/10.1109/CCGRID.2010.46

[17] W. Fang, X. Liang, S. Li, L. Chiaraviglio, N. Xiong, Vmplanner: optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers, Comput. Netw. 57 (1) (2013) 179–196. https://doi.org/10.1016/j.comnet.2012.09.008

[18] J.W. Jiang, T. Lan, S. Ha, M. Chen, M. Chiang, Joint vm placement and routing for data center traffic engineering, in: 2012 Proceedings IEEE INFOCOM, 2012, pp. 2876–2880. https://doi.org/10.1109/INFCOM.2012.6195719

[19] L. Liu, H. Wang, X. Liu, X. Jin, W.B. He, Q.B. Wang, Y. Chen, Greencloud: a new architecture for green data center, in: Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session, ICAC-INDST '09, New York, NY, USA, ACM, 2009, pp. 29–38. https://doi.org/10.1145/1555312.1555319

[20] J. Xu, J.A.B. Fortes, Multi-objective Virtual Machine Placement in Virtualized Data Center Environments, 2010, pp. 179–188. https://doi.org/10.1109/GreenCom-CPSCom.2010.137

[21] L.A. Barroso, U. Hölzle, P. Ranganathan, The datacenter as a computer: designing warehouse-scale machines, Synth. Lect. Comput. Arch. 13 (3) (2018) 189.

[22] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation 2 (2005) 273–286. NSDI'05, USENIX Association, http://dl.acm.org/citation.cfm?id=1251203.1251223.

[23] H. Jin, L. Deng, S. Wu, X. Shi, X. Pan, Live virtual machine migration with adaptive, memory compression, in: 2009 IEEE International Conference on Cluster Computing and Workshops, 2009, pp. 1–10. https://doi.org/10.1109/CLUSTR.2009.5289170

[24] C. Jo, E. Gustafsson, J. Son, B. Egger, Efficient live migration of virtual machines using shared storage, in: Proceedings of the 9th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, VEE '13, New York, NY, USA, ACM, 2013, pp. 41–50. https://doi.org/10.1145/2451512.2451524

[25] C. Jo, B. Egger, Optimizing live migration for virtual desktop clouds, IEEE 5th International Conference on Cloud Computing Technology and Science 1 (2013) 104–111. https://doi.org/10.1109/CloudCom.2013.21

[26] P. Svärd, B. Hudzia, J. Tordsson, E. Elmroth, Evaluation of delta compression techniques for efficient live migration of large virtual machines, in: Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '11, New York, NY, USA, ACM, 2011, pp. 111–120. https://doi.org/10.1145/1952682.1952698

[27] H. Liu, H. Jin, X. Liao, L. Hu, C. Yu, Live migration of virtual machine based on full system trace and replay, in: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, HPDC '09, New York, NY, USA, ACM, 2009, pp. 101–110. https://doi.org/10.1145/1551609.1551630

[28] T. Hirofuchi, H. Nakada, S. Itoh, S. Sekiguchi, Reactive consolidation of virtual machines enabled by postcopy live migration, in: Proceedings of the 5th International Workshop on Virtualization Technologies in Distributed Computing, VTDC '11, New York, NY, USA, ACM, 2011, pp. 11–18. https://doi.org/10.1145/1996121.1996125

[29] P. Svard, J. Tordsson, B. Hudzia, E. Elmroth, High performance live migration through dynamic page transfer reordering and compression, in: Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, CLOUDCOM '11, Washington, DC, USA, IEEE Computer Society, 2011, pp. 542–548. https://doi.org/10.1109/CloudCom.2011.82

[30] Z. Liu, W. Qu, W. Liu, K. Li, Xen live migration with slowdown scheduling algorithm, in: Proceedings of the 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT '10, Washington, DC, USA, IEEE Computer Society, 2010, pp. 215–221. https://doi.org/10.1109/PDCAT.2010.88

[31] A. Gupta, S. Namasudra, A novel technique for accelerating live migration in cloud computing, Autom. Softw. Eng. 29 (2022). https://doi.org/10.1007/s10515-022-00332-2

[32] J. Li, J. Zhao, Y. Li, L. Cui, B. Li, L. Liu, J. Panneerselvam, imig: toward an adaptive live migration method for kvm virtual machines, Comput. J. 58 (6) (2014) 1227. https://doi.org/10.1093/comjnl/bxu065

[33] S. Nathan, U. Bellur, P. Kulkarni, On selecting the right optimizations for virtual machine migration, in: Proceedings of the12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '16, New York, NY, USA, ACM, 2016, pp. 37–49. https://doi.org/10.1145/2892242.2892247

[34] P. Svärd, B. Hudzia, S. Walsh, J. Tordsson, E. Elmroth, Principles and performance characteristics of algorithms for live vm migration, SIGOPS Oper. Syst. Rev. 49 (1) (2015) 142–155. https://doi.org/10.1145/2723872.2723894

[35] Y. Cho, C. Jo, H. Kim, B. Egger, Towards economical live migration in data centers, in: Economics of Grids, Clouds, Systems, and Services, GECON '20, Cham, Springer International Publishing, 2020, pp. 173–188. https://doi.org/10.1007/978-3-030-63058-4_15

[36] M.E. Elsaid, H.M. Abbas, C. Meinel, Virtual machines pre-copy live migration cost modeling and prediction: a survey, Distributed and Parallel Databases, 2022. https://doi.org/10.1007/s10619-021-07387-2

[37] S. Soma, S. Rukmini, Virtual machine and container live migration algorithms for energy optimization of data centre in cloud environment: a research review, in: IoT Based Control Networks and Intelligent Systems, Springer Nature Singapore, 2023, pp. 637–647.

[38] C. Jo, Y. Cho, B. Egger, A machine learning approach to live migration modeling, in: ACM Symposium on Cloud Computing, SoCC'17, 2017. https://doi.org/10.1145/3127479.3129262

[39] B. Egger, E. Gustafsson, C. Jo, J. Son, Efficiently restoring virtual machines, Int. J. Parallel Program. 43 (3) (2015) 421–439. https://doi.org/10.1007/s10766-013-0295-0

[40] M. Fenn, M.A. Murphy, J. Martin, S. Goasguen, An evaluation of kvm for use in cloud computing, in: Proc. 2nd International Conference on the Virtual Computing Initiative, NC, USA, 2008.

[41] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP '03, New York, NY, USA, Association for Computing Machinery, 2003, pp. 164–177. https://doi.org/10.1145/945445.945462

[42] M.R. Hines, K. Gopalan, Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, in: ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '09, New York, NY, USA, ACM, 2009, pp. 51–60. Proceedings of the 2009. https://doi.org/10.1145/1508293.1508301

[43] F. Bellard, Qemu, a fast and portable dynamic translator, USENIX Ann. Tech. Conf. 41 (2005) 46.

[44] Meta, Smaller and faster data comparession with Zstandard, 2016. https://engineering.fb.com/2016/08/31/core-data/smaller-and-faster-data-compression-with-zstandard/.

[45] A. Koto, K. Kono, H. Yamada, A guideline for selecting live migration policies and implementations in clouds, in: Proceedings of the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, CLOUDCOM '14, Washington, DC, USA, IEEE Computer Society, 2014, pp. 226–233. https://doi.org/10.1109/CloudCom.2014.36

[46] S. Nathan, U. Bellur, P. Kulkarni, Towards a comprehensive performance model of virtual machine live migration, in: Proceedings of the Sixth ACM Symposium on Cloud Computing, SoCC '15, New York, NY, USA, ACM, 2015, pp. 288–301. https://doi.org/10.1145/2806777.2806838

[47] S. Akoush, R. Sohan, A. Rice, A.W. Moore, A. Hopper, Predicting the performance of virtual machine migration, in: Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS '10, Washington, DC, USA, IEEE Computer Society, 2010, pp. 37–46. https://doi.org/10.1109/MASCOTS.2010.13

[48] T. He, A.N. Toosi, R. Buyya, Camig: Concurrency-Aware Live Migration Management of Multiple Virtual Machines in SDN-Enabled Clouds, Technical Report, 2021. https://arxiv.org/abs/2111.08942.

[49] A. Belgacem, S. Mahmoudi, M.A. Ferrag, A machine learning model for improving virtual machine migration in cloud computing, J. Supercomput. 79 (9) (2023) 9486–9508.

[50] R.M. Haris, M. Barhamgi, A. Nhlabatsi, K.M. Khan, Optimizing pre-copy live virtual machine migration in cloud computing using machine learning-based prediction model, Computing 106 (2024) 3031–3062. https://doi.org/10.1007/s00607-024-01318-6

[51] Y. Gong, J. Huang, B. Liu, J. Xu, B. Wu, Y. Zhang, Dynamic resource allocation for virtual machine migration optimization using machine learning, 2024. https://arxiv.org/abs/2403.13619.

[52] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and dynamicity of clouds at scale: google trace analysis, in: Proceedings of the Third ACM Symposium on Cloud Computing, SoCC '12, ACM 7 (2012) 13. https://doi.org/10.1145/2391229.2391236

[53] M. Tirmazi, A. Barker, N. Deng, M.E. Haque, Z.G. Qin, S. Hand, M. Harchol-Balter, J. Wilkes, Borg: the next generation, in: Proceedings of the Fifteenth European Conference on Computer Systems, EuroSys '20, New York, NY, USA, Association for Computing Machinery, 2020. https://doi.org/10.1145/3342195.3387517

[54] D. Park, H. Kim, Y. Cho, C. Jo, B. Egger, Can VM live migration improve job throughput? Evidence from a real world cluster trace, in: Economics of Grids, Clouds, Systems, and Services, GECON '21, Cham, Springer International Publishing, 2021, pp. 17–26. https://doi.org/10.1007/978-3-030-92916-9_2

[55] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42. https://doi.org/10.1007/s10994-006-6226-1

[56] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (3) (2004) 199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

[57] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT press, 2002. https://doi.org/10.7551/mitpress/4175.001.0001

[58] E.G. Coffman, Jr, M.R. Garey, D.S. Johnson, Approximation Algorithms for Bin Packing: A Survey, Approximation algorithms for NP-Hard Problems, 1997. 46–93

[59] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Fut. Gen. Comput. Syst. 28 (5) (2012) 755–768. https://doi.org/10.1016/j.future.2011.04.017
[60] T. Wood, G. Tarasuk-Levin, P. Shenoy, P. Desnoyers, E. Cecchet, M.D. Corner, Memory buddies: exploiting page sharing for smart colocation in virtualized data centers, in: ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '09, ACM, 2009, pp. 31–40. Proceedings of the 2009. https://doi.org/10.1145/1508293.1508300
[61] gRPC, 2024. https://grpc.io/.
[62] C. Bienia, S. Kumar, J.P. Singh, K. Li, The parsec benchmark suite: characterization and architectural implications, in: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, 2008, pp. 72–81.
[63] D.H. Bailey, E. Barszcz, J.T. Barton, D.S. Browning, R.L. Carter, L. Dagum, R.A. Fatoohi, P.O. Frederickson, T.A. Lasinski, R.S. Schreiber, H.D. Simon, V. Venkatakrishnan, S.K. Weeratunga, The NAS parallel benchmarks-summary and preliminary results, in: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing, Supercomputing '91, New York, NY, USA, Association for Computing Machinery, 1991, pp. 158–165. https://doi.org/10.1145/125826.125925

Youngsu Cho received the B.S degree in Department of Computer Science and Engineering from Seoul National University of Science and Technology, Seoul, Repbulic of Korea, in 2015. He is currently pursuing his Ph.D. degree in Computer Science and Engineering at Seoul National University, where he has been since 2016. Concurrently, he is working as a Developer at SAP Labs Korea, Seoul, Republic of Korea. Youngsu's research interests include cloud computing, machine learning, data center optimization, etc.

Changyeon Jo received a Ph.D. in Computer Science from Seoul National University in 2021 and a BS in Computer Science from Hanyang University ERICA in 2012. Throughout his Ph.D. studies, he worked on software-based memory disaggregation techniques, performance modeling of VM live migration using machine learning, and efficient VM live migration techniques. Changyeon is currently with MangoBoost where he is working on an RMDA-related DPU software stack.

Reza Entezari-Maleki received the B.S. and M.S. degrees from the Iran University of Science and Technology, in 2007 and 2009, respectively, and the Ph.D. degree from the Sharif University of Technology, in 2014, all in computer engineering. He worked at the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran, as a post-doctoral researcher, from 2015 to 2018. He is currently an assistant professor in the School of Computer Engineering, Iran University of Science and Technology. His main research interests are performance/dependability modeling and evaluation of distributed computing systems.

Jörn Altmann is Professor for Technology Management, Economics, and Policy at the College of Engineering at Seoul National University. Prior to this, he has been a postdoc at EECS and ICSI of UC Berkeley, taught computer networks at UC Berkeley, and worked as a senior scientist at Hewlett-Packard Labs. Dr. Altmann's research centres on Internet economics with a focus on economic analysis of Internet services and on integrating economic models into Internet infrastructures.

Bernhard Egger is a professor in the Department of Computer Science and Engineering at Seoul National University (SNU). He received a combined B.Sc./M.Sc. degree in Computer Science from the Swiss Federal Institute of Technology Zurich (Eidgenössische Technische Hochschule Zürich, ETHZ) in 2001 and a Ph.D. in Computer Science from SNU in 2008. After obtaining his Ph.D., he spent three years as a senior research engineer at the Samsung Advanced Institute of Technology (SAIT) before rejoining SNU in March 2011. His research focuses on efficient runtimes, compilers, and system software for parallel systems.