Computing (2025) 107:64
https://doi.org/10.1007/s00607-025-01423-0

REGULAR PAPER

A comprehensive performance evaluation of container migration strategies

Leonel Feitosa[1] · Vandirleya Barbosa[1] · Arthur Sabino[1] · Luiz Nelson Lima[1] · Iure Fé[1] · Luis Guilherme Silva[1] · Gustavo Callou[2] · Juliana Carvalho[1] · Erico Leão[1] · Tuan Anh Nguyen[3] · Paulo Rego[4] · Francisco Airton Silva[1]

Received: 16 September 2024 / Accepted: 19 January 2025 / Published online: 5 February 2025

Abstract

Containerization has emerged as a transformative technology in modern data centers, offering significant advantages in efficiency and resource management across diverse applications. While it is often associated with platforms like Docker and Kubernetes, its applications extend further and have been widely embraced by leading cloud providers such as Amazon Web Services, Microsoft Azure, and Google Cloud. Effective software resource migration, particularly container migration, is essential in large-scale data centers for managing server downtime, resource consolidation, and disaster recovery, especially in edge mobile computing contexts. Checkpoint/Restore in Userspace enhances container migration by allowing applications to freeze and save their state, facilitating seamless relocation. The study addresses the high cost-both financial and resource-related-of conducting real-world experiments to predict the migration performance of container migrations in real-world scenarios. This paper proposes stochastic Petri net models featuring an absorbing state and the other without. These models assess container migration strategies-Cold, PreCopy, PostCopy, and Hybrid-focusing on metrics such as migration total time (MTT), mean migration time, utilization, discard probability, and migration rate. The models consider the number of elements migrated simultaneously and the system's parallel migration capacity. The absorbing state model also calculates the cumulative probability distribution function. A sensitivity analysis using the design of experiment was also conducted for the hybrid migration policy within the absorbing state model. The results indicate that the cold policy presents lower MTT in scenarios of a high migration arrival rate. At the same time, PostCopy maintains the lowest discard probability, making it suitable for high-demand scenarios.

Keywords Container · Live migration · Performance · Sensitivity analysis · Stochastic Petri net

MSC classification 68M20

Extended author information available on the last page of the article

1 Introduction

Containerization is a method of operating system virtualization that isolates applications from other processes, enabling them to operate in distinct environments. While closely associated with Docker and Kubernetes, container technology has existed since 2008 [1]. Its growing popularity has led major public cloud providers, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud, to support containerized applications on their platforms to manage workloads efficiently [2]. Containers are now essential for modern web applications. Large-scale virtualized computing platforms, such as those found in data centers, require effective software resource migration [3]. Container migration, which supports scheduling server downtime, resource consolidation, and disaster recovery, is a crucial strategy for maintaining service proximity to users, particularly in edge mobile computing contexts [4, 5]. Platforms like Kubernetes [6] and OpenStack [7]. The combination of container technology's flexibility and the efficiency of migration processes highlights the suitability of containers for reallocating resources in dynamic, scalable environments. Stateless memory containers are handy for such reallocations. Kubernetes, for instance, can shut down a stateless container and initiate a new one on a different physical node, streamlining the relocation process and dynamically adjusting to changing demands [8]. However, migrating stateful containers, such as those hosting databases or online game servers, is more complex. The state of these containers must be preserved in volumes and synchronized across replicated containers [9].

Checkpoint/Restore in Userspace (CRIU) provides a solution that enhances Kubernetes' capabilities by allowing running containers or applications to freeze, saving their current states to files. This data can then be used to restore the application to run precisely as it did at the moment of the checkpoint [10]. CRIU supports various migration policies, including Cold, PreCopy, PostCopy, and HybridCopy, each offering different strategies for container transfer between nodes regarding service downtime and migration efficiency [11]. In this paper, we select CRIU as the migration tool due to its popularity and adaptability. Stochastic Petri nets enhance classic Petri nets by incorporating stochastic event timing, making them highly effective for performance modeling in systems where timing is crucial [12]. By integrating randomness with Petri nets' structural modeling capabilities, SPNs offer a detailed analysis of systems with stochastic behaviors, such as those found in environments with asynchronous and concurrent processes. Each transition in an SPN is tied to a probabilistic distribution, typically exponential, to model real-world scenarios where delays and processing times vary predictably [13]. SPNs are ideal for assessing performance metrics like response times, throughput, and resource utilization and for exploring system behaviors under various operational conditions. They are precious in complex scenarios like mobile edge computing (MEC) [14] and real-time data processing applications such as Apache Storm [15], where they help optimize system configurations and predict performance impacts. Furthermore, SPNs can simulate potential bottlenecks and optimize system parameters, reducing the need for costly experimental

testing [16]. Thus, SPNs provide a robust, structured method for performance analysis, proving indispensable in designing and fine-tuning system architectures.

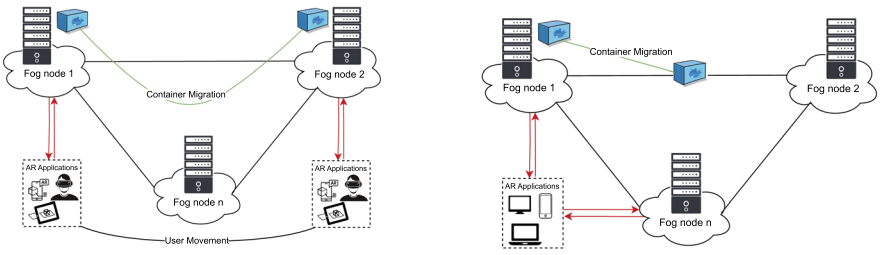
1.1 Problem motivation and illustrative use cases

Container migration plays a pivotal role across various applications within Fog computing. The mobility use case facilitates swift service transfers across fog nodes to ensure low latency and fast responses, which is crucial for supporting mobile users and IoT devices. On the other hand, the orchestration use case focuses on dynamic resource management, optimizing the migration of services between fog and edge data centers to enhance operational efficiency. Both scenarios necessitate migration to sustain performance, latency, and resource management as application demands evolve.

Integrating the Internet of Things (IoT) further underscores the necessity for migration. Devices such as sensors and actuators, whether connected directly to the Internet or via a sensor network, utilize a gateway for communication. This gateway, equipped with computational and storage capacities, facilitates code execution in containers, which are an effective solution in these contexts [17, 18].

Migration is particularly critical in scenarios where the mobility of users or IoT devices is inherent, such as sensors and actuators used in mobile settings or transported by vehicles [19]. For instance, consider an autonomous vehicle that processes environmental data to alert nearby vehicles of potential hazards. Service migration is essential to minimize communication latency and enable quick vehicle responses. Similarly, operators using portable augmented reality devices in industrial settings require real-time data on products and procedures. Through rapid image analysis and augmentation, Fog computing services support augmented reality functionalities effectively, necessitating migration to maintain service quality amidst mobility [20].

Figure 1 illustrates the architectures for both the mobility and orchestration use cases, highlighting the role of container migration in supporting dynamic and responsive systems. In Fig. 1a, the architecture within an industrial context emphasizes the importance of migration to manage data flow and service continuity, especially in time-sensitive scenarios. Here, data from equipment is continuously



(a) Mobility Support

(b) Platform Orchestration

Fig. 1 Container migration use cases

collected, processed, and delivered to operators, who may use augmented reality devices, such as smart glasses, to receive real-time equipment status updates. This setup requires constant data analysis and migration to ensure that the augmented reality applications maintain a seamless flow of information, even as the operator moves within the industrial space. The migration mechanism allows AR applications to adapt quickly to changes in data sources and maintain low latency, essential for real-time decision-making in industrial environments.

Figure 1b showcases the architecture for the orchestration use case, focusing on efficient resource management across a multi-layered infrastructure that spans cloud and fog environments. In this setup, container migration plays a pivotal role in dynamically reallocating resources to meet fluctuating application demands and ensure operational resilience. By enabling seamless service migration between nodes, this architecture helps balance the computational load across the fog layer, which is closer to end-users, and the cloud layer, where more extensive resources are available. For instance, when an edge node experiences high traffic or resource constraints, services are migrated to a neighboring fog node or even escalated to the cloud if needed, thereby minimizing latency for end-users while optimizing resource use. This layered approach not only aids in load balancing but also supports fault tolerance, allowing the system to adapt to failures or spikes in demand without interrupting service continuity. Migration in this context requires careful coordination to ensure minimal downtime and maintain data consistency across nodes, as multiple instances of a service may need to synchronize in real time. This is particularly important in scenarios with critical applications, where service reliability and responsiveness are paramount.

This study addresses these needs by proposing an applicable approach to predict container migration performance using SPN models. We introduce an SPN model with an absorbing state, which allows for estimating the MTT for batch container migrations and a cumulative probability distribution function (CDF) to assess migration completion within specified timeframes. A sensitivity analysis of this model helps understand how different variables affect migration performance. An SPN model without an absorbing state also calculates classical performance metrics such as mean migration time (MMT), resource utilization, discard probability, and migration rate. Case studies in the paper guide the application of these models, showing adaptability to variations in service time parameters, arrival rates, and parallel processing capabilities.

1.2 Key contributions and findings

The research presented in this paper makes several substantial contributions to the field of container migration technologies. These are detailed as follows:

- *Development of absorbing and non-absorbing spn models for container migration* The study introduces two sophisticated SPN models designed to simulate and predict the performance of container migrations across different

strategies (Cold, PreCopy, PostCopy, and hybrid), enabling a comprehensive assessment of migration dynamics and potential bottlenecks.

- *Extensive utilization of a suite of performance metrics for evaluating container migration efficiency* The research leverages an array of meticulously selected performance metrics, including MTT, MMT, system utilization, discard probability, and migration rate to evaluate the efficacy of various container migration strategies quantitatively.
- *Conduct of a DoE-based sensitivity analysis on hybrid migration policy* This part of the study employs a systematic sensitivity analysis using the design of experiment (DoE) approach, specifically focusing on the hybrid migration policy to identify and analyze factors that significantly influence the performance of container migrations.
- *Exploration of the impact of simultaneous parallel container migrations on system performance* The research explores the operational impacts and benefits of executing multiple container migrations in parallel, enhancing the understanding of system scalability and efficiency in response to varying operational loads.
- *Empirical validation and scenario-based assessment of proposed SPN models* By subjecting the developed SPN models to rigorous validations through real-world experiments and detailed scenario analyses, the study confirms the models' precision and utility in forecasting the outcomes of container migrations.

The study led to several key findings that enhance our understanding of the dynamics and efficiency of container migrations:

- *Predictive accuracy of SPN models in foreseeing container migration outcomes* The key finding demonstrates that the newly developed SPN models reliably predict various aspects of container migration processes, aligning accurately with real-world experimental data, thus confirming their applicability and predictive prowess.
- *Varied influences of distinct migration policies on key performance metrics* The analysis distinctly highlights how different migration policies variably impact performance metrics, with strategies like Hybrid and PreCopy showing advantages in terms of resource utilization and lower probabilities of migration discards compared to the cold migration approach.
- *Dynamics of parallel migration processes and their scalability limits* Increasing the scale of parallel migrations improves the migration process efficiency significantly up to a saturation point, beyond which the incremental benefits diminish, indicating an optimal scale for parallel migrations.
- *Identification of dominant factors affecting migration efficiency through sensitivity analysis* Sensitivity analysis reveals that the number of containers being migrated and the specific migration policy adopted are pivotal in determining the overall efficiency and timing of the migration process.

The implications of these findings are profound and suggest several actionable strategies for the industry:

- *Strategic enhancement of container migration strategies for optimized performance* Insights from this study enable IT infrastructure managers and system architects to refine and customize container migration strategies to optimize performance and reliability in real-world settings.
- *Informed selection of appropriate migration policies based on performance impact* The detailed understanding of the impact of various migration policies provided by this study aids stakeholders in choosing the most suitable policies that align with their specific operational needs and constraints.
- *Efficient resource allocation and system scaling based on parallel migration insights* The findings concerning parallel migrations provide valuable guidelines for resource allocation and system scaling, ensuring that resources are optimally utilized and the system's capacity is aligned with demand.

1.3 Paper organization

The structure of this document is outlined as follows: Sect. 2 provides a background overview, discussing fundamental concepts and theories essential for understanding the scope of our research. Section 3 reviews related literature and positions our research within the existing body of knowledge. Section 4 describes the methodology applied in our study. Section 5.1 explains the SPN model that includes an absorbing state, provides validation for this model, and discusses a relevant case study and a sensitivity analysis. Section 5.2 elaborates on the SPN model without an absorbing state, along with a detailed case study. Section 7 discusses the findings and limitations, including the constraints of physical infrastructure. Finally, Sect. 8 concludes the paper by summarizing the key findings from our research.

2 Migration policies

This section introduces key concepts for understanding container migration platforms that use technologies like Kubernetes and runC. runC, developed under the open container initiative (OCI), is a lightweight, portable solution for container execution employed by Docker and other container-based virtualization systems. For container migration on Linux systems, runC integrates with CRIU, a software component offering checkpoint/restore functionality for Linux applications. We explore four container migration techniques-Cold, PreCopy, PostCopy, and hybrid-through SPN modeling in Figs. 2, 3, 4, and 5, to assess their performance.

Cold migration policy architecture Figure 2 illustrates the cold migration policy stages: checkpoint, transfer, and restore. The checkpoint stage involves freezing the container at the source node and capturing metadata such as CPU state and memory contents. This metadata is then transferred to the destination node during the transfer stage. At the destination, the container service resumes from the frozen state during the restore phase [21, 22]. The total migration time, T_{cm}, is the sum of checkpoint, transfer, and restoration times:

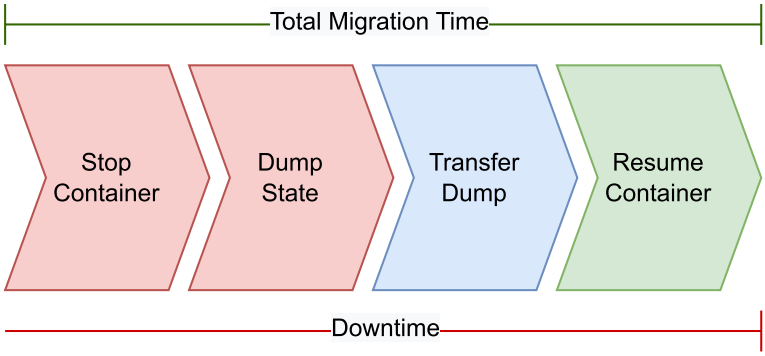


Fig. 2 Cold migration method

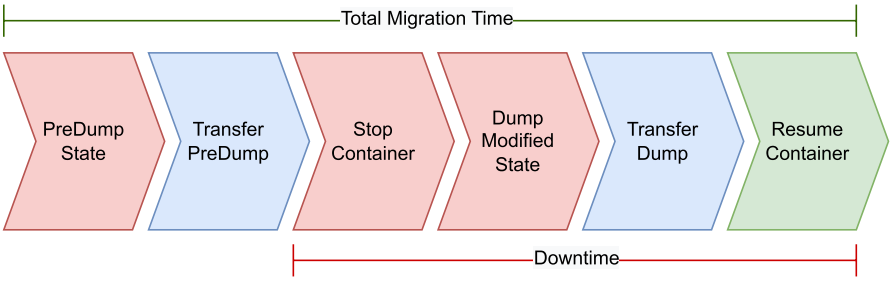


Fig. 3 PreCopy migration method

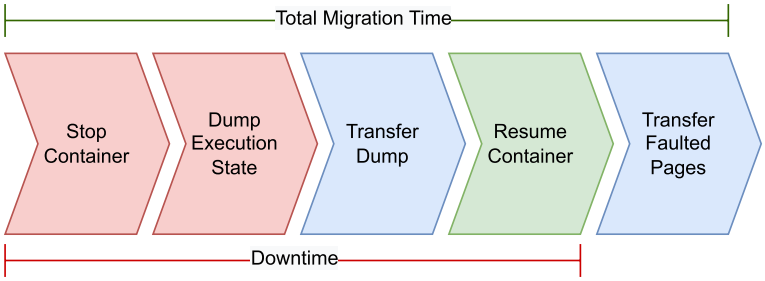


Fig. 4 PostCopy migration method

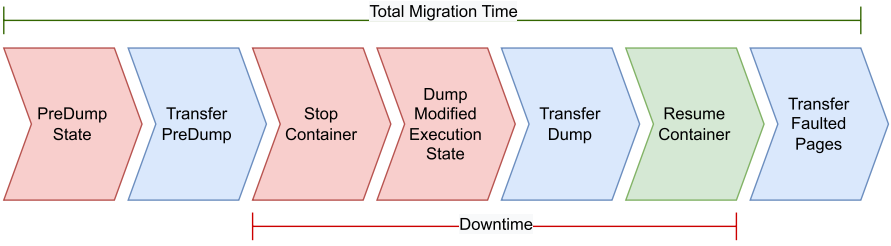


Fig. 5 Hybrid migration method

$$T_{cm} = T_{sc} + T_t + T_r \tag{1}$$

Here:

- T_{cm} is the migration total time,
- T_{sc} is the checkpoint time,
- T_t is the transfer time,
- T_r is the restoration time.

During cold migration, T_{cm} also represents the downtime of the container service.

PreCopy migration policy architecture Figure 3 depicts the PreCopy migration method, which includes stages of PreDump State Save, PreDump Transfer, Container Pause, and Changes Save. In the PreDump stage, initial state and memory information are collected, while the save modifications stage gathers changes in memory pages. This metadata is transferred to the destination node for restoration [23]. The migration total time for the PreCopy method is given by:

$$T_{prm} = T_{pd} + T_{pdt} + T_d + T_{dt} + T_r \tag{2}$$

where:

- T_{prm} is the full container migration time under PreCopy,
- T_{pd} is the PreDump time,
- T_{pdt} is the PreDump data transfer time,
- T_d is the Dump time,
- T_{dt} is the Dump data transfer time,
- T_r is the restoration time.

The advantage of PreCopy is that the service remains active during the PreDump phase.

PostCopy migration policy architecture The PostCopy method, shown in Fig. 4, reverses the PreCopy strategy and consists of container pause, state save, state transfer, container restore, and requested page transfer phases. It is based on the demand paging strategy, known as *lazy* migration, where the resumed container accesses missing memory pages, causing page faults handled by the *lazy* Page Daemon [24]. The migration time for PostCopy is:

$$T_{ptm} = T_{sc} + T_{td} + T_r + T_{tfp} \tag{3}$$

where:

- T_{ptm} is the full migration time under PostCopy,
- T_{sc} is the checkpoint time,
- T_{td} is the dump transfer time,
- T_r is the restoration time,
- T_{tfp} is the failed page transfer time.

This method efficiently manages data volume by copying each memory page only once.

Hybrid migration policy architecture Figure 5 represents the hybrid migration approach, blending the strengths of PreCopy and PostCopy. It begins with a PreDump of the entire state, followed by a pause and dump of the state, combining both strategies. Any changes during the PreDump are transferred to the destination node along with the entire state [25]. The migration time for the Hybrid approach is:

$$T_{hcm} = T_{pd} + T_{pdt} + T_d + T_{dt} + T_r + T_{tfp} \tag{4}$$

where:

- T_{hcm} is the full hybrid migration time,
- T_{pd} is the PreDump time,
- T_{pdt} is the PreDump data transfer time,
- T_d is the Dump time,
- T_{dt} is the Dump data transfer time,
- T_r is the restoration time, and
- T_{tfp} is the failed page transfer time.

This method leverages the best aspects of both PreCopy and PostCopy to optimize migration efficiency and minimize downtime.

The Table 1 outlines the main pros and cons of the migration policies discussed in this study, comparing the cold, PreCopy, PostCopy, and hybrid strategies. The analysis highlights that each policy has specific strengths and limitations, making them suitable for different contexts. For instance, cold minimizes drop rates under heavy loads but results in more downtime, while hybrid optimizes efficiency but adds complexity and increases the risk of packet loss. Ultimately, selecting the ideal policy depends on the system's availability and resource requirements.

2.1 CRIU - checkpoint restore in userspace

CRIU, a powerful tool for application migration, allows for state preservation while transferring applications from one *host* to another. It is beneficial in edge computing, where resource constraints may require application migration. By maintaining the application state during transfers, CRIU enhances system reliability, availability, and fault tolerance [26]. Technologies that support CRIU include Docker, LXC, Podman, and RKT. Notable use cases for CRIU are:

- *Fault tolerance and disaster recovery* CRIU facilitates fault tolerance and disaster recovery in edge computing by enabling the migration of applications to alternative *hosts* in response to failures or disasters, thereby maintaining service availability and minimizing downtime.
- *Mobility* CRIU supports mobility in edge computing environments by allowing applications to migrate between *hosts*, enabling scenarios such as mobile edge computing where services can follow users.

Table 1 Pros and Cons of each migration policy [21–25]

Migration policy	Pros	Cons
Cold	Lower packet loss under high load conditions; Reduced mean migration time (MTT) under heavy load; Requires less active memory during migration	High downtime; Ineffective for low-latency applications; Does not support live migrations; Limited in scenarios requiring high availability
PreCopy	Service remains active during pre-copy phase; Supports live migrations; Minimizes service interruption; Increases fault tolerance during migration	Longer migration time compared to Cold; High resource demand; Delays due to retransmission of modified pages; Increased bandwidth usage
PostCopy	Transfers each memory page only once; Reduces bandwidth usage; Minimizes execution time on the target node	Susceptible to page faults; High downtime if faults occur; Potential instability if network fails
Hybrid	Combines advantages of PreCopy and PostCopy; High performance under heavy load; Reduces downtime and bandwidth usage	Higher CPU and memory consumption; Susceptible to execution failures and page faults; Requires significant bandwidth and memory under high load

- *Dynamic scaling* By dynamically relocating applications based on demand, CRIU aids in optimizing resource utilization and responsiveness in edge computing systems.
- *Energy efficiency* CRIU can contribute to energy efficiency by transferring applications to less active *hosts*, thereby reducing overall energy consumption and enhancing sustainability.

Recent developments have introduced decentralized container management and orchestration protocols, such as Caravela [27] and Nion Network [28]. These networks employ a decentralized node arrangement to evenly distribute container loads, using scalable blockchain consensus mechanisms for secure decision-making. This arrangement allows for effective migration management and failure mitigation through redundancy, where multiple nodes host containerized applications to provide backups, enhancing network efficiency and reducing the risk of state losses. CRIU supports these networks by capturing regular application snapshots and facilitating quick restoration on alternative nodes after failures.

3 Related work

This section reviews significant studies relevant to the methodologies and applications discussed in this paper, specifically focusing on migration policies, performance metrics, evaluation methods, and tools. We categorize related works based on their migration policies due to their critical role in the migration total time, discarded containers, and downtime. Table 2 outlines the contributions of critical works related to our research.

3.1 Studies on PreCopy migration policies

The PreCopy migration policy is a focal point in research due to its efficacy in live migrations, mainly because it minimizes downtime and enhances predictability, which is vital in environments that prioritize continuous service availability. Xu et al. [33] introduced Sledge, a live migration system that maintains component integrity during migrations by managing images and contexts seamlessly. CloudHopper, developed by Benjaponpitak et al. [36], facilitates live container migrations across multiple cloud platforms, automating connection maintenance and traffic redirection. Fan et al. [39] proposed a locale live migration template for Docker containers using the PreCopy policy, considering factors like container proximity, bandwidth, and costs. Smimite et al. [42] explored hybrid virtualization for live-migrating monolithic applications, like video streaming in nested containers, focusing on processor and disk utilization, migration timing, and downtime.

ELASTICDOCKER, proposed by Al-Najjar et al. [43], autonomously manages the vertical elasticity of Docker containers, performing live migrations when resources on the host machine are insufficient. Bhardwaj et al. [48] evaluated the performance of the LXD/CR container migration technique compared to PreCopy

Table 2 Related work

Work	Policies	Metrics	Use of CRIU	Sensitivity analysis	Evaluation method
[10]	Cold	MRT	✓	No	Measurement
[29]	PreCopy, PostCopy, hybrid	MRT	✓	No	Measurement
[30]	Non-explicit	MRT, network bandwidth, network latency	✓	No	Measurement
[31]	Non-explicit	MRT	✓	No	Measurement
[9]	Non-explicit	MRT, downtime, usage	✓	No	Measurement
[32]	Non-explicit	MRT	No	No	Measurement and modeling
[33]	PreCopy	MRT, downtime, image extraction time	✓	No	Measurement
[34]	Non-explicit	Usage	✓	No	Measurement
[35]	PreCopy, PostCopy, hybrid	MRT, downtime	✓	No	Measurement
[36]	PreCopy	MRT, throughput	✓	No	Measurement
[37]	Non-explicit	Usage, response time	✓	No	Measurement
[38]	Cold, PreCopy	MRT, downtime	✓	No	Measurement
[39]	PreCopy	MRT, usage	No	No	Measurement
[40]	Non-explicit	MRT, downtime	No	No	Measurement
[41]	Cold, PreCopy, PostCopy	MRT, downtime, confidentiality, integrity	No	No	Measurement
[42]	PreCopy	MRT, memory consumption, network traffic	✓	No	Measurement
[43]	PreCopy	Usage, response time	✓	No	Measurement
[44]	PreCopy, PostCopy	MRT, downtime, aging	✓	No	Measurement
[45]	Non-explicit	Throughput, latency	✓	No	Measurement
[46]	Non-explicit	R2, MAPE, MAE	✓	No	Modeling
[47]	Cold	Usage	✓	No	Measurement
[48]	PreCopy	MRT, downtime, usage	✓	No	Measurement
[49]	PreCopy	MRT, downtime	✓	No	Measurement

Table 2 (continued)

Work	Policies	Metrics	Use of CRIU	Sensitivity analysis	Evaluation method
[50]	Non-explicit	MRT, downtime	✓	No	Measurement
[51]	Cold, PreCopy, PostCopy	Migration rate, energy	✓	No	Measurement
This work	Cold, PreCopy, PostCopy, hybrid	MRT, MMT, discard probability, utilization, migration rate	✓	✓	Measurement and modeling

VM migration schemes. Ramanathan et al. [49] analyzed live migration in VM and container virtualizations to support network functions virtualization (NFV) requirements.

3.2 Studies on multiple migration policies

Adopting multiple migration policies often reflects the specific needs of the context studied, aiming to assess how different conditions impact migration outcomes and determine the most suitable policy for each scenario. Stoyanov et al. [29] utilized CRIU's image cache/proxy feature to enhance migration times and reduce downtimes by bypassing secondary storage delays. Govindaraj et al. [35] introduced a redundancy migration scheme using CRIU to decrease downtime relative to traditional Linux container migrations. Chou et al. [44] implemented a checkpoint-based migration using a shared non-volatile memory pool, evaluating downtime, migration time, and application performance. Ramanathan et al. [38] developed a framework to migrate virtualized network functions into containers, comparing the efficiency and service interruptions between VMs and containers. Pecholt et al. [41] focused on maintaining integrity and confidentiality in live OS container migrations, utilizing encrypted VMs for transparent and continuous execution.

3.3 Studies without specified migration policies

Several studies have introduced innovative approaches that have yet to specify their underlying migration policies. Ma et al. [30] developed an edge platform for mobile service migration, emphasizing a layered storage structure to minimize overhead. Di et al. [31] proposed a real-time Docker container migrations tool, optimizing multi-container strategies to reduce migration total time. Tay et al. [32] provided a mathematical framework for workload placement and migration in data centers, comparing the performance impacts on containers and VMs. Gonzalez et al. [34] developed Herd-Monitor, a system for monitoring container and host performance metrics. Abdullah et al. [37] proposed an algorithm to manage container migrations based on resource availability, deadlines, and node locations, ensuring migrations to optimal nodes during events.

3.4 Studies on cold migration policies

The cold migration policy is characterized by its non-real-time transfer approach, which generally has less impact on availability compared to live migrations, though it is less frequently studied. Torre et al. [10] investigated container migrations under various conditions to measure performance and migration duration. Karhula et al. [47] used Docker and CRIU for checkpoints in long-running functions on IoT devices, aiming to enhance resource efficiency during execution.

3.5 Key contributions

This work introduces two SPN models for analyzing container migration policies, significantly extending the current research landscape as presented in Sect. 2. Below, we outline the five primary contributions of our study:

- *Policies* Our research extensively explores a variety of migration policies, providing a detailed analysis of different policy configurations. This approach enables a thorough understanding of the advantages and limitations associated with each policy, tailored to specific computational infrastructures and service level agreements in public computing environments. The integration of performance evaluation across four distinct policies -cold, pre-copy, post-copy, and hybrid- and analytical modeling distinguishes our work from existing studies.
- *Metrics* We have selected and utilized five key metrics to evaluate migration performance:
  - *MTT* Measures the time from the migration process's start to the container's availability on the destination host.
  - *MMT* Calculates the average time taken to migrate all containers.
  - *Discard probability* Assesses the likelihood of a container failing to complete the migration.
  - *Utilization* Observes the use of computational resources by the migration system.
  - *Migration rate* Tracks the frequency of container transfers within the computing environment.

  The comprehensive application of these metrics provides a robust framework for planning and optimizing container migration.
- *Evaluation method* The modeling techniques employed in this study provide a systematic representation and analysis of various container migration strategies. SPN modeling allows for the mathematical prediction and understanding of system behaviors under various conditions, as it performs a stationary analysis, yielding results that are truly accurate due to its reliance on exact calculations [52]. The use of analytical models, which are both cost-effective and highly representative, offers a significant advantage over traditional empirical measurement methods, which typically involve higher costs and complexities.
- *Sensitivity analysis* We used the DoE sensitivity analysis to identify critical factors influencing container migration performance. This analysis, unique in container migration studies, enables a deep dive into the migration process, pinpointing elements that significantly impact the MTT. The capability of DoE to refine system optimization processes is particularly notable.
- *Parallel migration* A novel aspect of our study is the focus on parallel container migration, which involves simultaneous transferring multiple containers across different environments. This approach is crucial during unexpected system failures and enhances the resilience and efficiency of the migration process. Our SPN model adeptly handles the computation of relevant metrics for any num-

ber of containers undergoing migration, showcasing its scalability and real-world applicability in dynamic settings.

4 Methodology for modeling and evaluating migration policies

This study aims to develop a model to evaluate container migration using four distinct migration policies, as summarized in Fig. 6. The methodology involves several steps:

1. *Understanding the application* This initial phase involves comprehending the application's operation, identifying key components, detailing the data flow, and specifying metadata transmission during migration. It is crucial to understand the limitations of the migration tools used.
2. *Definition of metrics* We define relevant metrics to assess system performance, including MTT, MMT, discard probability, usage, and migration rate.
3. *Parameter definition* Parameters are set to dictate the behavior and capacity of migration system components, including the number of containers to be migrated (K) and the associated migration capacity (C).
4. *Generation of analytical models* We develop two SPN performance models-one with and one without an absorbing state-considering the defined metrics and parameters to suit the low-complexity nature of container migration scenarios.

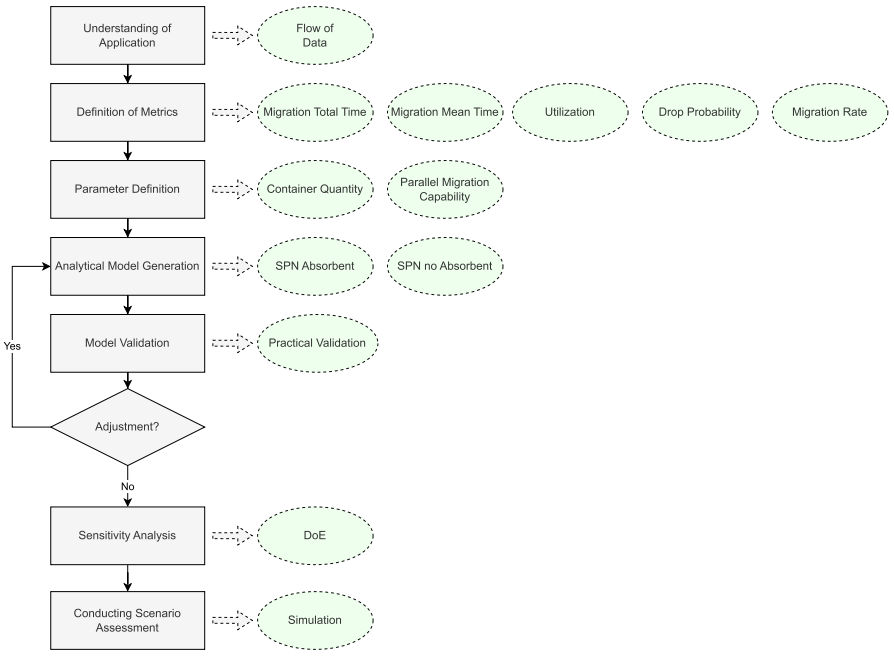


Fig. 6 Methodology for developing and using the analytical model

5. *Model validation* The model is tested in a real environment with actual container migration, and results are evaluated using statistical methods like confidence intervals and T-tests. Any discrepancies lead to adjustments in the model.
6. *Sensitivity analysis* A DoE approach identifies vital factors influencing container migration. This analysis helps understand the effects of factor-level variations and their interactions on system performance.
7. *Conducting scenario assessment* Two scenarios are simulated using SPN. The first limits the total number of containers that can be migrated using a model with an absorbing state. The second assesses migration based on migration rates using a stateless model. Simulations help adjust relevant factors and measure performance across different configurations, ultimately determining optimal settings for container migration.

This structured approach ensures a comprehensive evaluation of container migration, addressing various operational, environmental, and technical aspects to optimize system performance and reliability.

5 Proposed SPN models

This section presents the SPN models developed to analyze the performance of different container migration strategies. Two main models are described: one with an absorbing state and another without an absorbing state. These models are designed to represent the stochastic behaviors involved in the migration process, enabling the simulation of scenarios with multiple containers and parallel migration strategies.

5.1 SPN absorbing model

This study used a modeling approach to analyze and simulate different container migration strategies. Unlike direct measurements, constrained by specific parameters and execution environments, modeling provides flexibility in predicting system behavior under various configurations. For instance, while a direct measurement might show the migration outcomes for two elements, modeling can extrapolate this to envision the behavior with one hundred elements migrating concurrently within a capable infrastructure.

Consequently, this section introduces an SPN model designed to compute the performance characteristics of networks using different container migration strategies. This model is intended to aid system administrators in tuning parameters to optimize performance. Moreover, it allows the evaluation of potential system modifications before actual implementation.

Figure 7 depicts the proposed absorbing SPN model, which is further explained in this section. Previously, in Sect. 2, we described each migration policy's operation and detailed each component's functionalities. This section focuses on the overall functioning of the model within the migration process.

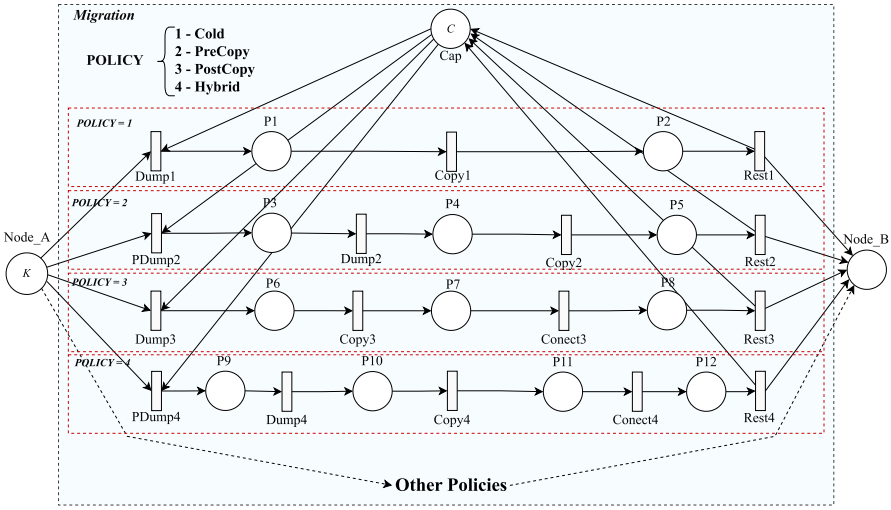


Fig. 7 SPN model to calculate migration total time using different policies

In the model, *tokens* represent containers to be migrated, initially located at Node_A. These *tokens* can migrate to Node_B, provided there is sufficient migration capacity, represented by the capacity marking (C) at place Cap. Node_B is an absorbing node, where the migration completes upon the arrival of *tokens*.

The transition of *tokens* from Node_A to Node_B is contingent on available migration capacity, and the migration path followed by each *token* is determined by guard conditions associated with transitions such as Dump1, PDump2, Dump3, and PDump4. These transitions are influenced by the POLICY variable, which determines the migration strategy:

- `POLICY = 1`: triggers the cold migration policy.
- `POLICY = 2`: initiates the PreCopy migration policy.
- `POLICY = 3`: activates the PostCopy migration policy.
- `POLICY = 4`: engages the hybrid migration policy.

It is important to note that the model can accommodate N different migration policies. The transitions within this model follow an exponential probability distribution to align with real-world scenarios, as confirmed by validation experiments. Each transition employs *infinite server* semantics, allowing simultaneous activations by multiple *tokens* [53].

The model's performance, specifically the probability of all *tokens* transitioning from Node_A to Node_B, is quantified using the CDF metric, calculated based on the state of Node_B. Table 3 provides an overview of the model's key elements, illustrating its comprehensive structure and potential applications in various migration scenarios.

Table 3 Description of the main elements of the model

Type	Element	Description
Places	Node_A	Location where the migration element to be migrated is located
	Node_B	Location to which the migration element will be migrated
	Cap	Migration capacity is the maximum simultaneous container migrations, influenced by system resources, network bandwidth, and strategy efficiency.
Timed transitions	Dump1	Time associated with creating the restore point
	Copy1	Associated time to transfer element metadata to be migrated
	Rest1	Associated time for restoring the container to the policy cold migration
	PDump2	Time associated with checking the application on the source node
	Dump2	Time associated with collecting information from modified memory pages
	Copy2	Time associated with transferring page information modified memory
	Rest2	Associated time for restoring the container to the policy PreCopy migration
	Dump3	Time associated with creating the restore point
	Copy3	Time associated with transferring page information memory
	Conect3	Associated time to connect Node_A to Node_B
	Rest3	Associated time for restoring container to policy PostCopy migration process and transfer of failed pages
	PDump4	Time associated with checking the application on the source node
	Dump4	Time associated with creating the restore point
	Copy4	Time associated with transferring page information modified memory
	Conect4	Time associated to connect Node_A to Node_B
	Rest4	Associated time for restoring the container to the policy hybrid migration process and transfer of failed pages
Seating markings	K	Container to be migrated
	C	Maximum migration capacity allowed simultaneouslys

5.1.1 Metric

The MTT is the average time required to migrate all designated elements. It represents the time expected to reach a state of marking deadlock, which is a condition in Petri nets and Markov chains where a *token* cannot move to any further states. We employ transient analysis to calculate the absorbing state, which is the state where the *token* has no transitions. SPN generates continuous-time Markov Chain (CTMC) and uses numerical methods to determine the time to absorption [54].

The dynamics of the CTMC are governed by the Kolmogorov equation, with the initial probability vector $\pi(0)$, as shown in Eq. 5. The expected total time that the CTMC spends in state i before time t is given by Eq. 6. The time to absorption, considering only non-absorbing states (denoted as N) is determined by $lim_{t\to\infty} L_N(t)$. This condition satisfies Eq. 7, where $\pi_N(0)$ is the initial probability vector restricted to the states in set N and Q_N is the infinitesimal generator matrix for the non-absorbing states [55]. The overall MTT is computed as shown in Eq. 8, representing the sum of times spent in each non-absorbing state until absorption [56].

$$\frac{d\pi(t)}{dt} = \pi(t)Q \tag{5}$$

$$L_i(t) = \int_0^t \pi_i(x)dx \tag{6}$$

$$L_N(\infty)Q_N = -\pi_N(0) \tag{7}$$

$$MTT = \sum_{i\in N} L_i(\infty) \tag{8}$$

Using Fig. 8 as an illustrative example, MTT is calculated based on the probabilities of a *token* transitioning from the start to the end. The migration process involves transitions labeled T1 and T2. The parallel migration of K elements is facilitated by the availability of maximum migration capacity allowed simultaneouslys, denoted by the capacity marking (C) [53, 57]. This model allows us to understand and quantify the migration times under various configurations and conditions.

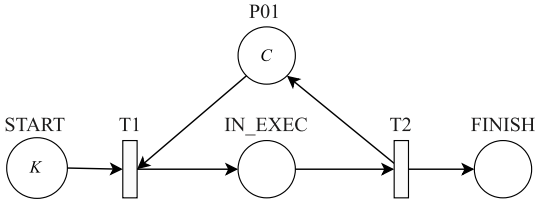


Fig. 8 Example of SPN model

5.2 SPN non-absorbing model

This section introduces a non-absorbing stateless SPN model designed to assess the dynamics and outcomes of various container migration policies. These policies, detailed in Session 2, incorporate multiple stages, including rules, copying, and processing, which together influence the choice and efficacy of a migration policy. The model developed here calculates four key metrics: MMT, discard probability, utilization, and migration rate, with a focus on the impact of varying *arrival delay* (AD). This approach allows data center administrators to effectively plan and optimize the migration process during the initial phases of computational architecture design.

Figure 9 depicts the SPN model that operates without an absorbing state, continuously receiving new migration requests at intervals determined by the AD. This setup ensures a constant inflow of containers into the migration process, unlike models with an absorbing state where input ceases once all tokens are absorbed.

In this model, each migration request enters the system through a source node, which is represented by a gray-colored transition with a deterministic distribution, ensuring a new container is ready to migrate at each specified interval. The overall flow of the model mirrors that of its absorbing counterpart, with transitions and places guiding the migration process.

A crucial distinction in the non-absorbing model is the presence of multiple pathways that a token can take at certain junctions within the network, specifically at places P2, P6, P8, and P14. Here, tokens face immediate transitions (TI1, TI2, TI3, or TI4), which, if taken, indicate a failure in the migration process due to issues in copying from Node_A to Node_B. Conversely, if tokens proceed through transitions TI5, TI6, TI7, or TI8, the migration continues successfully without errors in the copying stage.

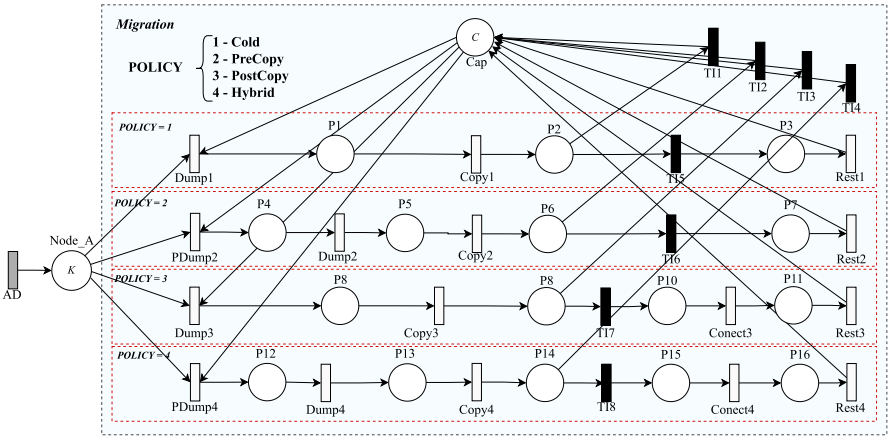


Fig. 9 Non-absorbing model

5.2.1 Metrics

The MMT is derived using Little's Law, which requires a stable system where the request rate is lower than the server's processing rate [58]. Little's Law equates MMT to the product of the number of requests within the system and the interarrival time (AD). Here, MMT is analogous to the average response time (MRT), and requests are the migrating containers. The count of elements within the system is summed up as tokens across all relevant places. For our model, this sum is easily obtained through $C - Esp\{Cap\}$, where C denotes the maximum total parallel migration and $Esp\{Cap\}$ represents the expected number of tokens at place Cap at any given time.

$$\mathbf{MMT} = (C - Esp\{Cap\}) \times AD \tag{9}$$

`Discard Probability (DP_PROB)`: This metric estimates the likelihood of request losses when no queuing capacity remains at the system input. It is calculated using the probability that all capacity is utilized ($\#Cap = 0$) and there are pending tokens at the source ($\#Node\_A > 0$).

$$\mathbf{DP\_PROB} = P\{(\#Cap = 0) \wedge (\#Node\_A > 0)\} \tag{10}$$

`Utilization (U)`: Utilization is the ratio of the expected number of tokens in active migration to the total migration capacity. It reflects how effectively the system resources are being used.

$$\mathbf{U} = \frac{C - Esp\{Cap\}}{C} \tag{11}$$

`Migration Rate (MR)`: The migration rate is defined by the throughput of migrations, measured by dividing the number of tokens at a particular place by the time of the corresponding transition. This rate varies depending on the active migration policy.

$$\mathbf{MR} = \frac{Esp\{P3\}}{T\{Rest1\}} + \frac{Esp\{P7\}}{T\{Rest2\}} + \frac{Esp\{P11\}}{T\{Rest3\}} + \frac{Esp\{P16\}}{T\{Rest6\}} \tag{12}$$

6 Empirical evaluation of migration policies

This section presents an empirical evaluation of container migration policies, analyzing the cold, PreCopy, PostCopy, and hybrid strategies in two case studies using absorbing and non-absorbing SPN models. Additionally, we present the results of an experiment conducted in a controlled environment to validate the modeling.

6.1 Empirical validation

Validating an analytical model is crucial to confirm its accuracy and relevance. This validation compares model predictions with real-world measurements or

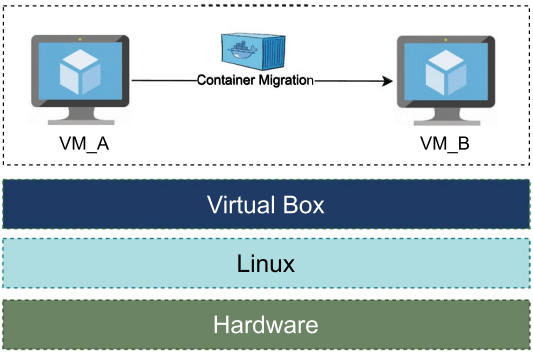


Fig. 10 Experiment setup

simulation outcomes to ensure reliability [58]. Here, we describe the validation of our proposed absorbing SPN model, specifically the MTT, against observed data from real-world experiments.

For this purpose, we constructed a synthetic prototype and deployed it in a controlled test environment. The experimental setup is depicted in Fig. 10 and involved running various migration policies-cold, PreCopy, PostCopy, and hybrid-as detailed in Sect. 2. We utilized two virtual machines, each equipped with 5000 MB of RAM and two processing cores, running Ubuntu 22.04 and Linux Kernel, with CRIU 3.17.1 for *checkpoint* and *restore* operations, OpenSSL 3.0.2 for secure file transfers, and RunC 1.1.9 for container runtime. The hardware setup includes an AMD Ryzen 5 3500U processor with Radeon Vega Mobile Graphics, operating at a maximum frequency of 2.1 GHz and a minimum of 1.4 GHz, with 4 cores and 8 threads (2 threads per core), along with an L1 cache of 128 KiB (data) and 256 KiB (instructions), L2 cache of 2 MiB, and L3 cache of 4 MiB. The machine has 16 GiB of RAM and a 240 GB SSD with a transfer speed of 77.6 MB/s. The size of the migrated files was approximately 4 MB.

The application used for testing is a Python script calculating Fibonacci sequences, designed to be RAM and CPU-intensive. It generates images for each Fibonacci number and stores these in RAM, inspired by Dayo et al. [59] similar approach with Docker containers. This setup ensures substantial memory consumption during the test.

The experimental flow, maps out the container migration steps for each policy. The cold policy sequence involves direct dumping and copying followed by a restart. PreCopy and PostCopy include additional intermediate steps such as Pdump2 and Dump2 for PreCopy and Dump3 and Copy3 for PostCopy. The Hybrid approach begins with Pdump4, then moves through Dump4 and Copy4 to Restart4. We meticulously recorded the duration of each step for later analysis and logged the MTT after each restart. The CRIU documentation[1] provided guidelines for accurate policy implementation.

We executed each policy 50 times to ensure the robustness of our findings. The experimental data were used to populate the transition times in the SPN model, enhancing the model's precision.

[1] CRIU Documentation: https://criu.org/.

Table 4 Time configuration parameters used in the case study

POLICY	Name	Time(s)
Cold	Dump1	0.84352
	Copy1	4.55536
	Rest1	0.95682
PreCopy	PDump2	0.77930
	Dump2	0.75376
	Copy2	9.35288
	Rest2	0.90658
PostCopy	Dump3	4.98894
	Copy3	0.54768
	Conect3	3.70338
	Rest3	0.63900
Hybrid	PDump4	0.69168
	Dump4	5.48088
	Copy4	4.06736
	Conect4	4.11074
	Rest4	0.71172

Table 5 Comparison of measurement results with the proposed model

Policies	MTT (s)		Confidence interval		Standard deviation		*P*-value
	Exp	Mod	Exp	Mod	Exp	Mod	
Cold	6.355	6.308	[6.405–6.305]	[6.019–6.598]	2.161	4.659	0.949
PreCopy	11.792	11.847	[11.742–11.842]	[11.269–12.425]	2.733	9.317	0.968
PostCopy	9.879	9.749	[9.829–9.929]	[9.369–10.128]	0.902	6.121	0.883
Hybrid	15.062	15.269	[14.743–15.795]	[14.743–15.795]	1.260	8.471	0.865

Table 4 lists the parameters from the experiment used to calibrate the model, while Table 5 displays the validation results, the results can also be observed in the graph shown in Fig. 11. The SPN model results were derived through simulation with a 2% error margin. We employed a two-sample T-test[2] to compare the experimentally observed MTT with the model-generated MTT statistically. All data sets followed a normal distribution. The T-test's *p*-values were above 0.05, indicating no significant difference between the model predictions and experimental results at a 95% confidence level. Thus, the model accurately reflects real-world operations, validating its effectiveness and reliability in simulating container migration scenarios.

[2] Sample T-Test https://tinyurl.com/yanthw4e.

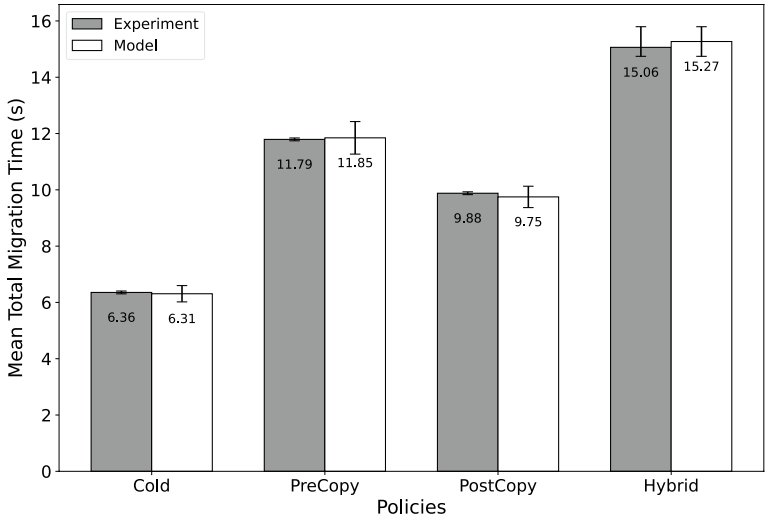


Fig. 11 Comparison of measurement results with the proposed model

6.2 Case study: absorbing model

This section presents simulation results using the proposed absorbing SPN model to explore its range of applications. The primary focus is on two parameters: the number of elements to be migrated and the parallel migration capacity. By analyzing these parameters, we aim to understand the model's behavior and its implications on real system performance. The transition times incorporated in the model are those derived from the validation phase. Stationary analysis is widely used for simulation in certain contexts due to its ability to provide accurate results based on exact calculations[60]. Results and numerical analyses were conducted using the Mercury tool [61].

Impact of element quantity on MTT

Figure 12 illustrates the effect of varying the number of elements (K) on the MTT. We tested with K values ranging from 10 to 100 in increments of 10. For these simulations, the parallel migration capacity was fixed at two agents (C = 2). As expected, the MTT increases with the number of elements due to the limited parallel migration capacity. Notably, the hybrid policy exhibited the highest MTT at approximately 750 s, whereas the cold policy consistently showed the lowest MTT. The divergence between the policies becomes more pronounced as the number of elements increases, with the hybrid policy's MTT rising sharply by 247 s in the range K=[50–100], compared to a 158-second increase for the cold policy. The choice of migration policy becomes critically important as the volume of elements increases.

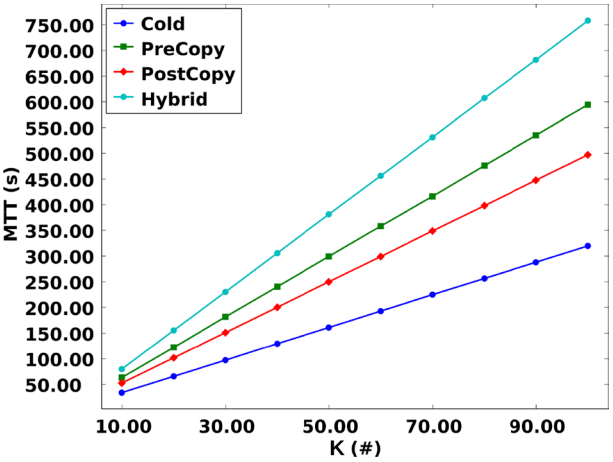


Fig. 12 MTT depending on the number of containers to be migrated (K)

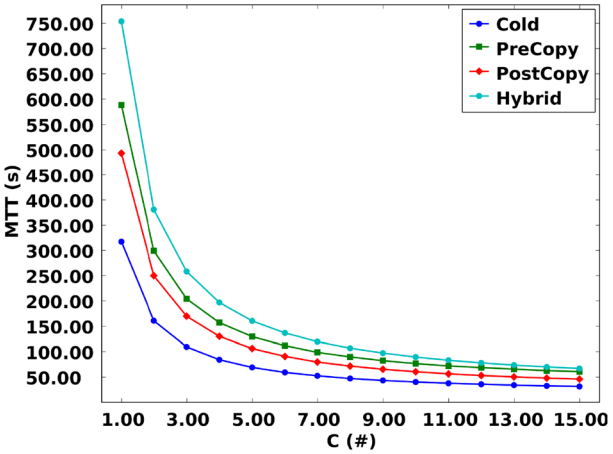


Fig. 13 MTT depending on the capacity for parallel migrations (C)

Effect of parallel migration capacity on MTT

Figure 13 displays the MTT as a function of the parallel migration capacity (C), tested over a range from 1 to 15. The analysis used a constant of 50 elements ($K = 50$) for migration. Initially, all four policies started with high MTT values, especially the hybrid policy, which had the worst initial performance. As C increased, MTT values decreased and eventually stabilized for values of $C \geq 13$, showing minimal improvement beyond $C = 3$. Beyond a certain point, increasing C does not significantly enhance performance, underscoring the importance of optimizing resource allocation by identifying an effective C threshold.

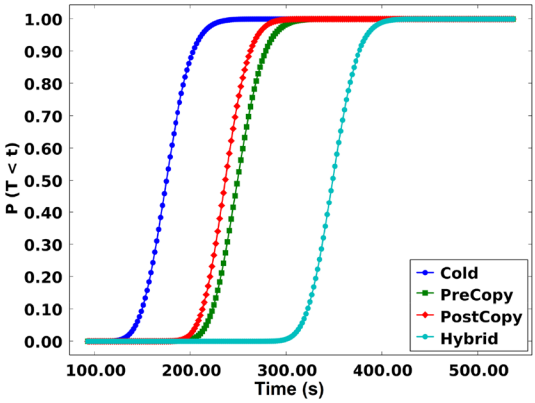


Fig. 14 CDF considering the migration total time

Probability analysis using CDF

The model's capability to calculate the CDF allows for estimating the probability of completing the migration within specific time windows. Figure 14 uses a scenario with 50 elements (K = 50) and a parallel migration capacity of two (C = 2) to show how likely each policy is to complete the migration within various time frames. For instance, at MTT = 250 s, the probability of completing migration is 100% for Cold, 52% for PreCopy, 76% for PostCopy, and 0% for Hybrid. This differentiation is crucial for service level agreements where meeting specific time criteria is essential. For example, the cold policy guarantees completion within 250 s, making it the safest choice under stringent time constraints. In contrast, PostCopy offers a 76% chance of meeting the same deadline, posing a higher risk. Overall, these case studies illustrate the model's utility in predicting system behavior under different configurations and help administrators make informed decisions about resource allocation and policy selection to optimize container migration processes.

6.2.1 Sensitivity analysis using DoE

This section outlines a sensitivity analysis conducted using the DoE method [14, 62–66]. The primary goal of this analysis was to determine which components significantly affect the MTT and thereby deepen our understanding of the container migration process. Through DoE, we could pinpoint the critical factors impacting migration.

Table 6 details the factors and their respective levels, which were manipulated during the DoE. Each factor's level was adjusted by either increasing or decreasing it by 50% from its baseline value. The outcomes of these manipulations across 32 different experimental runs are documented in Table 7.

Figure 15 depicts the individual and interactive effects of these factors on MTT. Among the tested factors, the number of elements to be migrated (K) showed the most substantial influence on MTT, followed by the parallel migration capacity (C) and their interaction (C and K). Notably, the interaction between

Table 6 Table of factors and levels for hybrid migration policy

Factors	Low setting	High setting
K	5,0	15,0
C	1,0	3,0
Dump4	2,74(s)	8,221(s)
Copy4	2,033(s)	6,101(s)
Conect4	2,055(s)	6,166(s)

Table 7 Combinations for hybrid migration policy

K	C	Dump4(s)	Copy4(s)	Conect4(s)	MTT(s)
5,00	1,00	2,74	2,03	2,05	41,14
5,00	1,00	2,74	2,03	6,16	61,72
5,00	1,00	2,74	6,10	2,05	61,47
5,00	1,00	2,74	6,10	6,16	82,02
5,00	1,00	8,22	2,03	2,05	68,58
5,00	1,00	8,22	2,03	6,16	89,11
5,00	1,00	8,22	6,10	2,05	88,91
5,00	1,00	8,22	6,10	6,16	109,47
5,00	3,00	2,74	2,03	2,05	17,62
5,00	3,00	2,74	2,03	6,16	26,96
5,00	3,00	2,74	6,10	2,05	26,83
5,00	3,00	2,74	6,10	6,16	35,62
5,00	3,00	8,22	2,03	2,05	30,49
5,00	3,00	8,22	2,03	6,16	39,03
5,00	3,00	8,22	6,10	2,05	38,93
5,00	3,00	8,22	6,10	6,16	47,36
15,00	1,00	2,74	2,03	2,05	123,46
15,00	1,00	2,74	2,03	6,16	185,09
15,00	1,00	2,74	6,10	2,05	184,53
15,00	1,00	2,74	6,10	6,16	246,17
15,00	1,00	8,22	2,03	2,05	205,68
15,00	1,00	8,22	2,03	6,16	267,31
15,00	1,00	8,22	6,10	2,05	266,73
15,00	1,00	8,22	6,10	6,16	328,41
15,00	3,00	2,74	2,03	2,05	45,01
15,00	3,00	2,74	2,03	6,16	68,08
15,00	3,00	2,74	6,10	2,05	67,81
15,00	3,00	2,74	6,10	6,16	90,29
15,00	3,00	8,22	2,03	2,05	76,21
15,00	3,00	8,22	2,03	6,16	98,41
15,00	3,00	8,22	6,10	2,05	98,19
15,00	3,00	8,22	6,10	6,16	120,34

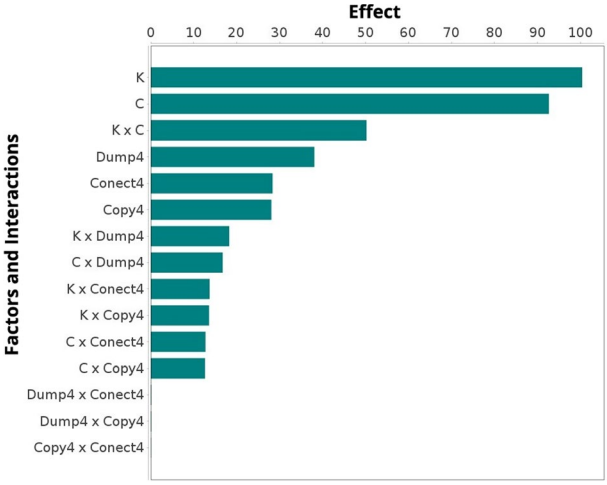


Fig. 15 Factors and interactions and their respective effects on the hybrid migration policy

C and K had a significant effect, demonstrating the importance of these two factors when combined.

Other notable interactions included those involving the K factor and the Dump4 operation and the C factor with Dump4, each showing an approximate 16.0 effect on MTT. Lesser but still relevant effects were observed from the interactions of K with Conect4, C with Conect4, K with Copy4, and C with Copy4, all impacting MTT to a lesser extent, around an approximate value of 12.

The least impactful interactions were between Dump4 and Copy4, Dump4 and Conect4, and Copy4 and Conect4, all of which showed no meaningful effect on MTT. This indicates that any choice among these interactions will not influence the MTT outcome.

These findings are crucial for system architects and developers aiming to enhance the efficiency of container migrations, particularly under the hybrid migration policy. By understanding and prioritizing these critical factors, more effective strategies can be developed to improve MTT in container migration processes.

Interaction between K and C Figure 16a displays the interaction between the number of elements to be migrated (K) and the parallel migration capacity (C). There is a notable interaction between C at levels 1 and 3 concerning K. The analysis suggests that for single migrations, fewer elements should be migrated to minimize MTT, although increasing K only slightly raises MTT when three containers are migrated simultaneously. For configurations where minimizing migration time is a priority, it is recommended to adjust K and C jointly to explore the positive interactive effect. Increasing C reduces MTT up to a saturation point, so identifying this limit is recommended to avoid over-allocating resources without significant performance gains. Information regarding the unit of measure for the x-axis is provided in Table 6.

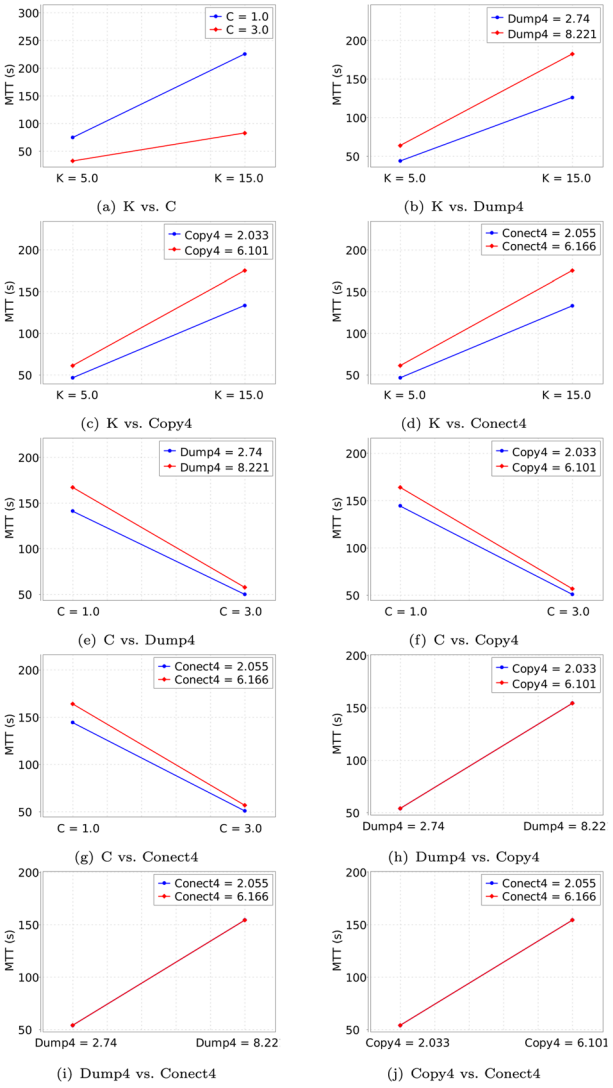


Fig. 16 Sensitivity analysis about the interaction between factors

Interaction between K and Dump4 Figure 16b examines the interaction between K and the time taken by Dump4. The most efficient combination to reduce MTT is achieved with Dump4 set to 2.74 s and K to 5, resulting in a MTT of approximately 44.0 s. However, increasing Dump4 to 8.221 s impacts MTT more significantly, pushing it to about 63.0 s for the same K value. In environments where migration capacity is limited, keeping the Dump4 time short is essential to optimize MTT, especially when K needs to be increased. This configuration helps maintain performance in scenarios with a larger number of containers.

Interaction between K and Copy4 Figure 16c focuses on the interaction between K and the time taken by Copy4. The optimal setting to minimize MTT pairs Copy4 at 2.033 s with K at 5, leading to a MTT of approximately 46.0 s. It is noteworthy that increasing Copy4 to 6.101 s significantly affects MTT when K is increased to 15. In scenarios where K needs to be elevated, it is recommended to keep Copy4 around 2.033 s to minimize the impact on MTT, ensuring efficiency in data copying.

Interaction between K and Conect4 As shown in Fig. 16d, adjusting Conect4 to 2.055 s with K set to 5 optimizes MTT. Altering Conect4 to 6.166 s while keeping K at five does not significantly change the outcome. When the number of containers is small, Conect4 can be adjusted to 2.055 s to minimize MTT, providing flexibility in environments requiring low latency.

Interaction between C and Dump4 Figure 16e illustrates that setting Dump4 to 8.221 s is optimal when C is 3. This finding indicates that a higher parallel migration capacity (C) allows for a longer duration on Dump4 with minimal MTT impact. In high-capacity environments, this Dump4 configuration provides greater stability in the data collection process without compromising the total migration time.

Interaction between C and Copy4 According to Fig. 16f, the lowest MTT results from a combination of Copy4 at 2.033 s and C at 3, which achieves a MTT of approximately 51.0 s. Variations in Copy4 affect the impact on MTT differently depending on the set value of C. This combination is ideal for high-capacity scenarios, ensuring that the copying time is optimized in parallel migration operations.

Interaction between C and Conect4 Figure 16g shows that setting Conect4 to 2.055 s is ideal when C is 1-increasing C to 3 results in negligible differences in MTT, providing flexibility in choosing either setting. This flexibility allows migration capacity adjustments without compromising performance, which is advantageous in dynamic environments where resource utilization may vary.

Interaction between Dump4 and Copy4 Figure 16h indicates no significant interaction between Dump4 and Copy4. The choice of Copy4 setting does not impact MTT regardless of whether C is set to 1 or 3. The interactions between Dump4 and Copy4, as well as between Dump4 and Conect4, had negligible effects on MTT, allowing adjustments to these parameters as needed without impacting performance.

Lack of interaction in other factors Figure 16i and j illustrate no significant interactions between K and Dump4, Copy4, and Conect4. These factors may be less critical in optimizing MTT and can be deprioritized in further evaluations. These findings facilitate a more nuanced understanding of how different factors and their interactions influence the efficiency of container migration, providing valuable insights for system architects to optimize the migration process effectively.

6.3 Case study: non-absorbing model

Variation in arrival rate (AR) Figure 17a provides a more detailed analysis of the impact of AR on MMT for the cold, Pre-Copy, Post-Copy, and hybrid migration policies. When AR is low, around 0.01 messages per second, all policies perform similarly, maintaining stable and low migration times. As AR increases to 0.12 msg/s, the cold policy stands out, maintaining constant migration times, while the

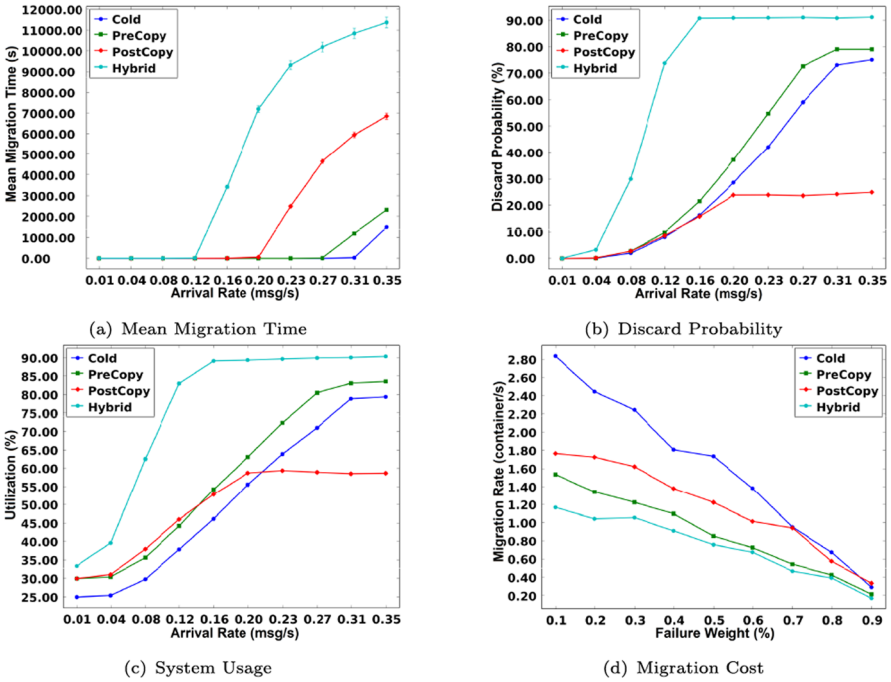


Fig. 17 Comparative analysis of different performance metrics

Pre-Copy and Hybrid policies begin to show significant increases in MMT. When AR reaches 0.35 msg/s, the differences between policies become more pronounced. The hybrid policy, which initially performs well at low ARs, experiences the most significant increase in MMT, surpassing 10,000 s. The increase is directly related to the complexity of hybrid migrations, which involve combining different migration techniques. The Pre-Copy policy, although more stable than the hybrid, also exhibits a significant rise in MMT as AR increases, reflecting the heavy demands that high load places on this method. Overall, the cold policy proves to be the most effective under high-load conditions, making it the ideal solution for network scenarios with heavy traffic.

Discard probability (DP) Figure 17b shows how DP evolves as AR increases. Initially, when AR is very low (around 0.01 messages per second), all policies maintain a DP close to 0%. However, as AR exceeds 0.12 msg/s, the hybrid policy begins to experience drops in the migration process. When AR reaches 0.27 msg/s, the DP for Hybrid increases rapidly, exceeding 80%. At an AR of 0.35 msg/s, Hybrid reaches a DP of 91%, indicating that the complexity of managing both Pre-copy and Post-copy together results in higher packet losses. In contrast, the cold policy maintains a significantly lower DP, even as AR increases. The Pre-Copy and Post-Copy strategies, while showing some increase in DP, manage to keep drop rates well below those of the Hybrid approach, both remaining under 40% throughout the test. These strategies are more suitable for scenarios where packet loss must be minimized, even in high-load environments.

Utilization with Varying AR Figure 17c shows how system utilization changes as AR increases. Utilization starts at 25% for all policies. As AR reaches 0.12 msg/s, the hybrid policy experiences a sharp rise in utilization, quickly reaching around 90%. Suggesting that hybrid strategies make full use of available system resources when handling high loads. However, the high utilization may be accompanied by an increased discard probability, as shown in Fig. 17b. On the other hand, Post-Copy shows a more consistent and efficient use of resources over time. When AR reaches 0.35 msg/s, Post-Copy stabilizes at a utilization rate of around 58.7%. Cold and Pre-Copy policies, although showing slower growth in utilization, maintain a more stable resource use across different AR levels. Such stability is relevant in systems where resource efficiency is prioritized over migration speed.

Impact of failure probability on migration rate Figure 17d illustrates the impact of failure probability on migration rates across different policies. As the failure probability increases from 0.1 to 0.9, the cold policy initially shows the highest migration rate, indicating that it makes better use of available resources when handling low failure rates. However, as the probability of failure rises, the differences between the policies narrow. By the time failure probability reaches 0.7, migration rates across all policies converge to similar levels. At higher failure probabilities, the advantages of individual migration strategies become negligible, and the system's ability to sustain high migration rates decreases significantly. When the failure probability hits 0.9, all policies show similar migration rates, suggesting that under extreme conditions, none of the strategies can outperform the others. In such scenarios, factors like system stability and resilience against failures become more critical.

This case study exemplifies how variations in system inputs-like arrival rate and failure probability-affect the performance metrics of container migration policies, thereby guiding administrators in selecting appropriate policies based on system demands and conditions.

7 Discussion

The findings of this study highlight the effectiveness of stochastic Petri net (SPN) models in simulating and evaluating container migration strategies. By assessing the cold, PreCopy, PostCopy, and hybrid policies, we identified distinct advantages and limitations for each strategy. For example, the cold policy minimizes total migration time in high-load scenarios, while PostCopy excels in maintaining a low discard probability under high demand. These results provide actionable insights for optimizing container migration in dynamic environments.

However, the study faced challenges related to physical infrastructure. The experimental setup was limited to two virtual machines with constrained resources, which restricted the scale and complexity of scenarios that could be evaluated. This limitation underscores the importance of scalable infrastructure to validate the findings across diverse real-world conditions. Future studies should incorporate larger and more varied testbeds to generalize these results.

Furthermore, the study carefully considered threats to validity. Internal validity was ensured by rigorous experimental design, minimizing biases in measurement

and execution. External validity was addressed by aligning our SPN models with realistic operational parameters, though the scalability of our findings remains limited by the experimental setup. Construct validity was strengthened by employing metrics like migration total time (MTT) and discard probability that are directly relevant to container migration performance. Conclusion validity was supported by statistical tests, confirming the reliability of our observations.

The implications of this research extend to practical applications. IT infrastructure managers can leverage these findings to refine migration strategies, balancing efficiency and reliability. Nonetheless, the inherent limitations in infrastructure capacity during the study highlight the necessity for investments in robust physical setups to support more comprehensive evaluations and implementations in diverse environments.

8 Conclusion

This paper presented two comprehensive container migration models designed to assist computing infrastructure administrators in selecting optimal migration policies based on critical factors such as the number of elements to be migrated, the system's parallel migration capacity, probability of failure, and arrival rate. Our absorbing model demonstrates how the CDF and MTT are affected, with findings showing that the cold policy typically results in the lowest MTT, particularly as the number of migrated elements increases. In contrast, the hybrid policy exhibits the highest MTT. Increasing the parallel migration capacity reduces MTT but shows diminishing returns on performance gains at capacities greater than or equal to 13. The non-absorbing model focuses on MMT, DP, utilization, and migration rate, highlighting that the cold policy performs best in high arrival rate scenarios. In contrast, the PostCopy policy maintains the lowest discard probability, making it ideal in high-demand situations. For future work, we will conduct experiments in distributed environments with multiple VMs and nodes, with the aim of evaluating performance in large-scale scenarios. In addition, we will address aspects of concurrency and contention in the modeling, to understand how these factors impact performance. This expansion will allow for a more comprehensive validation and will contribute to the generalization of the results of this study. We will also explore additional migration policies and assess their performance on different infrastructure types such as private, public, and hybrid clouds, evaluate the impact of network latency on migration times and DP, and consider the implications of large-scale migrations and container size on migration processes. This comprehensive study not only advances the theoretical understanding of container migration but also provides actionable guidelines for system administrators to optimize migration strategies effectively.

Author contributions The authors contributed equally to this work.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Turnbull J (2014) The Docker book: containerization is the new virtualization. James Turnbull, San Francisco, CA. https://doi.org/10.21275/sr24810090806
2. Statista (2023) Container technology—statistics and facts. https://doi.org/10.5040/9781509957361.ch-005
3. Varasteh A, Goudarzi M (2015) Server consolidation techniques in virtualized data centers: a survey. IEEE Syst J 11(2):772–783. https://doi.org/10.1109/jsyst.2015.2458273
4. Conforti L, Virdis A, Puliafito C, Mingozzi E (2021) Extending the quic protocol to support live container migration at the edge. In: 2021 IEEE 22nd international symposium on a world of wireless, mobile and multimedia networks (WoWMoM). IEEE, pp 61–70. https://doi.org/10.1109/wowmom51794.2021.00019
5. Junior PS, Miorandi D, Pierre G (2020) Stateful container migration in geo-distributed environments. In: 2020 IEEE international conference on cloud computing technology and science (CloudCom). IEEE, pp 49–56. https://doi.org/10.1109/cloudcom49646.2020.00005
6. Authors TK (2021) Kubernetes documentation. Acessado em: 2 de novembro de 2024. https://kubernetes.io/docs/home/
7. Vaughan-Nichols S (2022) Migrating from VMware to OpenStack: optimizing your infrastructure to save money and avoid vendor-lock-in. Acessado em: 2 de novembro de 2024. https://www.openstack.org/vmware-migration-to-openstack-white-paper/
8. Burns B, Beda J, Hightower K, Evenson L (2022) Kubernetes: up and running. O'Reilly Media, Inc., Sebastopol, CA. https://doi.org/10.1002/9781119814795.ch11
9. Kotikalapudi SVN (2017) Comparing live migration between linux containers and kernel virtual machine: investigation study in terms of parameters. https://doi.org/10.21275/v5i3.nov161797
10. Torre R, Urbano E, Salah H, Nguyen GT, Fitzek FH (2019) Towards a better understanding of live migration performance with docker containers. In: European wireless 2019; 25th European wireless conference. VDE, pp 1–6. https://doi.org/10.2991/icmii-15.2015.106
11. Pickartz S, Eiling N, Lankes S, Razik L, Monti A (2016) Migrating linux containers using criu. In: High performance computing: ISC high performance 2016 international workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P$^{\hat{}}$ 3MA, VHPC, WOPSSS, Frankfurt, Germany, June 19–23, 2016, Revised Selected Papers 31. Springer, pp 674–684. https://doi.org/10.1007/978-3-319-46079-6_47
12. Malhotra M, Trivedi KS (1995) Dependability modeling using petri-nets. IEEE Trans Reliab 44(3):428–440. https://doi.org/10.1109/24.406578
13. Ciardo G, Muppala JK, Trivedi KS et al (1989) SPNP: stochastic petri net package. In: PNPM, vol 89, Citeseer, pp 142–151. https://doi.org/10.1201/9781003210160-47
14. Carvalho D, Rodrigues L, Endo PT, Kosta S, Silva FA (2020) Mobile edge computing performance evaluation using stochastic petri nets. In: 2020 IEEE symposium on computers and communications (ISCC). IEEE, pp 1–6. https://doi.org/10.1109/ISCC50000.2020.9219650
15. Requeno J-I, Merseguer J, Bernardi S (2017) Performance analysis of Apache storm applications using stochastic petri nets. In: 2017 IEEE international conference on information reuse and integration (IRI). IEEE, pp 411–418. https://doi.org/10.1109/IRI.2017.64
16. Silva FA, Fé I, Gonçalves G (2021) Stochastic models for performance and cost analysis of a hybrid cloud and fog architecture. J Supercomput 77(2):1537–1561. https://doi.org/10.1007/s11227-020-03310-1
17. Puliafito C, Mingozzi E, Anastasi G (2017) Fog computing for the internet of mobile things: issues and challenges. In: 2017 IEEE international conference on smart computing (SMARTCOMP). IEEE, pp 1–6. https://doi.org/10.1109/smartcomp.2017.7947010
18. Jiang Y, Huang Z, Tsang DH (2017) Challenges and solutions in fog computing orchestration. IEEE Netw 32(3):122–129. https://doi.org/10.1109/mnet.2017.1700271

19. Zhu C, Tao J, Pastor G, Xiao Y, Ji Y, Zhou Q, Li Y, Ylä-Jääski A (2018) Folo: latency and quality optimized task allocation in vehicular fog computing. IEEE Internet Things J 6(3):4150–4161. https://doi.org/10.1109/jiot.2018.2875520
20. Fernández-Caramés TM, Fraga-Lamas P, Suárez-Albela M, Vilar-Montesinos M (2018) A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard. Sensors 18(6):1798. https://doi.org/10.3390/s18061798
21. Deshpande L, Liu K (2017) Edge computing embedded platform with container migration. In: 2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, pp 1–6. https://doi.org/10.1109/uic-atc.2017.8397578
22. Sindi M, Williams JR (2019) Using container migration for HPC workloads resilience. In: 2019 IEEE high performance extreme computing conference (HPEC). IEEE, pp 1–10. https://doi.org/10.1109/hpec.2019.8916436
23. Nie H, Li P, Xu H, Dong L, Song J, Wang R (2017) Research on optimized pre-copy algorithm of live container migration in cloud environment. In: Parallel architecture, algorithm and programming: 8th international symposium, PAAP 2017, Haikou, China, June 17–18, 2017, Proceedings 8. Springer, pp 554–565. https://doi.org/10.1007/978-981-10-6442-5_53
24. Puliafito C, Vallati C, Mingozzi E, Merlino G, Longo F, Puliafito A (2019) Container migration in the fog: a performance evaluation. Sensors 19(7):1488. https://doi.org/10.3390/s19071488
25. Puliafito C, Vallati C, Mingozzi E, Merlino G, Longo F (2021) Design and evaluation of a fog platform supporting device mobility through container migration. Pervasive Mob Comput 74:101415. https://doi.org/10.1016/j.pmcj.2021.101415
26. Tošić A (2023) Run-time application migration using checkpoint/restore in userspace. Preprint at arXiv:2307.12113. https://doi.org/10.13052/jwe1540-9589.2357
27. Pires A, Simão J, Veiga L (2021) Distributed and decentralized orchestration of containers on edge clouds. J Grid Comput 19:1–20. https://doi.org/10.1007/s10723-021-09575-x
28. Tošić A, Vičič J, Burnard M, Mrissa M (2023) A blockchain protocol for real-time application migration on the edge. Sensors 23(9):4448. https://doi.org/10.3390/s23094448
29. Stoyanov R, Kollingbaum MJ (2018) Efficient live migration of Linux containers. In: High performance computing: ISC high performance 2018 international workshops, Frankfurt/Main, Germany, June 28, 2018, Revised Selected Papers 33. Springer, pp 184–193. https://doi.org/10.1007/978-3-030-02465-9_13
30. Ma L, Yi S, Carter N, Li Q (2018) Efficient live migration of edge services leveraging container layered storage. IEEE Trans Mob Comput 18(9):2020–2033. https://doi.org/10.1109/tmc.2018.2871842
31. Di Z, Shao E, Tan G (2021) High-performance migration tool for live container in a workflow. Int J Parallel Prog 49:658–670. https://doi.org/10.1007/s10766-021-00697-z
32. Tay Y, Gaurav K, Karkun P (2017) A performance comparison of containers and virtual machines in workload migration context. In: 2017 IEEE 37th international conference on distributed computing systems workshops (ICDCSW). IEEE, pp 61–66. https://doi.org/10.1109/icdcsw.2017.44
33. Xu B, Wu S, Xiao J, Jin H, Zhang Y, Shi G, Lin T, Rao J, Yi L, Jiang J (2020) Sledge: towards efficient live migration of docker containers. In: 2020 IEEE 13th international conference on cloud computing (CLOUD). IEEE, pp 321–328. https://doi.org/10.1109/cloud49709.2020.00052
34. González AE, Arzuaga E (2020) Herdmonitor: monitoring live migrating containers in cloud environments. In: 2020 IEEE international conference on big Data (Big Data). IEEE, pp 2180–2189. https://doi.org/10.1109/bigdata50022.2020.9378473
35. Govindaraj K, Artemenko A (2018) Container live migration for latency critical industrial applications on edge computing. In: 2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA), vol 1. IEEE, pp 83–90. https://doi.org/10.1109/etfa.2018.8502659
36. Benjaponpitak T, Karakate M, Sripanidkulchai K (2020) Enabling live migration of containerized applications across clouds. In: IEEE INFOCOM 2020-IEEE conference on computer communications. IEEE, pp 2529–2538. https://doi.org/10.1109/infocom41043.2020.9155403
37. Abdullah DB, Hadeed W et al (2022) Container live migration in edge computing: a real-time performance amelioration. Int J Appl Sci Eng 19(3):1–8. https://doi.org/10.1007/978-3-319-91337-7_19

38. Ramanathan S, Kondepu K, Razo M, Tacca M, Valcarenghi L, Fumagalli A (2021) Live migration of virtual machine and container based mobile core network components: a comprehensive study. IEEE Access 9:105082–105100. https://doi.org/10.1109/access.2021.3099370
39. Fan W, Han Z, Li P, Zhou J, Fan J, Wang R (2019) A live migration algorithm for containers based on resource locality. J Signal Process Syst 91:1077–1089. https://doi.org/10.1007/s11265-018-1401-8
40. Machen A, Wang S, Leung KK, Ko BJ, Salonidis T (2017) Live service migration in mobile edge clouds. IEEE Wirel Commun 25(1):140–147. https://doi.org/10.1109/mwc.2017.1700011
41. Pecholt J, Huber M, Wessel S (2021) Live migration of operating system containers in encrypted virtual machines. In: Proceedings of the 2021 on cloud computing security workshop, pp 125–137. https://doi.org/10.1145/3474123.3486761
42. Smimite O, Afdel K (2019) Impact of hybrid virtualization using VM and container on live migration and cloud performance. In: Lecture notes in real-time intelligent systems. . Springer, pp 196–208. https://doi.org/10.1007/978-3-319-91337-7_19
43. Al-Dhuraibi Y, Paraiso F, Djarallah N, Merle P (2017) Autonomic vertical elasticity of docker containers with elasticdocker. In: 2017 IEEE 10th international conference on cloud computing (CLOUD). IEEE pp 472–479. https://doi.org/10.1109/cloud.2017.67
44. Chou CC, Chen Y, Milojicic D, Reddy N, Gratz P (2019) Optimizing post-copy live migration with system-level checkpoint using fabric-attached memory. In: 2019 IEEE/ACM workshop on memory centric high performance computing (MCHPC). IEEE, pp. 16–24. https://doi.org/10.1109/mchpc49590.2019.00010
45. Das R, Sidhanta S (2023) Live migration of containers in the edge. SN Comput Sci 4(5):479. https://doi.org/10.5220/0011060800003200
46. Majeed AA, Kilpatrick P, Spence I, Varghese B (2020) Modelling fog offloading performance. In: 2020 IEEE 4th international conference on fog and edge computing (ICFEC). IEEE, pp 29–38. https://doi.org/10.1109/icfec50348.2020.00011
47. Karhula P, Janak J, Schulzrinne H (2019) Checkpointing and migration of IoT edge functions. In: Proceedings of the 2nd international workshop on edge systems, analytics and networking, pp 60–65. https://doi.org/10.1145/3301418.3313947
48. Bhardwaj A, Rama Krishna C (2022) A container-based technique to improve virtual machine migration in cloud computing. IETE J Res 68(1):401–416. https://doi.org/10.1080/03772063.2019.1605848
49. Ramanathan S, Kondepu K, Zhang T, Mirkhanzadeh B, Razo M, Tacca M, Valcarenghi L, Fumagalli A (2021) A comprehensive study of virtual machine and container based core network components migration in openroadm SDN-enabled network. Preprint at arXiv:2108.12509. https://doi.org/10.23919/ondm51796.2021.9492500
50. Kakakhel SRU, Mukkala L, Westerlund T, Plosila J (2018) Virtualization at the network edge: a technology perspective. In: 2018 third international conference on fog and mobile edge computing (FMEC). IEEE, pp 87–92. https://doi.org/10.1109/fmec.2018.8364049
51. Baccarelli E, Scarpiniti M, Momenzadeh A (2018) Fog-supported delay-constrained energy-saving live migration of VMS over multipath TCP/IP 5G connections. IEEE Access 6:42327–42354. https://doi.org/10.1109/access.2018.2860249
52. Maciel PRM (2023) Performance, reliability, and availability evaluation of computational systems, volume I: performance and background. Chapman and Hall/CRC
53. Marsan MA, Balbo G, Conte G, Donatelli S, Franceschinis G (1998) Modelling with generalized stochastic petri nets. ACM SIGMETRICS Perform Eval Rev 26(2):2. https://doi.org/10.1007/978-3-663-11521-2_8
54. Nelson R (2013) Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling. Springer, New York, NY. https://doi.org/10.2307/2669886
55. Pinheiro T, Silva FA, Fe I, Kosta S, Maciel P (2018) Performance and data traffic analysis of mobile cloud environments. In: 2018 IEEE international conference on systems, man, and cybernetics (SMC). IEEE, pp 4100–4105. https://doi.org/10.1109/smc.2018.00695
56. Bolch G, Greiner S, De Meer H, Trivedi KS (2006) Queueing networks and Markov Chains: modeling and performance evaluation with computer science applications, 2nd edn. John Wiley & Sons, Hoboken, NJ. https://doi.org/10.1002/0471791571.ch2
57. German R (2000) Performance analysis of communication systems with non-Markovian stochastic Petri nets. John Wiley & Sons Inc, Hoboken, NJ. https://doi.org/10.1201/9781482274530-29

58. Jain R (1990) The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling. John Wiley & Sons, Hoboken, NJ. https://doi.org/10.1093/comjnl/35.3.207
59. Dayo AO (2021) A multi-containerized application using docker containers and kubernetes clusters. Int J Comput Appl 183(44):55–60. https://doi.org/10.5120/ijca2021921843
60. Koltai M, Noel V, Zinovyev A, Calzone L, Barillot E (2019) Exact calculation of stationary solution and parameter sensitivity analysis of stochastic continuous time boolean models. bioRxiv, 794230
61. Maciel P, Matos R, Silva B, Figueiredo J, Oliveira D, Fé I, Maciel R, Dantas J (2017) Mercury: performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In: 2017 IEEE 22nd Pacific rim international symposium on dependable computing (PRDC). IEEE, pp 50–57. https://doi.org/10.1109/prdc.2017.16
62. Feitosa L, Gonçalves G, Nguyen TA, Lee JW, Silva FA (2021) Performance evaluation of message routing strategies in the internet of robotic things using the d/m/c/k/fcfs queuing network. Electronics 10(21):2626. https://doi.org/10.3390/electronics10212626
63. Gonçalves I, Rodrigues L, Silva FA, Nguyen TA, Min D, Lee J-W (2021) Surveillance system in smart cities: a dependability evaluation based on stochastic models. Electronics 10(8):876. https://doi.org/10.3390/electronics10080876
64. Santos L, Cunha B, Fé I, Vieira M, Silva FA (2021) Data processing on edge and cloud: a performability evaluation and sensitivity analysis. J Netw Syst Manag 29(3):1–24. https://doi.org/10.1007/s10922-021-09592-x
65. Santos B, Soares A, Nguyen T-A, Min D-K, Lee J-W, Silva F-A (2021) IoT sensor networks in smart buildings: a performance assessment using queuing models. Sensors 21(16):5660. https://doi.org/10.3390/s21165660
66. Victor C, Nguyen TA, Silva LA, Andrade E, Santos GL, Min D, Lee JW, Silva FA (2021) Performability assessment and sensitivity analysis of a home automation system. In: 2021 IEEE/ACM 25th international symposium on distributed simulation and real time applications (DS-RT). IEEE, pp 1–4. https://doi.org/10.1109/ds-rt52167.2021.9576142

Authors and Affiliations

Leonel Feitosa[1] · Vandirleya Barbosa[1] · Arthur Sabino[1] · Luiz Nelson Lima[1] · Iure Fé[1] · Luis Guilherme Silva[1] · Gustavo Callou[2] · Juliana Carvalho[1] · Erico Leão[1] · Tuan Anh Nguyen[3] · Paulo Rego[4] · Francisco Airton Silva[1]

✉ Francisco Airton Silva
faps@ufpi.edu.br

Leonel Feitosa
leonelfeitosa@ufpi.edu.br

Vandirleya Barbosa
vandirleya.barbosa@ufpi.edu.br

Arthur Sabino
arthursabino@ufpi.edu.br

Luiz Nelson Lima
luizznelson@ufpi.edu.br

Iure Fé
iure.fe@ufpi.edu.br

Luis Guilherme Silva
luis.e@ufpi.edu.br

Gustavo Callou
gustavo.callou@ufrpe.br

Juliana Carvalho
julianaoc@ufpi.edu.br

Erico Leão
ericoleao@ufpi.edu.br

Tuan Anh Nguyen
anhngt@huit.edu.vn

Paulo Rego
paulo@dc.ufc.br

[1] Laboratory of Applied Research to Distributed Systems (PASID), Federal University of Piauí (UFPI), Picos, Piauí 64607-670, Brazil

[2] Department of Computing, Federal Rural University of Pernambuco (UFRPE), Recife, Pernambuco, Brazil

[3] Department of Software Engineering, Faculty of Information Technology, University of Industry and Trade (HUIT), Ho Chi Minh City 700000, Vietnam

[4] Department of Computer Science, Federal University of Ceará (UFC), Fortaleza, Ceará, Brazil