A Literature Review on Live Migration in Container-Based Distributed Systems

João Pedro Resende Barroso[1], **Aleardo Manacero**[1], **Renata Spolon Lobato**[1], **Roberta Spolon**[2]

[1] São Paulo State University - UNESP – São José do Rio Preto, SP

[2]São Paulo State University - UNESP – Bauru, SP

joao.barroso@unesp.br, aleardo.manacero@unesp.br

renata.spolon@unesp.br, roberta.spolon@unesp.br

Abstract. *Task schedulers are a cornerstone of high-performance distributed computing systems, efficiently distributing workloads across multiple nodes to optimize resource utilization and enhance application performance. By balancing computational tasks, they ensure scalability, minimize latency, and maintain system reliability. Live container migration is an emerging technology in the area of task schedulers; it involves making the task of rescheduling containers as fast as possible, thereby reducing the idle time of the system. This article reviews the most relevant works in the area of live migration in container-based distributed systems, identifying the most significant innovations on the subject.*

1. Introduction

Distributed systems are fundamental for executing tasks of medium to high complexity, particularly in scenarios involving large-scale data processing, high computational demands, or the need for high availability and fault tolerance. Instead of relying on a single powerful machine, these systems distribute the workload across multiple nodes, enabling parallelism, scalability, and improved performance. To ensure coordination among the nodes, distributed systems typically utilize a component known as middleware.

The scheduler module is a vital component in container-based distributed systems, managing the allocation of tasks and resources across nodes. Its efficiency directly impacts cluster performance, influencing resource utilization, workload distribution, scalability, and overall system responsiveness.

A part of the scheduler is the rescheduling of tasks and resources, which balances the workload on the system depending on the requirements of the tasks during runtime and the situation of each machine. Usually, in container orchestration systems like Kubernetes, the job of migrating tasks, or rescheduling tasks, consists of deleting the running container and then running it on another node with the previous data, which can be slow and cumbersome.

On this matter, there is an innovation called live migration, which is a concept of saving the actual status of a running container and running it on another machine, without the container needing to restart, allowing for an almost instant migration, accelerating the tasks on the cluster. It is already applied in the context of virtual machines, but in the case

of containers, there are challenges, because of the interconnection between the container runtime and the host machine.

Therefore, this article presents a literature review on live container migration, aiming to provide an understanding of the current state of the method and the prevailing research trends in the field.

1.1. Organization of the paper

This article is organized as follows:

The next chapter explores the area of live container migration, emphasizing its relevance to current applications. Following that, the review methodology is detailed, including the research questions, search strategy, and article selection criteria.

Subsequently, a set of guiding questions is introduced, along with the rationale behind each one. The results chapter then presents insights gathered from the literature review. Finally, the article concludes with a brief summary of the findings and discusses potential future directions in the field of live container migration.

2. Context

Today, high-end applications, such as machine learning and real-time simulations, work on tens of thousands of terabytes of data at runtime [32], which means that data processing must also be very fast. Therefore, the use of parallelization in distributed systems is necessary.

The speedup that parallelization brings to an application depends on the system's capacity to execute operations simultaneously. Once the system does not have sufficient resources for parallel execution, there will be a bottleneck. Therefore, understanding the workload of an application is of utmost importance, allowing the system manager to divide the resources appropriately and achieve greater performance.

Based on the understanding of an application's workload, developers can design specialized schedulers tailored to specific requirements[24]. This enables tasks to be distributed more effectively, potentially achieving greater speedup compared to scenarios where the distribution is performed manually or using a generic approach. In certain cases, performance gains can be even more significant if the system is able to adapt to tasks dynamically during their execution.

To achieve this, a rescheduling module can be employed, leveraging runtime information to continuously balance resources and ensure performance improvements, particularly in situations where workloads become concentrated on specific nodes [6].

In order to reduce the time required to rebalance tasks across the nodes of a system, the concept of live migration may be applied. As shown in Figure 1, this technique involves the action of checkpointing a container, capturing all the execution state of a task (including memory data, CPU state, etc.) and restoring it to another node, thereby allowing execution to continue seamlessly.

In the context of distributed systems based on virtual machines, this process is facilitated by the fact that a VM is encapsulated within a single process, with all system and

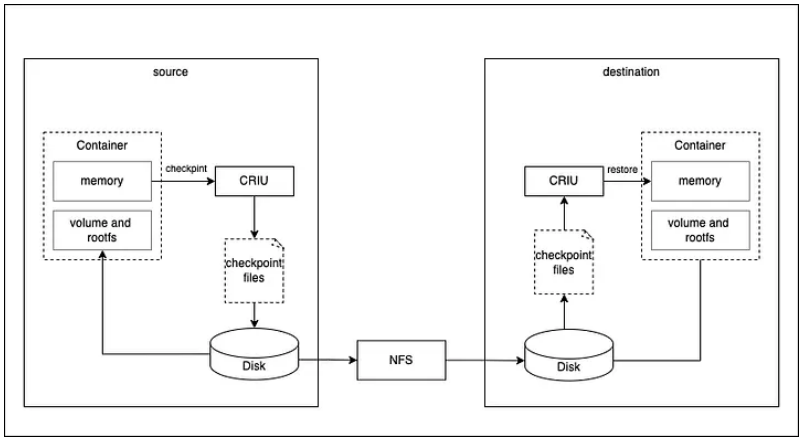

Figure 1. Example of checkpoint/restore of a container. Source: https://surenraju.medium.com/migrate-running-containers-by-checkpoint-restoring-using-criu-6670dd26a822

hardware information isolated from the host environment[9]. Conversely, in container-based distributed systems, where each container shares the host's kernel, migration poses greater challenges, as containers rely on host-level resources and data.

2.1. Related Works

Surveys on live container migration often concentrate on specific contexts. For example, Soussi, Gür, and Stiller focuses on designing live migration modules and comparing them within cloud computing environments. Similarly, Patel, Mehta, and Patel provides a brief survey of migration in edge and hybrid computing, emphasizing application-specific use cases.

Other surveys, such as Soma and Rukmini and Lohumi et al., examine live migration mainly in the context of virtual machines with the use of containers, which introduces distinct challenges and strategies.

By contrast, this article distinguishes itself by reviewing newer implementations across broader domains and highlighting emerging trends in live container migration, with the aim of identifying areas where live migration is advancing and exploring how this method can benefit other domains.

3. Methodology

This review adopts the methodology proposed by Barbara Kitchenham [13] due to its rigor in guiding the creation of literature reviews. It examines projects developed over the last five years in the field of live migration for container-based distributed systems, with a focus on implementations targeting container orchestrators such as Kubernetes.

The review focuses on projects published in leading journals and conferences indexed by ACM, IEEE, and Scopus. The selection of these articles was based on the relevance of the projects and the novelty of the technologies developed by the researchers. The methodology employed in this study is outlined below.

3.1. Research Questions

The main question this article attempts to answer is:

What is the state of research on live migration in container-based distributed systems?

Therefore, the purpose of this article is to investigate the implementations of live migration in container orchestrators, conducting an in-depth analysis of their innovations. The results highlight the main challenges that the adoption of such a method may pose to system administrators in the industry, as well as the potential performance gains that can be achieved.

3.2. Search Strategy

The first step in the review consists of the choice of keywords that retrieve related articles to the theme. Accordingly, the following expressions were used:

```
[[Abstract: live migration] AND [Abstract: container]] OR [[All: checkpoint restore] AND [Abstract: container]]
```

These expressions were first searched within the abstracts of the articles to better filter out unrelated projects, since the term *container* may have different meanings in the text of a project, which could make the selection of relevant articles more difficult. With this search, the following sum of articles was returned:

- ACM, 19 articles
- IEEE, 42 articles
- Scopus, 40 articles

The search was conducted using the database's search engine, with a restriction to articles published within the last five years. This limitation aims to capture the current state of research, identifying areas that require further attention and development.

3.3. Criteria of Inclusion and Exclusion

To achieve the best outcomes, it's essential to filter works based on specific criteria to ensure only those relevant to this article's purpose are selected. Below are the criteria applied:

Criteria	ID	Description
Inclusion	I1	The article presents a implementation of live container migration.
Exclusion	E1	The article is a performance analysis of an implementation.
	E2	The article is a literature review.
	E3	The article focuses on live virtual machine migration.
	E4	The article does not have a DOI.

Table 1. Criteria of exclusion and inclusion.

3.3.1. Reason for the criteria

- **I1,E1**: To be included, an article must present a novel implementation of live migration, rather than merely analyze an existing project, as the latter is not within the scope of this article.

- **E2**: Articles that focus solely on literature reviews are excluded, as they do not contribute to the development of live container migration modules.
- **E3**: Projects that use a live container migration module but do not focus on it are excluded, as this is not the objective of the article.
- **E4**: Only articles published with a DOI are included, as this facilitates searching for them and ensures access to the original work.

3.4. Filtering the articles

The initial search done in July 2025 returned 101 articles, across all databases. The number of articles was still too big, making it necessary to use further criteria to select the most relevant projects. With the full reading of the abstract, 61 articles were removed, and with the other articles read, only 33 articles were used for the review. For the sake of not making this article bigger, articles with no citations in three years were removed, removing seven articles and with a final number of 26 articles used in this review.

4. Data Extraction

To achieve a clearer understanding of the current state of research in the field of live container migration, a set of questions was formulated to address the characteristics present in each project that may influence the performance and usability of the implementations. Upon completion of the review of these studies, these questions can be answered.

- **Q1: Is the project open source?**
  The availability of the source code of the proposed implementations allows it to be reviewed by third-party researchers and compared with other implementations, which accelerates the development of improved tools.
- **Q2: Usage of simulators for testing?**
  The use of distributed systems simulators allows third-party researchers to easily replicate the results of the project and facilitates its review. It also creates opportunities for further testing on real distributed systems, enabling researchers to validate findings under practical conditions, evaluate system performance at scale, and explore potential optimizations or limitations that may not be evident in simulated environments.
- **Q3: Utilized CRIU in the implementation?**
  CRIU (Checkpoint/Restore in Userspace) is a Linux tool for saving and restoring the state of processes and applications at runtime. It can be used for live container migration, although it is still in an experimental phase for this use case. The use of an established tool allows projects to be more easily extended and improved, as researchers can build on a reliable foundation, reproduce experiments more accurately, and focus on optimizing migration strategies rather than developing the underlying checkpoint/restore mechanisms from scratch.
- **Q4: What language was used for the implementation?**
  The choice of programming language can significantly affect the performance of the module. Languages such as Go and C, which are used by container orchestrators like Kubernetes, have a small memory footprint and execute functions quickly. In contrast, implementations in Python, a high-level language, may exhibit lower performance, presenting opportunities for further optimization and improvement.

- **Q5: On which application domain does the implementation focus?**
  Understanding the primary application domain of the live migration module is important, as different domains have distinct requirements and challenges. For instance, some implementations may target real-time simulations, where minimizing downtime is critical to maintaining simulation accuracy; others may focus on machine learning workloads, where resource elasticity and fast migration between nodes improve training efficiency; and some may address high-performance computing or cloud services, where scalability, fault tolerance, and overall system throughput are the main concerns. Identifying the domain helps contextualize the design choices and performance trade-offs of each implementation.
- **Q6: Concentrates on stateful or stateless applications?**
  The application type used for testing the live container migration module is highly significant. Stateless applications are much easier to migrate at runtime, as only the running processes need to be saved and restored with minimal preparation. In contrast, stateful applications are more challenging to migrate, since the runtime data must also be saved and accurately restored on another machine, often requiring additional coordination and synchronization.

5. Results

This section presents the results obtained from the review of the selected articles, addressing the questions outlined in the previous chapters and providing a comprehensive understanding of the current state of the art.

5.1. Q1: Is the project open source?

During the analysis of the articles, it was observed that most of them do not have their projects open-sourced. From the 26 articles, only 5 of them have a GitHub page, namely KubeSPT [35], UMS [7], and LIMOCE[10]. The other 22 articles do not share their sources, but they present their methodology represented by pseudocode, allowing researchers to implement the proposed project in a more challenging manner.

The action of sharing the source code of the proposed project is good practice because it allows further improvement by third-party researchers and provides an easy way to review the results presented in the paper. For a simpler analysis, we considered open source only cases where the researchers provided the link to the project code. There was no further research on the Internet for the project's source code.

5.2. Q2: Usage of simulators for testing?

The majority of the analyzed live container migration projects were tested on real, container-based distributed systems. Only two articles, Singh [29] and Rukmini [27], employed the generic cloud simulator CloudSim, while another two, namely Hashemi [12] and Rukmini [27], used the fog distributed system simulator iFogSim. This indicates a strong preference for experimentation on actual systems rather than simulated environments.

This preference can be attributed to the relative affordability and ease of setting up distributed systems for testing container orchestrators. In contrast, simulation environments, while useful for preliminary studies, are less appealing when evaluating the performance and behavior of live container migration in realistic conditions.

5.3. Q3: Utilized CRIU on the implementation?

Out of the 26 analyzed articles, 15 employed the tool CRIU in their implementations of live container migration modules. In the case of the project by Poggiani et al., the researchers used the CRI-O container runtime for its compatibility with the checkpoint/restore tool. The remaining 11 articles developed their own checkpoint/restore mechanisms.

This distribution can be attributed to the fact that CRIU's support for containers is still in an experimental stage, which may hinder the development of new projects. As a result, many researchers opt for alternative implementations that are better suited to their specific requirements. In the case of the implementation by Koziolek, Burger, and Puthan Peedikayil, a new checkpoint/restore tool was implemented because it could create smaller checkpoints than CRIU, which is ideal in the context of industrial controller systems.

5.4. Q4: What language was used for the implementation?

The majority of the projects utilized the Go language and the YAML scripting language, as they are what the developers of container orchestrators, such as Kubernetes, utilize. The language Go was utilized for the implementation of the modules, as in the case of the article by Zhang et al. Meanwhile, YAML scripts are used for the configuration of the clusters, which serve as the method of communication and configuration in Kubernetes.

The language Python is also used in many of the articles because of its ease of learning and its ability to implement complex projects, as is the case with a live container migration system. The loss of performance of the language can be compensated for by the speed of development that the language provides to researchers. In the case of some articles, such as Benjaponpitak, Karakate, and Sripanidkulchai, which use a python script for managing the transfer of the checkpoints created by the live migration module, the performance difference between languages is dismissible.

On the projects by Singh et al. and Hashemi et al., the Java programming language was used for the implementation of live container modules for use in simulators such as CloudSim and iFogSim, both of which are written in Java. In the articles by Koziolek, Burger, and Puthan Peedikayil, Das and Sidhanta, and Lu and Jiang, the researchers chose to implement the migration modules in the languages C and C++ because of their low-level functions and low memory usage, which are necessary for low resource machines used in their projects.

5.5. Q5: On which application domain does the implementation focus?

The majority of the projects focus on cloud computing, with test applications primarily consisting of relational databases, such as MySQL, deployed within cloud service environments like AWS and Google Cloud. Relational databases are commonly used for testing live container migration because they are stateful applications that maintain critical data and transactional consistency, making them ideal benchmarks for evaluating the reliability and efficiency of migration techniques.

Furthermore, ten of the articles propose live container migration modules for applications at the edge, with six of them focusing on 5G networks. These kinds of applica-

tions take advantage of live migration due to the necessity for an always-online runtime and the demand for low latency in tasks such as remote surgery and traffic management.

Also, the articles presented by Lim and Lee, Addad et al., and Sarrigiannis et al. focus on MEC (Multi-Access Edge Computing), which demands quick migration from different nodes for a seamless connection with clients. Aside from that, Aleyadeh et al. focuses on hybrid computing between the fog and edge, and Manatura et al. focuses on liquid computing, which comprises a seamless experience among cloud, fog, and edge computing.

Finally, the article by Chanikaphon and Salehi focuses on live migration in the context of autonomous computing in self-driving cars, which have a high necessity for very low latency in the communication of the application. Another article by Koziolek, Burger, and Puthan Peedikayil focuses on industrial controllers, which require a quick means of communicating states between nodes and must always be online for the duration of the process.

Article	Q1	Q2	Q3	Q4	Q5	Q6
Addad et al.[2]			X	Go	5G Network	X
Ramanathan et al.[23]			X	Go	5G Network	X
Lim and Lee[16]				Yaml	5G Network	X
Li et al.[15]			X	Go	5G Network	X
Xie et al.[33]				Python	5G Network	X
Bhattacharyya et al.[5]				Python	5G Network	X
Chanikaphon and Salehi[7]	X		X	Python	Autonomous Cmpt.	X
Zhang et al.[36]			X	Go	Cloud Cmpt	X
Lu and Jiang[18]			X	Go	Cloud Cmpt.	X
Zhang et al.[35]	X		X	Go	Cloud Cmpt.	X
Poggiani et al.[22]	X		X	Go	Cloud Cmpt.	X
Guitart[11]			X	Go	Cloud Cmpt.	X
Singh et al.[29]		CloudSim		Java	Cloud Cmpt.	X
Xu et al.[34]			X	Python	Cloud Cmpt.	X
Rukmini and Soma[26]				Python	Cloud Cmpt.	X
Benjaponpitak et al.[4]			X	Python	Cloud Cmpt.	X
Das and Sidhanta[10]	X		X	C/C++	Edge Cmpt.	X
Rong et al.[25]				Go	Edge Cmpt.	X
Sarrigiannis et al.[28]				Python	Edge Cmpt.	X
Abdullah and Hadeed[1]			X	Python	Edge Cmpt.	X
Rukmini et al.[27]		CloudSim, FogSim		Python	Fog Cmpt.	X
Hashemi et al.[12]		FogSim		Java	Fog Cmpt.	X
Chebaane et al.[8]			X	Python	Fog Cmpt.	X
Aleyadeh et al.[3]				Python	Hybrid Cmpt.	X
Koziolek et al.[14]					Industrial Controller	X
Manatura et al.[19]	X		X	Python	Liquid Cmpt.	X

Table 2. Comparison table between all articles. Abbreviations: *Cmpt.* = Computing

5.6. Q6: Concentrates on stateful or stateless applications?

All of the analyzed articles focus on stateful applications, where containers are migrated along with their runtime state. This is essential in domains such as databases, industrial controllers, and edge computing, where preserving the application's data and execution context is critical for consistency and reliability. Unlike stateless applications, stateful workloads pose greater challenges for live migration, since any disruption or data loss during the transfer may compromise the correctness of the system.

6. Discussion

A deeper discussion is necessary for a better understanding of the trends of research in the area. The first trend observed in the analysis of the articles is the importance of live migration in real-life applications, like databases, traffic managers, and IoT systems, where availability is of utmost importance and the possibility of changing the node at runtime for a better one is welcomed.

The languages Go and Python were the most used for the implementation, the first one because of its relation to container orchestrators, like Kubernetes, and its low-level application, which makes the migration module performative, and the second for its easy-to-program characteristics and the capacity of improving the project without syntax barriers.

Unfortunately, most projects are closed-source, only having parts of their implementation shared as pseudo-code, which makes the job of reviewing the results of their tests harder, as well as the task of improving the implementation presented. We believe that making the project code open source allows it to have a bigger relevance in the area, because anyone could assert its claims and improve the project further.

The usage of CRIU on more than half of articles shows that even if its support for containers is still experimental, its implementation of checkpoint/restore is very usable, allowing researchers to use it to facilitate its implementation of live container migration.

Finally, the focus on implementing live container migration for stateful applications is understandable, as stateless programs are comparatively easier to migrate and do not require extensive investigation.

Nevertheless, the selected applications indicate that more complex domains, such as machine learning and real-time simulations, remain largely unexplored, highlighting the need for further research in these areas.

One reason for the lack of articles on live container migration in machine learning is that it is a complex task requiring a distinct set of skills and knowledge often beyond the typical expertise of AI developers. Additionally, distributed machine learning is still emerging, with many tasks currently running on single, powerful machines. As tools mature and algorithms grow more complex, the demand for live migration in ML workflows is likely to increase.

The same applies to real-time simulations, where the need for low latency and the reliance on in-memory state make live container migration significantly more complex. In most cases, implementing such migration is too costly and cumbersome due to the current limitations of experimental tools and the lack of widespread support from developers.

7. Conclusion

This review highlights the main trends in the field of live container migration. The most prominent application domains at this moment are big data and edge/fog computing, while areas such as autonomous systems and industrial applications are beginning to gain attention. At the same time, domains where migration could be highly beneficial, such as machine learning and real-time simulations, remain largely unexplored, primarily due to their complexity and dependence on specialized hardware accelerators like GPUs and TPUs. Looking ahead, the growing reliance on AI-driven applications, combined with the continued maturation of tools such as CRIU, is expected to enable the adoption of live container migration modules in a broader range of domains, as evidenced by the works reviewed.

References

[1] Dhuha Basheer Abdullah and Wael Hadeed. "Container live migration in edge computing: a realtime performance amelioration". In: *International Journal of Applied Science and Engineering* 19.3 (2022). DOI: 10.6703/IJASE.202209_19(3).007.

[2] Rami Akrem Addad et al. "Fast Service Migration in 5G Trends and Scenarios". In: *IEEE Network* 34.2 (2020). DOI: 10.1109/MNET.001.1800289.

[3] Sam Aleyadeh et al. "Optimal Container Migration/Re-Instantiation in Hybrid Computing Environments". In: *IEEE Open Journal of the Communications Society* 3 (2022). DOI: 10.1109/OJCOMS.2022.3140272.

[4] Thad Benjaponpitak, Meatasit Karakate, and Kunwadee Sripanidkulchai. "Enabling Live Migration of Containerized Applications Across Clouds". In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020. DOI: 10.1109/INFOCOM41043.2020.9155403.

[5] Abhishek Bhattacharyya et al. "Multi-vendor OpenROADM Testbed Supporting the Live-Migration of a 5G gNB-CU-UP Container". In: *2025 International Conference on Optical Network Design and Modeling (ONDM)*. 2025. DOI: 10.23919/ONDM65745.2025.11029220.

[6] Carmen Carrión. "Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges". In: *ACM Comput. Surv.* 55.7 (Dec. 2022). DOI: 10.1145/3539606.

[7] Thanawat Chanikaphon and Mohsen Amini Salehi. "UMS: Live Migration of Containerized Services across Autonomous Computing Systems". In: *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. 2023. DOI: 10.1109/GLOBECOM54140.2023.10437519.

[8] Ahmed Chebaane, Simon Spornraft, and Abdelmajid Khelil. "Container-based Task Offloading for Time-Critical Fog Computing". In: *2020 IEEE 3rd 5G World Forum (5GWF)*. 2020. DOI: 10.1109/5GWF49715.2020.9221486.

[9] Christopher Clark et al. "Live migration of virtual machines". In: *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*. NSDI'05. USA: USENIX Association, 2005, pp. 273–286.

[10] Rohit Das and Subhajit Sidhanta. "LIMOCE: Live Migration of Containers in the Edge". In: *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 2021. DOI: 10.1109/CCGrid51090.2021.00070.

[11] Jordi Guitart. “Practicable live container migrations in high performance computing clouds: Diskless, iterative, and connection-persistent”. In: *Journal of Systems Architecture* 152 (2024). DOI: 10.1016/j.sysarc.2024.103157.

[12] Sayed Mohsen Hashemi et al. “A new approach for service activation management in fog computing using Cat Swarm Optimization algorithm”. In: *Computing* 106.11 (2024). DOI: 10.1007/s00607-024-01302-0.

[13] Barbara Kitchenham. “Procedures for Undertaking Systematic Reviews”. In: (Jan. 2004).

[14] Heiko Koziolek, Andreas Burger, and Abdulla Puthan Peedikayil. “Fast state transfer for updates and live migration of industrial controller runtimes in container orchestration systems”. In: *Journal of Systems and Software* 211 (2024). DOI: https://doi.org/10.1016/j.jss.2024.112004.

[15] Xiaoyu Li et al. “Software-based Live Migration for Containerized RDMA”. In: *Proceedings of the 8th Asia-Pacific Workshop on Networking*. APNet ’24. Sydney, Australia: Association for Computing Machinery, 2024. DOI: 10.1145/3663408.3663416.

[16] Yeonjoo Lim and Jong-Hyouk Lee. “Container-based Service Relocation for Beyond 5G Networks”. In: *Journal of Internet Technology* 23.4 (2022). DOI: 10.53106/160792642022072304026.

[17] Yogesh Lohumi et al. “Recent Trends, Issues and Challenges in Container and VM Migration”. In: *2023 International Conference on Computer Science and Emerging Technologies (CSET)*. 2023, pp. 1–5. DOI: 10.1109/CSET58993.2023.10346895.

[18] Yahui Lu and Yu Jiang. “A Container Pre-copy Migration Method Based on Dirty Page Prediction and Compression”. In: *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. 2023. DOI: 10.1109/ICPADS56603.2022.00097.

[19] Sorawit Manatura et al. “ FastMig: Leveraging FastFreeze to Establish Robust Service Liquidity in Cloud 2.0 ”. In: *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. Los Alamitos, CA, USA: IEEE Computer Society, July 2024. DOI: 10.1109/CLOUD62652.2024.00019.

[20] M Patel, A Mehta, and Sachin Patel. “IJTPE Journal CONTAINER MIGRATION AND PLACEMENT IN HYBRID CLOUD-FOG ENVIRONMENT: SYSTEMATIC REVIEW”. In: *International Journal on Technical and Physical Problems of Engineering* 14 (Mar. 2022), pp. 130–135.

[21] Leonardo Poggiani et al. “Live Migration of Multi-Container Kubernetes Pods in Multi-Cluster Serverless Edge Systems”. In: 2024. DOI: 10.1145/3660319.3660330.

[22] Leonardo Poggiani et al. “Live Migration of Multi-Container Kubernetes Pods in Multi-Cluster Serverless Edge Systems”. In: *Proceedings of the 1st Workshop on Serverless at the Edge*. SEATED ’24. Pisa, Italy: Association for Computing Machinery, 2024, pp. 9–16. DOI: 10.1145/3660319.3660330.

[23] Shunmugapriya Ramanathan et al. “Demonstration of Containerized Central Unit Live Migration in 5G Radio Access Network”. In: *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. 2022. DOI: 10.1109/NetSoft54395.2022.9844071.

[24] Zeineb Rejiba and Javad Chamanara. “Custom Scheduling in Kubernetes: A Survey on Common Problems and Solution Approaches”. In: *ACM Comput. Surv.* 55.7 (Dec. 2022). DOI: 10.1145/3544788.

[25] Chenghao Rong et al. “Exploring the Layered Structure of Containers for Design of Video Analytics Application Migration”. In: *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. 2022. DOI: 10.1109/WCNC51071.2022.9771659.

[26] S. Rukmini and Shridevi Soma. “An Optimized Beluga Whale Approach for Migration to Reduce Power and Service Level Agreement in Real-Time System”. In: *Contemporary Mathematics (Singapore)* 6.1 (2025). DOI: 10.37256/cm.6120253854.

[27] S. Rukmini, Shridevi Soma, and Rajkumar Buyya. “A Novel Approach for Energy-Efficient Container Migration Using GNBO”. In: *Contemporary Mathematics (Singapore)* 5.3 (2024). DOI: 10.37256/cm.5320243085.

[28] Ioannis Sarrigiannis et al. “Fog-Enabled Scalable C-V2X Architecture for Distributed 5G and Beyond Applications”. In: *IEEE Network* 34.5 (2020). DOI: 10.1109/MNET.111.2000476.

[29] Gursharan Singh et al. “A secure and lightweight container migration technique in cloud computing”. In: *Journal of King Saud University - Computer and Information Sciences* 36.1 (2024). DOI: 10.1016/j.jksuci.2023.101887.

[30] Shridevi Soma and 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.

[31] Wissem Soussi, Gürkan Gür, and Burkhard Stiller. “Democratizing Container Live Migration for Enhanced Future Networks - A Survey”. In: *ACM Comput. Surv.* 57.4 (Dec. 2024). DOI: 10.1145/3704436.

[32] Robert Welch et al. *Engineering Supercomputing Platforms for Biomolecular Applications*. 2025. arXiv: 2506.15585 [physics.bio-ph].

[33] Xingju Xie et al. “Multi-Container Migration Strategy Optimization for Industrial Robotics Workflow Based on Hybrid Tabu-Evolutionary Algorithm”. In: *IEEE Transactions on Services Computing* 17.5 (2024). DOI: 10.1109/TSC.2024.3440054.

[34] Bo Xu et al. “Sledge: Towards Efficient Live Migration of Docker Containers”. In: *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. 2020. DOI: 10.1109/CLOUD49709.2020.00052.

[35] Hansheng Zhang et al. “KubeSPT: Stateful Pod Teleportation for Service Resilience With Live Migration”. In: *IEEE Transactions on Services Computing* 18.3 (2025). DOI: 10.1109/TSC.2025.3564888.

[36] Liangbin Zhang et al. “Improved Pre-copy Container Live Migration Optimization”. In: *2024 7th International Conference on Electronics Technology (ICET)*. 2024. DOI: 10.1109/ICET61945.2024.10673096.