IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 10, October 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.524

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

Efficient Container Migration for Cloud Applications: A Secure and Cost-Effective Pre-Copy Strategy

Maria Anurag Reddy Basani[1], Anudeep Kandi[2]

Texas A&M University – Corpus Christi – USA[1]

Texas A&M University – Corpus Christi - USA[2]

ABSTRACT: Container migration is a critical capability in cloud computing, enabling dynamic resource allocation and maintaining high availability for applications. This study proposes an optimized container migration technique to enhance migration efficiency by minimizing downtime and reducing data transfer overhead. Unlike traditional methods—Cold, Pre-copy, Post-copy, and Hybrid—our approach utilizes a predictive algorithm to identify and transfer only the essential memory pages during the pre-dump phase. Experimental evaluations show that the proposed method achieves substantial improvements, with a total migration time of 40 seconds compared to 64–95.67 seconds in other techniques and a downtime reduction to 18 seconds, significantly lower than traditional methods. Additionally, our technique minimized data transfer to 5700 bytes, optimizing network usage in large-scale cloud environments. These results confirm that the proposed approach is well-suited for high-demand cloud applications requiring minimal disruption and efficient resource utilization. Future work may extend this method by incorporating machine learning models to adapt to varying workloads and enhance predictive accuracy dynamically.

KEYWORDS: Container Migration, Cloud Computing, Pre-copy Migration, Live Migration, Downtime Reduction, Data Transfer Optimization, Predictive Algorithm, Network Efficiency, Resource Utilization, Migration Techniques Comparison.

I. INTRODUCTION

Container migration involves isolating and detaching all active processes within a container's environment, enabling these processes to be moved seamlessly to a different host and aligned with the new operating system resources [1]. Container migration is critical in transitioning on-premises data to a cloud environment. As organizations look to modernize, migrating existing data and applications requires minimal service disruption, and containerization enables this by isolating workloads for more effortless transfer. By encapsulating applications and their dependencies, containers support scalable, cost-effective migration, ensuring that critical data-related operations continue seamlessly in the cloud. The benefits of container migration in this context extend to enhanced resource management, flexibility in scaling to meet increased workloads, and a structured approach to balancing on-premises and cloud-based operations, aligning with the cloud model. This study addresses these benefits by proposing an optimized migration approach tailored to cloud systems' specific data engineering needs.

Achieving live migration between physical hosts relies on techniques like checkpoint and restore [2], [3]. This process maintains container logs and open network connections, offering advantages such as clear separation of hardware and software, streamlined maintenance, fault resilience, enhanced data accessibility, and adaptable load balancing across physical hosts [4]. Migration can also help reduce power usage by consolidating containers onto fewer hosts, thus freeing idle hardware resources. During migration, resources such as CPU state, network state, and memory state are transferred, with memory state size varying based on the application type [5]. Fully transferring memory states can be time-prohibitive, mainly as containerized applications modify memory more rapidly than it can be transmitted over the network [6]. One significant challenge in container migration is selecting an efficient method for moving memory states to the target host [7]. Reducing downtime and overall migration duration is critical to improving live migration efficiency [8], [9]. Total migration time spans from initiation to completion of the migration process, while downtime refers to the period when the application is inactive on both the source and destination. Practical live migration algorithms aim to reduce downtime and optimize the total migration duration [10]. These migration techniques maintain service continuity and are integral

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

to modernizing on-premises data environments, as they allow workloads to transition efficiently between cloud and local resources in a hybrid setup.

Various live migration algorithms for containers are categorized into three main methods. The first, known as pre-copy migration, initiates by transferring memory states to the destination host, with the source remaining active while selectively updating memory pages during transfer [11]. The second approach, post-copy migration, suspends applications on the source host and transfers a minimal processor state before restoring applications on the target host. The third approach, hybrid migration, combines pre-copy and post-copy techniques by beginning with pre-copy and subsequently suspending and copying the application once the destination receives all memory pages [6]. Afterward, post-copy processes complete any further memory synchronization [12].

Studies indicate that most implementations adopt the pre-copy technique, allowing efficient memory page transfer, which reduces downtime and migration time by holding pages in volatile memory [13]. The effectiveness of pre-copy migration depends on the memory intensity of the application, impacting the migration duration and downtime. Live migration has gained attention due to growing interest in container technology, enabling load balancing and fault tolerance by relocating active applications between hosts. This approach facilitates the application's transfer to a new host, maintaining its original state [14]. Early checkpoint and restore implementations began within an in-kernel framework, with optimizations to reduce pre-dump size for reuse in successive steps. This approach enables the prediction of essential pages for updates, thus decreasing data transfer during migration.

This study's primary contributions address the challenge of repeatedly copying modified (dirty) pages in container checkpoint files. This process can significantly extend application downtime and raise migration costs, especially in AI-driven cloud environments where performance is susceptible to downtime impacts [15, 16]. To tackle this, we propose a predictive pre-copy migration technique that identifies and isolates modified pages, effectively reducing pre-dump size and data transmission overhead. Traditional checkpointing methods tend to re-transmit all pages during the pre-dump phase, increasing the load on subsequent iterative dumps. The specific contributions of this work include:

1. A predictive pre-dump model that minimizes data transfer at the outset of migration.
2. An algorithm to efficiently track memory pages and monitor activity status, with methods to calculate page update rates and streamline checkpointing.
3. A probability-based approach to selectively determine which memory pages should be migrated or discarded.
4. An experimental evaluation of the container-CloudSim 4.0 simulator verifies a reduction in pre-dump size and improved migration efficiency.

The research has a targeted approach to addressing downtime and data transfer overhead in container migration—a key concern for cloud applications requiring high availability and cost efficiency. As organizations increasingly adopt cloud strategies for data modernization, efficient container migration ensures minimal service interruption and resource optimization. The proposed predictive pre-copy method enhances the traditional migration process by intelligently selecting only the necessary memory pages for transfer, effectively reducing the data load and associated costs. This approach aligns with the goals of cost-effective cloud adoption by minimizing downtime and network usage. It enhances the security and reliability of live migration in dynamic, large-scale environments, ultimately supporting robust, uninterrupted services for end-users. This article organizes its findings as follows: Section 2 reviews relevant research, leading to the problem statement in Section 3. Section 3.1 explains the system's architecture, followed by the proposed pre-dump model in Section 3.2. Section 3.4 outlines the secure live migration methodology, with experimental evaluation and results discussed in Sections 5 and 6 and concluding insights provided in Section 8.

II. LITERATURE REVIEW

Live migration in containers involves relocating applications across physical devices or cloud environments without disrupting the client connections [17]. During migration, essential elements like the file system, memory, and network setup of containers hosted on bare-metal hardware are transitioned to a new host machine with minimal downtime [18]. Container migration is typically managed through two critical decisions: selecting the moving container and determining the most effective transfer method [18]. The primary techniques used for migration are pre-copy, post-copy, and hybrid approaches [19]. One of the central tasks in container migration is managing memory page transfers. In the pre-copy approach, an initial memory page copy occurs, followed by iterative page copying. In contrast, post-copy involves

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

shutting down the process initially and transferring container-specific state data to the target host, where the process is then restarted and synchronized with the source memory pages. These migration methods require adequate time to prepare necessary resources, with post-copy timing influenced mainly by workload speed and network connectivity. Pre-copy migration is often favored for containers with shorter lifespans to simplify the migration process.

Al-Dhuraibi et al. (2017) introduced a container model that autonomously adjusts vertical elasticity, scaling memory and CPU usage based on workload demands, thus lowering container operating costs. This vertical elasticity model enhanced resource utilization and boosted performance by 37.63. Yu and Huan (2015) detailed the phases of pre-copy migration and discussed how modified (dirty) pages are identified during live migration. Their experiments revealed that SRVM effectively migrated packets without compromising application data integrity, reducing downtime [20]. Additionally, Soltesz et al. (2007) proposed using Linux-VServer to manage resource isolation within containerized environments, comparing it to Xen and highlighting superior device efficiency in Linux-based systems [21]. Luo et al. (2015) presented a method using data compression and deduplication, precisely the RLE approach, to eliminate redundant memory data during migration. Their approach improved storage efficiency and reduced CPU resource overhead by applying hash-based fingerprints and LRU hash tables for runtime image identity [22]. Ansar et al. (2018) proposed an optimized pre-copy algorithm utilizing a Gray-Markov prediction model to track memory modification rates. This model identified high modification pages, which decreased migration time and downtime while enhancing resource utilization and user experience [23].

In alignment with the Open Container Initiative (OCI), Nadgowda et al. (2017) developed Voyager, which combines CRIU memory-based migration with data federation to minimize downtime by synchronizing data between source and destination hosts [24]. Molto et al. (2017) explored an open standard workflow for deploying coherent HDCI applications on virtual machines and Docker containers. This approach supported simultaneous deployment across various platforms using the TOSCA standard and DevOps tools, allowing applications to be installed on VMs seamlessly [25]. Mirkin et al. (2018) proposed the OpenVZ container checkpoint function, which supports transparent app and network connection restarts on different hosts. Using loadable kernel modules and user utilities, they found that minimizing dump file size reduced service delays between stop and resume operations during migration [14]. Elghamrawy et al. (2017) discussed discrepancies in prediction models for memory page behavior, highlighting the need for optimized live migration. They used a genetic algorithm with a non-dominated sorting approach to improve container management and resource allocation, which yielded superior results in cloud management tasks [11].

Bhardwaj and Krishna (2019) compared pre-copy, post-copy, and hybrid migration techniques, focusing on reducing memory transfer frequency to optimize migration time by predicting operational trends [26]. Patel et al. (2019) addressed migration efficiency from an energy-saving standpoint, proposing an approach that reduces power consumption by tracking modified memory pages through reuse distances. However, the complexity of this algorithm presented significant implementation challenges [27]. Lastly, Al-Dhuraibi et al. (2017) recommended a scheduling model that uses memory compression techniques like RLE and Huffman coding to expedite migration, balancing system load and ensuring efficient resource distribution [28].

In pre-copy migration, resource availability between the source and destination is confirmed before migration. This method ensures that memory data and application information are shared across hosts. During iterative copying, the source sends memory pages in rounds, where only updated pages from previous iterations are transferred in subsequent rounds. Finally," Stop-and-Copy" suspends the container to send high-modification memory pages, enhancing migration efficiency by forecasting the probability of changes in memory pages [29].

Table 1. SUMMARY OF RELATED WORK IN CONTAINER MIGRATION

Study	Focus	Method/Approach	Key Findings
[17]	Container live migration	Migration of file system, memory, and network with minimum downtime	Effective minimal downtime migration of containers across physical/cloud environments
[18]	Migration methods and con- trainer selection	Pre-copy, post-copy, and hybrid migration approaches	Outlined main tasks in container migration: efficient memory page handling and reducing resource time

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

[28]	Autonomous container elasticity	Vertical elasticity scaling based on workload	37.63% improvement in resource use and performance over horizontal elasticity, reducing migration time
[7]	Edge service management	Service migration to edge servers based on user location	Reduced file sync overhead and improved service transfer efficiency for mobile edge computing
[20]	Pre-copy phases and dirty page management	SRVM framework for live migration	Efficient packet migration with reduced downtime, maintaining data integrity
[21]	Resource containerization in Linux	Linux-VServer comparison with Xen	Demonstrated isolation efficiency in Linux systems for resource and security management
[22]	Memory data compression and deduplication	RLE and hash-based fingerprinting	Improved storage efficiency with reduced CPU overhead
[23]	Optimized pre-copy algorithm for live migration	Gray-Markov prediction model	Reduced migration time and improved resource utilization by identifying high-modification pages
[24]	OCI-compliant container migration	Voyager with CRIU memory and data federation	Minimized downtime by synchronizing source and target host data
[25]′	HDCI application deployment in VMs and Docker	TOSCA standard with DevOps tools	Facilitated multi-platform deployment and improved resource management
[14]	Container checkpoint and resume	OpenVZ with loadable kernel modules	Reduced service delays by optimizing checkpoint and restore functions
[11]	Predicting memory page stability	Genetic algorithm for memory page prioritization	Enhanced container resource management and elasticity
[26]	Comparison of migration techniques	Pre-copy, post-copy, and hybrid methods	Found pre-copy most effective, optimizing memory transfer by forecasting operational trends
[27]	Energy-efficient container migration	Reuse distance tracking for modified memory pages	Reduced energy consumption, though complex to implement
[28]	Memory compression techniques in migration	RLE, Huffman coding for memory compression	Improved migration time and resource load balancing
[29]	Iterative pre-copymigration optimization	Prediction model for memory page updates	Enhanced migration efficiency with reduced iteration count and downtime

III. METHODOLOGY

1. Problem Formulation

Containers enhance the performance of cloud applications by providing a stable and isolated environment, which is particularly beneficial for applications demanding uninterrupted service, such as healthcare systems [30]. However, efficient container migration is challenged by the need to minimize downtime and reduce network overhead. This work focuses on a predictive model to improve the migration process by optimizing the pre-dump phase. The goal is to reduce data transfer by selectively transferring only unmodified pages. This raises critical questions: How can we forecast page modifications with precision? What predictive model will minimize unnecessary data transfer? The primary objective is to reduce the total data transfer during the pre-dump phase. The total data transfer T is defined as:

$$T = \sum_{i=1}^{N} \delta(P_i).M(P_i)$$

Where $\delta(P_i)$ is an indicator function, equal to 1 if page P_i is modified and 0 otherwise, and $M(P_i)$ is the memory size of P_i. The goal is to minimize T by sending pages with low modification probability. To achieve this, we introduce a

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

predictive function $P_{mod}(P_i)$, representing the probability of a page P_i undergoing modifications. Pages are only included in the pre-dump set if $P_{mod}(P_i) < T_{mod}$, giving us the selection criterion:

$$Predump\ Set = \{P_i \mid P_{mod}(P_i) < T_{mod}\}$$

Using historical data, we define $P_{mod}(P_i)$ as:

$$P_{mod}(P_i) = \frac{1}{K}\sum_{k=1}^{K} mod_{status}(P_i, t_k)$$

where $mod_{status}(P_i, t_k)$ is 1 if P_i was modified at time t_k and 0 otherwise, and K is the number of historical observations.

2. System Model and Workflow

The system model, illustrated in Figure 2, includes components essential for isolated container operation, such as configuration files, OS dependencies, and kernel requirements. When a container requires migration, an image containing its state, dependencies, and memory contents is created. Let M_{total} represent the total memory, and M_{pre} denotes the memory selected for pre-dump. We aim to minimize M_{pre} to optimize bandwidth:

$$M_{pre} = \sum_{i \in predump\ set} M(P_i)$$

This selection process depends on historical modification probabilities to limit the initial data transferred.

3. Optimized Pre-Dump Selection Protocol

The optimized pre-dump selection protocol reduces unnecessary memory transfers by predicting page activity. Let A(Pi) denote the activity score of each page based on recent modifications, and DF(Pi) signify whether a page has recently been modified (" dirty flag"). Pages with DF(Pi) = 1 are excluded from the initial transfer.

Algorithm 1 Predictive Pre-Dump Page Selection Protocol

1: Initialize $A(P_i) = 0$, $DF(P_i) = 0$ for all P_i in memory
2: for each page, P_i do
3: Update $A(P_i)$ based on historical $P_{mod}(P_i)$
4: if $DF(P_i) = 0$ and $P_{mod}(P_i) < T_{mod}$ then
5: Add P_i to Pre-dump Pool
6: else
7: Mark Pi for iterative transfer
8: end if
9: end for

4. Page Update Rate Computation for Optimized Pre-Dump

To refine which pages qualify for pre-dump transfer, the algorithm calculates each page's update rate, determining its likelihood of modification within the next cycle. This is represented by $UR(P_i)$, the update rate of page P_i:

$$UR(P_i) = \frac{1}{R_{max}}\sum_{j=1}^{R_{max}} dirty_{status}(P_i, t_j)$$

Where R_{max} is the maximum number of rounds and dirty _status (P_i,t_j) is 1 if page P_i was marked dirty in round j, 0 otherwise.

Algorithm 2 Page Update Rate Computation

1: Initialize $UR(P_i) = 0$, $R_{max} = 10$
2: for each round, $j \leq R_{max,}$ do
3: for each page, P_i do
4: if $DF(P_i) \neq 0$ then

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

5: *Increment UR(Pi)*
6: end if 7: end for
8: end for
9: Compute UR(Pi) = UR(Pi)/Rmax for each Pi

---

5. Final Pre-Dump Checkpoint Selection

This algorithm finalizes which pages are added to the migration pool by assessing each page's update rate and predictive model. Pages with $UR(P_i) \leq 0.7$ and part of the active pool are marked for transfer.

Algorithm 3 Final Checkpoint Selection for Pre-Dump Phase

1: Initialize migration pools: *SendPool*[] and *DiscardPool*[]
2: for each page, P_i in memory, do
3: if $UR(P_i) \leq 0.7$ and $P_i \in$ Active Pool then
4: Add P_i to SendPool
5: else
6: Add Pi to DiscardPool
7: end if
8: end for
9: Transfer SendPool to the destination host

---

6. Secure and Verified Live Migration

Security in live migration is essential to prevent data tampering, especially over open networks. The probability of security threats, denoted as P_{threat}, is a function of the security protocols applied, such as encryption and authentication levels:

$$P_{threat} = f(encryption\ level, authentication\ policy)$$

Our model uses SSH and SFTP protocols with RSA-2048 encryption and SHA-256 checksum verification. During migration, the initial compressed file size F_{init} and re-transmission F_{retry} contribute to the total data sent:

$$F_{total} = F_{init} + \sum_{k=1}^{R} F_{retry}$$

Where R is the number of re-transmissions due to failed checksum validation.

Algorithm 4 Secure Data Transfer Protocol for Live Migration

1: Set src instance = IP_{src}, dest instance = IP_{dest}
2: Establish SSH connection to *src instance* using key_1
3: Compress checkpoint file, generate SHA-256 checksum
4: Transfer compressed file and checksum to dest instance over SFTP using key2
5: Establish SSH connection to dest instance using key3
6: while checksum validation fails to do
7: Re-transfer the file with the updated checksum
8: end while
9: Extract checkpoint file at dest instance, terminate src instance
10: Start container on the dest instance

This secure transfer protocol validates each step to ensure data integrity throughout the migration, making it reliable for cloud and network-sensitive applications.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

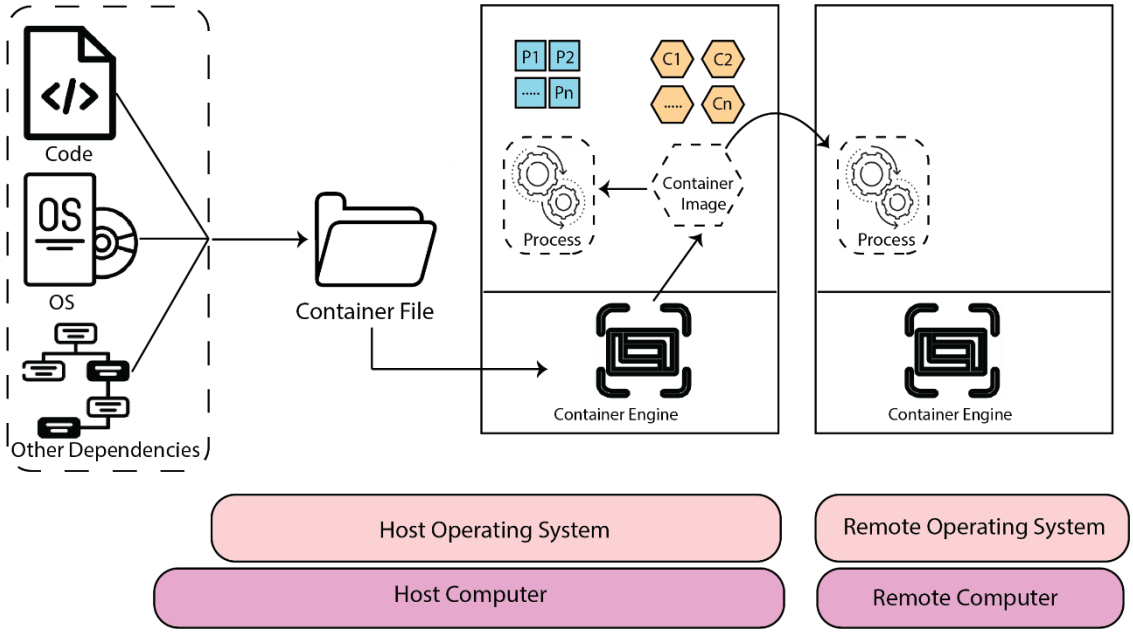

Figure 1. Architecture of Container Migration: Illustrates the migration of container files and memory states from host to remote systems, highlighting key components and synchronization for seamless migration in cloud environments.

IV. EXPERIMENTAL EVALUATION

The implementation was carried out to evaluate the proposed system using Container-CloudSim 4.0 and DL4J libraries, focusing on accurately identifying and tracking memory pages within containers along with their read and write states. The predictive model enables selective migration of only essential memory pages, effectively minimizing unnecessary data transfer. In cases where the entire memory content is transferred, the efficiency of this approach is tested under a worst-case scenario, wherein all container-related pages are sent without filtering. The pre-dump phase timing, determined by the container's size, application type, and workload intensity, reflects this. While the prediction method introduces some CPU load, this overhead remains manageable, as the approach primarily targets network efficiency—its primary objective. This added load impacts the pre-dump phase only, allowing the dump phase to proceed without extra network costs, which is beneficial given network constraints and bandwidth use. Before migrating the container's memory state to the destination, we monitor the page status for read/write activity over recent execution intervals. This information enables accurate assessments of active page counts and modification rates, as detailed in the following results.

V. AND ANALYSIS

To validate the effectiveness of the proposed pre-dump migration model (see Algorithm 1), the system was tested under multiple conditions with varying threshold levels. Here, the threshold levels of 0.6, 0.7, and 0.8 were implemented across three container sets of different sizes (5, 10, and 15 containers). Figures and tables provide space to illustrate these results, which capture key performance indicators under each threshold condition.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

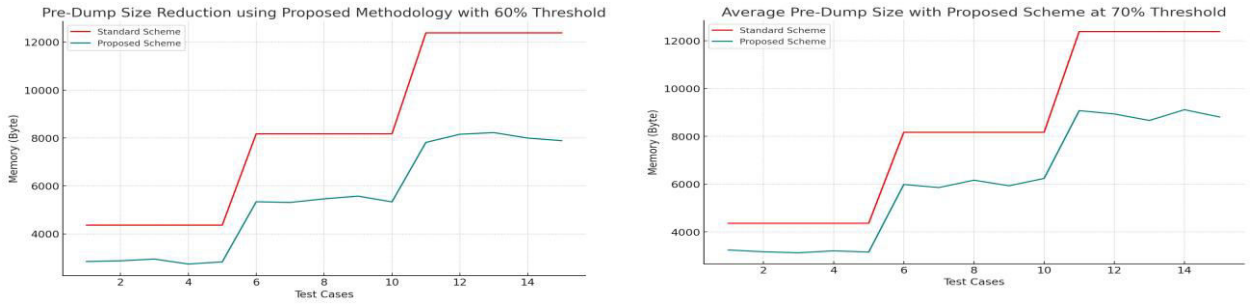

Figure 2. Pre-Dump Size Reduction using Proposed Methodology with 60% and 70% Threshold

In Table 2, we observe that with a 0.6 threshold level, significant reductions in data migration size were achieved. For instance, the pre-dump size for 5, 10, and 15 containers using the proposed approach was 2849.8B, 5404.8B, and 8017.8B, respectively, compared to 4367B, 8176B, and 12387B in standard migration. With threshold values at 0.7 and 0.8, further analysis illustrates the effect of increasing thresholds on data reduction and efficiency:

Table 2. TABLE II PRE-DUMP SIZE COMPARISON AT 60% AND 70% THRESHOLD LEVELS

Container Set	60% Threshold			70% Threshold		
	Pre-Dump Size	Pre-Dump Size (Proposed)	Reduction (%)	Pre-Dump Size	Pre-Dump Size (Proposed)	Reduction (%)
5 Containers	4367B	2849.8B	34.74%	4367B	3187.4B	27.01%
10 Containers	8176B	5404.8B	33.89%	8176B	6040.4B	26.12%
15 Containers	12387B	8017.8B	35.27%	12387B	8923.6B	27.9%

The results across various thresholds indicate that as the threshold level increases, the data sent during pre-dump also rises, reflecting a trade-off between migration speed and data reduction.

1) Comparative Analysis of Migration Techniques

Below are figures allocated to illustrate the migration performance across multiple approaches, including the traditional pre-copy, hybrid, and proposed methods. The standard pre-copy and hybrid methods transfer more data in the pre-dump phase, leading to more significant network usage.

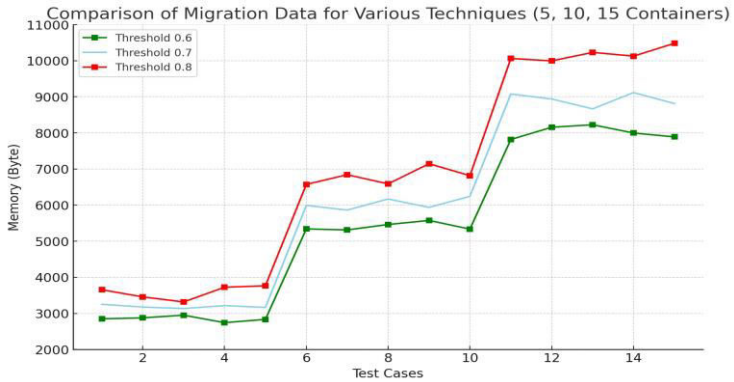

Figure 3. Comparison of Migration Data for Various Techniques (5, 10, 15 Containers)

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

Only pages with low modification probability are transferred in the pre-dump phase in the proposed approach, significantly reducing total data transferred, especially compared to virtual machine (VM)-based approaches. Another key performance indicator is the time required to transfer the pre-dump. Figure 3 (left) shows this comparison for four approaches. Notably, VM and hybrid methods involve higher data transfer times due to the complete memory dump, whereas the proposed method achieves faster migration by transferring only essential pages. To assess migration performance fully, downtime for each technique is compared in Figure 3 (right). As expected, downtime in VM migration is notably higher than in containerized systems, where downtime is minimized. This aligns with the goals of reducing service interruption in live environments.

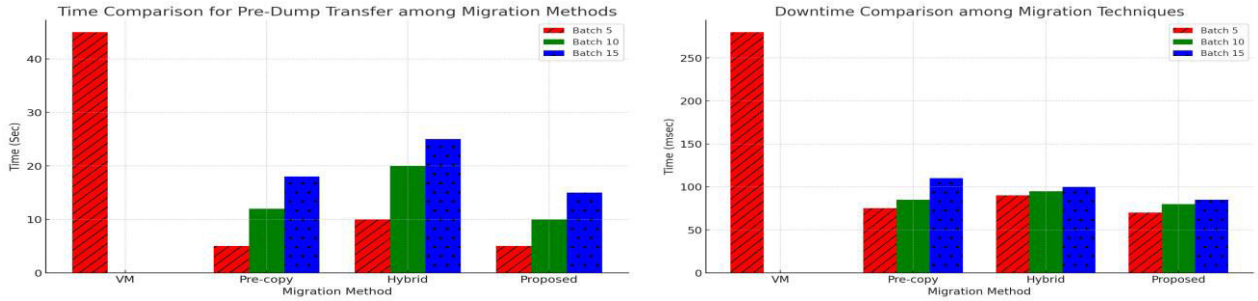

Figure 4. Uptime and Downtime Comparison for Pre-Dump Transfer among Migration Methods

2) Overall Migration Efficiency and Data Transfer Optimization

A final comparison across container migration techniques demonstrates the efficiency of the proposed method over traditional approaches (see Figure 4). By focusing on optimized pre-dump and iterative dump phases, the proposed technique achieves lower overall data transfer, reducing the load on network resources.

Table 3. TABLE IV COMPARISON OF VARIOUS CONTAINER MIGRATION TECHNIQUES IN THREE DIFFERENT SCENARIOS

Technique	Total Migration Time (s)	Downtime (s)	Dump Time (s)	Data Migrated (Bytes)
Cold	64.00	64.00	181.00	8310.00
Pre-copy	85.67	25.67	65.00	8427.67
Post-copy	73.67	11.00	21.00	8091.00
Hybrid	95.67	14.00	28.00	8556.67
Proposed Pre-copy	40.00	18.00	45.00	5700.00

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

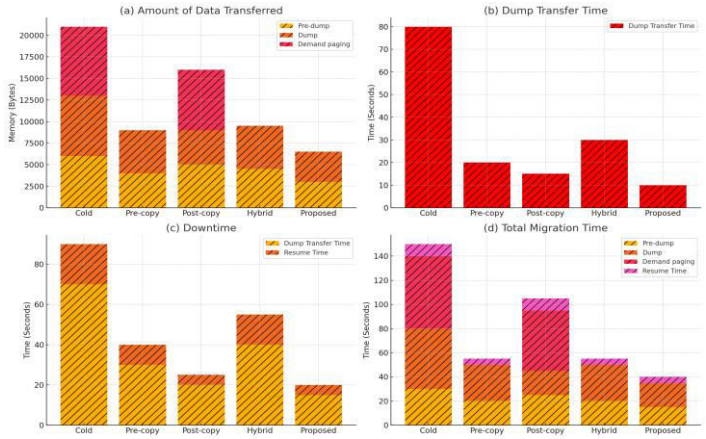

Figure 5. Data Transfer Efficiency across Migration Techniques

VI. CONCLUSION

This study introduced an enhanced container migration technique designed to address the inefficiencies found in traditional migration methods, including Cold, Pre-copy, Post-copy, and Hybrid techniques. Our approach significantly reduces data transfer requirements and minimizes network load by employing a predictive algorithm to select only essential memory pages in the pre-dump phase. Experimental results validate the effectiveness of our method, achieving a total migration time of 40 seconds, notably faster than the 64–95.67 seconds observed with traditional techniques. Downtime was also significantly reduced to 18 seconds, a significant improvement over Cold migration's 64 seconds and standard Pre-copy's 25.67 seconds. Additionally, the proposed technique achieved a dump time of 45 seconds and minimized data transferred to 5700 bytes, showing notable improvements across all metrics. These results demonstrate that the "Proposed Pre-copy" approach enhances migration efficiency and reduces service interruption. It is particularly well-suited for cloud environments with high availability and minimal downtime. By intelligently reducing migration overhead, our technique supports more efficient resource utilization, especially in large-scale cloud systems where network efficiency is essential. Future research could explore further improvements by incorporating adaptive algorithms, such as machine learning models, to predict memory page changes dynamically, thereby broadening the method's applicability across various cloud architectures and workloads.

REFERENCES

[1] R. Stoyanov and M.J. Kollingbaum. Efficient live migration of Linux containers. In International Conference on High Performance Computing, pages 184–193. Springer, 2018.
[2] A. Widjajarto, D.W. Jacob, and M. Lubis. Live migration using checkpoint and restore in userspace (criu): Usage analysis of network, memory, and CPU. Bulletin of Electrical Engineering and Informatics, 10(2):837–847, 2021.
[3] S. Oh and J. Kim. Stateful container migration employing checkpoint-based restoration for orchestrated container clusters. In 2018 International Conference on Information and Communication Technology Convergence, ICTC, pages 25–30. IEEE, 2018.
[4] R. Torre, E. Urbano, H. Salah, G.T. Nguyen, and F.H. Fitzek. Towards a better understanding of live migration performance with docker containers. In European Wireless 2019; 25th European Wireless Conference, pages 1–6. VDE, 2019.
[5] C. Kan. Docloud: An elastic cloud platform for web applications based on docker. In 2016 18th International Conference on Advanced Communication Technology, ICACT, pages 478–483. IEEE, 2016.
[6] G. Singh and P. Singh. A container migration technique to minimize the network overhead with a reusable memory state. International Journal of Computer Networks and Applications, 9(3):350–360, 2022.
[7] L. Ma, S. Yi, and Q. Li. Efficient service handoff across edge servers via docker container migration. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pages 1–13, 2017.
[8] S. Gopalakrishnan, S. Sridharan, S.R. Nayak, J. Nayak, and S. Venkataraman. Central hubs prediction for bio networks by directed hypergraph-ga with validation to covid-19 ppi. Pattern Recognition Letters, 153:246–253, 2022.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.524

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 13, Issue 10, October 2024

|DOI: 10.15680/IJIRSET.2024.1310001|

[9] K. Gupta, S. Roy, R.C. Poonia, S.R. Nayak, R. Kumar, K.J. Alzahrani, M.M. Alnfiai, and F.N. Al-Wesabi. Evaluating the usability of mHealth applications on type 2 diabetes mellitus using various mcdm methods. Healthcare, 10(1):4, 2021.

[10] Y. Al-Dhuraibi, F. Zalila, N. Djarallah, and P. Merle. Coordinating vertical elasticity of both containers and virtual machines. Journal of King Saud University - Computer and Information Sciences, 2018.

[11] K. Elghamrawy, D. Franklin, and F.T. Chong. Predicting memory page stability and its application to memory deduplication and live migration. In 2017 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, pages 125–126. IEEE, 2017.

[12] Y. Fang, Y. Chen, and J. Ge. Improvement of live migration mechanism for virtual machine based on pre-copy. In 2016, 3rd International Conference on Materials Engineering, Manufacturing Technology and Control. Atlantis Press, 2016.

[13] G. Singh and P. Singh. A taxonomy and survey on container migration techniques in cloud computing. In Sustainable Development Through Engineering Innovations: Select Proceedings of SDEI 2020, pages 419–429. Springer Singapore, 2021.

[14] A. Mirkin, A. Kuznetsov, and K. Kolyshkin. Containers checkpointing and live migration. In Proceedings of the Linux Symposium, volume 2, pages 85–90, 2018.

[15] V. Ravi, H. Narasimhan, C. Chakraborty, and T.D. Pham. Deep learning-based meta-classifier approach for covid-19 classification using ct scan and chest x-ray images. Multimedia Systems, pages 1–15, 2021.

[16] V. Ravi, H. Narasimhan, and T.D. Pham. A cost-sensitive deep learning-based meta-classifier for pediatric pneumonia classification using chest x-rays. Expert Systems, page e12966, 2022.

[17] T. Wu, N. Guizani, and J. Huang. Related dirty memory prediction mechanism for live migration enhancement in cloud computing environments. Journal of Network and Computer Applications, 3:1–14, 2017.

[18] K. Gupta, S. Roy, R.C. Poonia, R. Kumar, S.R. Nayak, A. Altameem, and A.K.J. Saudagar. Multi-criteria usability evaluation of mhealth applications on type 2 diabetes mellitus using two hybrid mcdm models: Codas-fahp and moora-fahp. Applied Sciences, 12(9):4156, 2022.

[19] H. Nie, P. Li, H. Xu, L. Dong, J. Song, and R. Wang. Research on optimized pre-copy algorithm of live container migration in cloud environment. In International Symposium on Parallel Architecture, Algorithm and Programming, pages 554–565. Springer, 2017.

[20] C. Yu and F. Huan. Live migration of docker containers through logging and replay. In 2015 3rd International Conference on Mechatronics and Industrial Informatics, ICMII 2015. Atlantis Press, 2015.

[21] S. Soltesz, H. Potzl, M.E. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: a scalable, high-performance alternative¨ to hypervisors. In Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pages 275–287, 2007.

[22] S. Luo, G. Zhang, C. Wu, S. Khan, and K. Li. Boafft: distributed deduplication for big data storage in the cloud. IEEE Transactions on Cloud Computing, 2015.

[23] M. Ansar, M.W. Ashraf, and M. Fatima. Data migration in cloud: A systematic review. American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), 48(1):73–89, 2018.

[24] S. Nadgowda, S. Suneja, N. Bila, and C. Isci. Voyager: Complete container state migration. In 2017 IEEE 37th International Conference on Distributed Computing Systems, ICDCS, pages 2137–2142. IEEE, 2017.

[25] G. Molto, M. Caballer, A. P´ erez, C. de Alfonso, and I. Blanquer. Coherent application delivery on hybrid distributed computing infrastructures of virtual´ machines and docker containers. In 2017 25Th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP, pages 486–490. IEEE, 2017.

[26] A. Bhardwaj and C.R. Krishna. A container-based technique to improve virtual machine migration in cloud computing. IETE Journal of Research, pages 1–16, 2019.

[27] M. Patel, S. Chaudhary, and S. Garg. vmeasure: Performance modeling for live vm migration measuring. In Advances in Data and Information Sciences, pages 185–195. Springer, 2019.

[28] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle. Autonomic vertical elasticity of docker containers with elasticdocker. In 2017 IEEE 10th International Conference on Cloud Computing, CLOUD, pages 472–479. IEEE, 2017.

[29] M. Patel, S. Chaudhary, and S. Garg. Improved pre-copy algorithm using statistical prediction and compression model for efficient live memory migration. International Journal of High Performance Computing and Networking, 11(1):55–65, 2018.

[30] M. Ganeshkumar, V. Ravi, V. Sowmya, and C. Chakraborty. Identification of intracranial haemorrhage (ich) using resnet with data augmentation using cyclegan and ich segmentation using segan. Multimedia Tools and Applications, pages 1–17, 2022.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details