European Journal of Computer Science and Information Technology, 13 (1), 1-15, 2025

Print ISSN: 2054-0957 (Print),

Online ISSN: 2054-0965 (Online)

Website: https://www.eajournals.org/

Publication of the European Centre for Research Training and Development -UK

Improving High Availability Services Using KVM Full Virtualization

Eda Tabaku

Department of Computer Science, University "Aleksandër Moisiu" Durrës, Albania

Email: edatabaku@uamd.edu.al

doi: https://doi.org/10.37745/ejcsit.2013/vol13n1115 Published January 07, 2025

Citation: Tabaku E (2025) Improving High Availability Services Using KVM Full Virtualization, *European Journal of Computer Science and Information Technology*, 13 (1), 1-15

Abstract: *In the digital era, critical sectors like banking, ISPs, and cloud-based services depend on robust, secure networks to ensure uninterrupted service availability. Key operations such as financial transactions, customer support, and online services rely on high system uptime, essential for business continuity and customer trust. Network and server administrators play a vital role in maintaining high availability (HA) for systems like file servers, web servers, database servers, backup systems, and enterprise applications. HA infrastructures often use multiple servers working collaboratively to ensure continuous service. If one server fails, another takes over seamlessly, preventing significant disruptions. Virtualization technology enhances HA systems by hosting services on virtual machines (VMs) that can migrate between physical nodes within a cluster. This ensures uninterrupted service even during hardware failures. This paper examines the use of open-source tools for managing HA services, focusing on cost efficiency and system optimization. The Heartbeat program facilitates real-time VM migration across cluster nodes, maintaining service continuity. To improve this process, the Perf+ algorithm is introduced. Perf+ optimizes CPU performance, reduces memory usage, and minimizes downtime by transferring fewer bytes of modified memory pages during migration, reducing CPU and network load. The proposed solution is implemented in an experimental HA system to evaluate the performance of real-time VM migrations. The research analyzes the impact of these migrations on system efficiency, aiming to advance HA infrastructures in virtualized environments through optimized resource utilization and reduced operational interruptions.*

Keywords: virtualization, KVM, full-virtualization, open source, high availability

INTRODUCTION

Virtualization

Virtualization is one of the most widely used and valued technologies by system administrators today. Its popularity and importance have grown due to several key factors. The continuous improvement in the power and performance of x86 hardware is a major reason, with modern processors supporting more memory and executing multiple processes simultaneously, thanks to multi-core architectures. Additionally, the integration of virtualization-specific technologies directly into the latest Intel and AMD processors has made virtualization more accessible and easier to implement in commercial products. The availability of a wide range of virtualization solutions, from desktop environments to server systems, has significantly expanded the options for tailored implementations. Open-source tools like XEN and KVM are particularly favored for their cost-reduction benefits and ability to enhance system capabilities. These advancements make virtualization indispensable, offering flexibility, efficiency, and scalability for modern computing needs.

High Availability and System Reliability

A system is considered available when it performs its intended functions without interruptions. Availability is defined as the probability of a system being operational over a specific time period and is calculated using the formula: Availability = Uptime / (Uptime + Downtime). Uptime and downtime are often replaced with "Mean Time Between Failures (MTBF)" and "Mean Time To Repair (MTTR)," respectively. Systems with high complexity generally have lower overall MTBF due to an increased likelihood of failures. High availability (HA) systems eliminate single points of failure by incorporating redundancy, such as multiple servers working as backups for each other. Critical services, such as banking or ISP operations, require continuous availability (24/7), while less demanding systems may tolerate occasional downtime.
Availability levels range from 99% to 99.99999%, corresponding to allowable downtime from days to mere seconds annually. Achieving HA involves using fault-tolerant designs, redundancy in hardware components, and efficient repair mechanisms to minimize downtime. Service Level Agreements (SLAs) define the required availability level based on business needs. In practice, HA systems use strategies like load-balancing servers or hardware duplication to ensure reliability, making them indispensable for critical applications.

The goal of this paper is to develop a user-mode algorithm that will communicate with the Linux kernel to achieve three main objectives: first, enable real-time migration of virtual machines between network nodes in the event of a controlled failure; second, use the HASH-MD5 technique to transfer the same amount of information with a smaller data size (fewer bytes); and third, reduce the number of modified pages of processes to be migrated between virtual machines. The project will be implemented using paravirtualization techniques via full virtualization through KVM. The objectives include providing an overview of virtualization, its types and techniques, and prior work on using virtualization in clusters with critical services; proposing and implementing a solution for Heartbeat to migrate virtual machines between physical nodes in real-time; developing an algorithm to improve CPU and memory utilization; and evaluating the system's performance during VM migration.

LITERATURE REVIEW

Virtualization has become a cornerstone technology for optimizing resource utilization and enhancing system performance, especially in data centers, cloud computing, and grid computing environments. Live migration of virtual machines (VMs) across hosts is a critical process, enabling load balancing, fault tolerance, and high availability while minimizing service disruption. Studies have shown that downtime during live migration can be reduced to as little as 60ms, as demonstrated by Barham et al. (2003) using KVM VMM. However, performance impacts, such as network bottlenecks and CPU/memory allocation challenges, remain significant, as highlighted by Cherkasova et al. (2007), who emphasized the need for continuous monitoring and diagnostics.

Further comparisons of virtualization platforms, such as those by Quétier et al. (2007), reveal varying trade-offs in isolation, scalability, and memory efficiency among tools like VMware, KVM, and KVM. For example, KVM excels in memory savings but struggles with network isolation, while VMware ensures high-performance isolation. Modern frameworks like Vagrant have facilitated live VM migration across heterogeneous hypervisors, supporting seamless maintenance and load balancing, as Liu et al. (2008) demonstrated. Similarly, advancements in KVM, such as Kernel Samepage Merging (KSM) and SELinux integration, improve resource management and isolation in resource-constrained environments.

Despite these advancements, challenges such as security risks during live migration and the impact on I/O-intensive applications persist. Researchers like Choudhary et al. have explored techniques like pre-copy and post-copy migration to address these issues, focusing on metrics that minimize migration overhead. Open-source platforms like KVM and KVM continue to evolve, offering features like snapshots and "copy on write" images, which enhance deployment efficiency. Collectively, these studies underline the dynamic nature of virtualization, its critical role in modern IT systems, and the ongoing efforts to overcome its limitations for broader, more efficient adoption.

METHODOLOGY

This study employs a structured approach to design, implement, and evaluate an optimized algorithm for resource management in virtualization technologies. The methodology begins with the creation of an advanced algorithm designed to enhance resource utilization—specifically CPU, memory, and I/O—within virtualized environments. The algorithm focuses on improving resource sharing efficiency among virtual machines (VMs), ensuring balanced and effective use of system capacities to achieve higher performance and scalability.

Following the development phase, the algorithm will undergo rigorous testing in a controlled experimental environment. The KVM virtualization platform will serve as the testing framework, allowing real-world implementation and assessment of the algorithm. Performance metrics such as processing speed, CPU utilization, memory consumption, and I/O throughput will be monitored and analyzed under varying workload conditions to evaluate the algorithm's effectiveness.

The final phase involves a comparative analysis of the experimental results against existing resource management techniques in virtualization. This analysis aims to measure improvements in system efficiency, stability, and scalability introduced by the proposed algorithm. Additionally, the study will identify areas for further optimization and offer recommendations for enhancing virtualization systems. By integrating these findings, the research contributes to advancing resource management solutions tailored to diverse IT environments.

DEVELOPMENT OF THE PERF+ ALGORITHM

This paper focuses on designing and implementing a high-availability (HA) infrastructure using Heartbeat on virtual servers, with the proposed Perf+ algorithm at its core. The goal is to enable real-time migration of virtual machines (VMs) between nodes in case of failure, optimize data transfer using the Hash-MD5 technique for reduced byte size, and minimize modified pages during migration. Utilizing full virtualizaton with KVM, the system ensures efficient resource management and minimal downtime. Heartbeat facilitates communication between network nodes via UDP protocol, ensuring stable performance. The Perf+ algorithm enhances HA by transferring modified pages more efficiently, reducing bandwidth usage and improving CPU performance. This approach achieves faster VM migrations and improved reliability, with the results evaluated in subsequent analysis.

Based on the description above, we will construct a general diagram related to the system design.

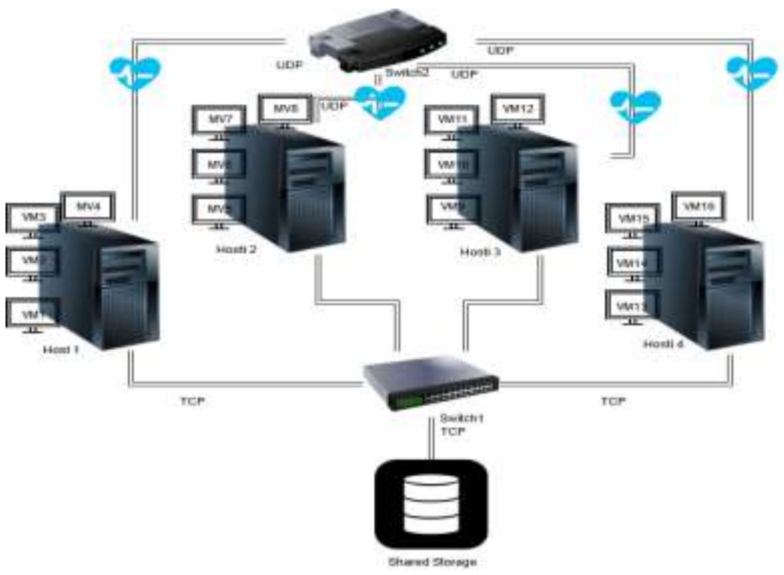

Figure. 1 The experimental environment

Figure 1 shows the topology used to implement the practical development of the paper. In Apendix is provided the flowchart of the Perf+ algorithm and code in the C language for the Perf+ Algorithm.

RESULTS, AND DISCUSSION

To achieve the objective of this work, two key questions arise that deserve special attention:

- Are critical services available during migration?
- How are the services provided to users impacted by the closure of a physical node for maintenance purposes?

The main objective of the measurements we will conduct is to analyze performance in the case of controlled failures, both before and after implementing the Perf+ algorithm, with the aim of improving system performance in scenarios where it provides high availability (HA) services on virtualized platforms with KVM, specifically for representatives of open-source platforms using para-virtualization and full virtualization technology.
Below is a list of measurements to be performed, with their details provided in the corresponding paragraphs:

1. *Downtime*
2. *CPU Performance*
3. *Bandwidth*
4. *Losses*
5. *SLA*

Each measurement experiment is structured into three parts: description, methodology, and results. The description and results of each experiment will be provided in the corresponding experiment paragraphs. The results for each experiment will be graphically compared at the end of each testing paragraph.
Since the methodology is the same for all experiments conducted, it will be presented below, and only the description and results of each experiment will be presented in the subsequent sections.

The results of the following tables are displayed using benchmarks based on Memory Intensive techniques, called: LMBench (Benchmark open-source)1. This tester is installed on all virtual machines at the client level and on the V Center virtual machine at the server level. The tester measures in real-time the entire migration of machines based on the physical memory page technique. The tester is pre-built, sourced from the internet, and written in the C programming language, chosen for its compatibility with hardware.

The algorithm relies on the TCP technique. A counter named Counter++ calculates how many times the SYN flag is set to 1. Each time this flag is set to 1, it indicates that a page has been migrated from one physical machine to another.

Measurements will be conducted in scenarios where 8 virtual machines fail. For this scenario, the number of modified pages migrated in each iteration is provided below. Measurements will consider only 4 iterations for migrating the modified pages. If all the information contained in the modified pages is not migrated within four iterations, the remaining data will be considered lost. The amount of lost information will be analyzed in the subsequent paragraphs.

Below is a graphical representation of the number of pages lost during each failure of the HA system implemented with KVM.

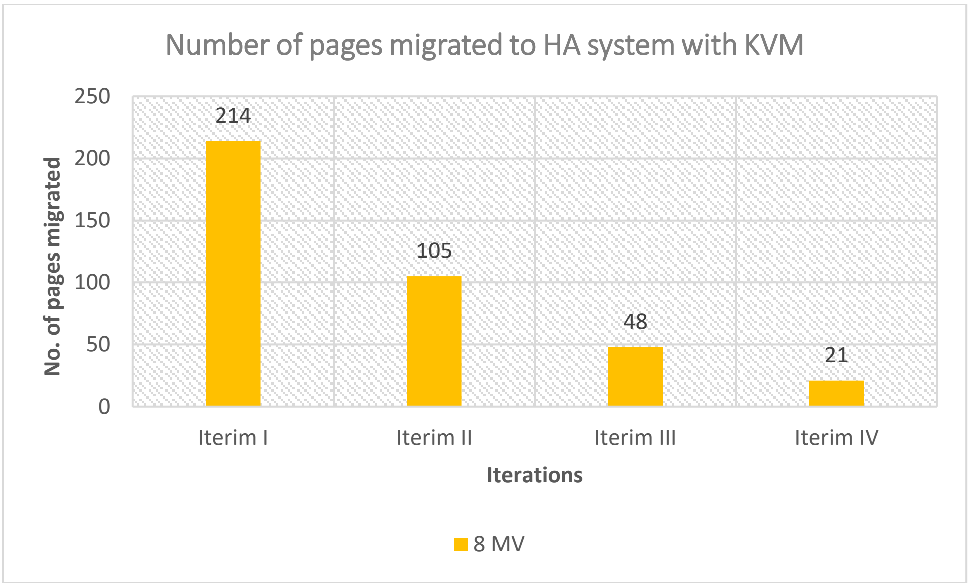

Chart 1 The number of migrated pages in the HA system with KVM

Below is a graphical representation of the number of pages lost during each failure of the HA system implemented with KVM and Perf+.

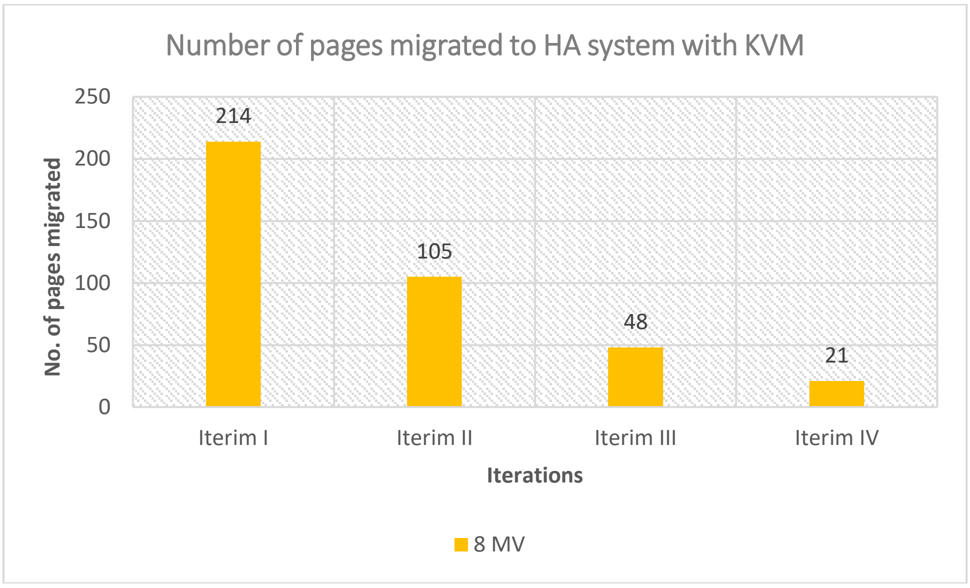

Chart 2 The number of migrated pages in the HA system with KVM and Perf+

Downtime

Description

The downtime will be analyzed, measured, and compared in the case of a controlled system failure, and as a result, a real-time migration will be performed when the system is running with KVM.

Results

- Measurement of the downtime of the HA system implemented with KVM.

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**Downtime**
8 VM	214 pages	105 pages	48 pages	21 pages	2055 ms

Table 3 Downtime of the HA system implemented with KVM

From Table 3, we derive the following results, which highlight the performance of the KVM hypervisor. This hypervisor demonstrates excellent performance, especially regarding memory and I/O devices.

There is also a notable improvement in CPU performance, although not following the same trend observed in the first two entities (as will be seen in the subsequent experimental tables). The advantage of this technology lies directly in the technique called Full virtualizaton with Post-Copy. (The pre-copy technique is not the subject of this study.) This is a powerful tool that significantly enhances the overall performance of the Hypervisor known as DOM0 as well as the Guest machines referred to as DOMU.

• Measurement of the downtime of the HA system implemented with KVM using the Perf+ algorithm

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**Downtime**
8 VM	121 pages	64 pages	31 pages	17 pages	1003 ms

Table 4 Downtime of the HA system implemented with KVM and Perf+.

Relying on the advantages of the full virtualizaton technique, we implement the Perf+ Algorithm in the user space (as explained in paragraph 5.8) by utilizing the Hash-MD5 technique. This approach significantly reduces the number of migrated pages, which in turn automatically reduces the bandwidth consumption. As seen in the table above, the migration time when using the Perf+ Algorithm improves by more than twice compared to the classic case.

Full virtualization requires the virtualization of all hardware components, including:
a. The execution unit
b. CPU memory
c. Physical memory
d. Interrupts and signaling
e. Secondary memory
f. I/O interfaces

The access to the aforementioned elements reduces the flexibility and impact of the algorithm on performance. Meanwhile, KVM, which relies on full virtualizaton techniques, is much more transparent to the entities mentioned above.

Measurement results indicate that the HA system implemented with KVM and Perf+ demonstrates good performance.

CPU Performance

Description
The CPU performance will be analyzed, measured, and compared in the case of a controlled system failure, resulting in real-time migration when the system is implemented with KVM, with and without the Perf+ algorithm.

Results

• CPU performance of the HA system implemented with KVM.

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**CPU Utilization**
8 VM	214 pages	105 pages	48 pages	21 pages	1.93% - 0.88%

Tabel 5 The CPU performance of the HA system implemented with KVM.

- CPU performance of the HA system implemented with KVM and the Perf+ algorithm.

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**CPU Utilization**
8 VM	121 pages	64 pages	31 pages	17 pages	1.87% - 0.68%

Tabel 6 The CPU performance of the HA system implemented with KVM and Perf+.

As seen in the two tables above, the CPU performance improves when using the Perf+ Algorithm compared to the classic case. However, this performance improvement is not as significant as observed in the case of memory. The reasons are:
a. The CPU performance in KVM does not have the same elasticity as memory.
b. The CPU load in KVM in this particular case is relatively low (talking about cases in a few KB).

CPU utilization when 8 virtual machines fail is 1.87%.

Bandwidth

Description
The bandwidth utilization will be analyzed, measured, and compared in the case of a controlled system failure, resulting in real-time migration when the system is running with KVM, with and without the Perf+ algorithm.

Results

- Bandwidth utilization of the HA system implemented with KVM

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**Bandwidth utilization (Network Adapter)**
8 VM	214 pages	105 pages	48 pages	21 pages	42,5%

Tabela 7 Bandwidth utilization of the HA system implemented with KVM.

- Bandwidth utilization of the HA system implemented with KVM and the Perf+ algorithm.

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**Bandwidth utilization (Network Adapter)**
8 VM	121 pages	64 pages	31 pages	17 pages	32,8%

Tabel 8 Bandwidth utilization of the HA system implemented with KVM and Perf+.

As seen in the tables above, KVM has excellent performance in utilizing the traffic passing through the network card. This performance continues to improve with the utilization of the Perf+ algorithm. This is due to the full virtualizaton techniques that the KVM hypervisor utilizes.

Bandwidth utilization when 8 virtual machines fail is 32.8%. The best performance is achieved when the HA system is implemented with KVM and Perf+.

Data Losses

Description
The amount of data lost during the transfer of processes and resources will be analyzed, measured, and compared in the case of a controlled system failure, resulting in real-time migration when the system is implemented with KVM, with and without the Perf+ algorithm.

Results

Using the LM Bench benchmark, the number of modified pages remaining in physical memory is obtained. To get this number, it is sufficient to check the number of M bits for each page using the integer counter.

- Lost data of the HA system implemented with KVM.

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**Number of lost pages**
8 VM	214 pages	105 pages	48 pages	21 pages	7 pages

Tabela9 Lost data of the HA system with KVM.

- Lost data of the HA system implemented with KVM and the Perf+ algorithm.

Number of failed machines	**Number of pages migrated in iteration I**	**Number of pages migrated in iteration II**	**Number of pages migrated in iteration III**	**Number of pages migrated in iteration IV**	**Number of lost pages**
8 VM	121 pages	64 pages	31 pages	17 pages	4 pages

Tabel 10 Lost data of the HA system with KVM and Perf+.

As seen, the best performance is achieved when the HA system is implemented with KVM and Perf+.

Costs in terms of SLA (Service Level Agreement)

SLA Calculation for HA System Implemented with KVM

To analyze the costs in terms of SLA (Service Level Agreement) during a controlled failure and resulting real-time migration, we will calculate the availability of the HA system based on the formula for SLA.

Service Level Agreement (SLA) – The SLA is a transparent agreement between the virtual machines' virtualization system and the service.

Availability Goal – The target availability time is 99.99995% within a month for Virtual Machines (VMs). Availability will be calculated based on the following formula:

P=(TT−TUTTT) ×100P = \ left(\frac{TT - TUT}{TT} \right) \times 100

Where:

- **P** = Availability percentage
- **TT** = Total time of the month in milliseconds (30 days * 24 hours * 60 minutes * 60,000 milliseconds = 2,592,000,000 ms)
- **TUT** = Total uptime (in milliseconds), which includes the downtime of the VM due to planned interruptions.

In the SLA terms, virtual machines are considered unavailable to the virtualization system when the node, or any of its components, is completely down or cannot be accessed or used as per the terms, excluding any conditions arising due to any planned events (as defined in standard terms).

Repair Time Objective – The repair time objective is set to be less than 1 hour from the moment of alert that the VM is in a failure state.

By applying this SLA formula, we can measure the system's availability during failures and migrations. The SLA cost is associated with the downtime and the time it takes to restore service. The better the system's ability to meet the availability target, the lower the SLA cost.

Number of failed machines	**SLA %**
8 VM	99.99999189814818

Table 11 SLA Evaluation in the HA System with KVM

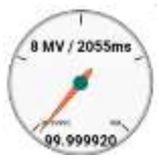

Fig. 3 SLA Evaluation in the HA KVM System

- The calculation of the SLA for the HA system implemented with KVM using the Perf+ optimization algorithm.

Nr i makinave të dështura	**SLA %**
8 VM	99.99999575617284

Table 12
The evaluation of the SLA in the HA system with KVM and Perf+.

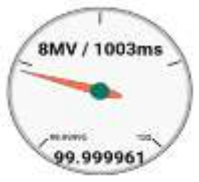

Fig. 4 The evaluation of the SLA in the HA system implemented with KVM and Perf+.

If an outage occurs in the time interval from 0 to 400 ms, 8 Virtual Machines in a "faulty" state will be affected, and the availability is expected to be in the Defect-Tolerant class.

Service Migration from Node 1 to Node 2

When Node 1 goes down, the virtualization system must ensure continuous service delivery from the virtual machine. The connection between nodes should be configured such that the service delivery delay is as close to 0 ms as possible, and in our experimental system, this delay is in the order of microseconds. The migration from Node 1 to Node 2 should utilize information or resources (or both) in such a way that node duplication in the network is eliminated. Additionally, it ensures the control and management of data.

Data is stored in backup files (Node 2) in case external factors cause issues. Therefore, information should be transferred from Node 1 to Node 2 without any data loss. Given the very small chance of data leakage, a protective mechanism for the "distortion" of data encoding should be used during the transmission. Consequently, during the data migration for storage or to prevent data loss, a security mechanism that includes content masking of the information is employed.

CONCLUSIONS

The standard method for failure recovery involves shutting down the virtual machine and starting it on another node. This method causes a significant disruption to the services provided by the virtual machine. Another issue with the standard method is the loss of connection between users and virtual machines. For this reason, it is necessary for the user to reconnect to the virtual machine from the beginning once it is up and running, causing extended downtime.

In this work, we developed an algorithm and implemented it so that Heartbeat manages virtual machines in high availability clusters in real-time (live).

The developed solution consists of a modification made to the script init.d/x0 and the creation of a process whose task is to execute on each node and remain on standby for migration requests. The init.d/x0 script is used when a virtual machine needs to be started, but if the virtual machine is already running somewhere in the cluster, this script will send a real-time migration request instead of restarting the virtual machine.

According to this solution, the preferred method for repairing failures in the system is the real-time migration of virtual machines between physical nodes in the cluster. This method minimizes the interruptions caused by controlled failures (i.e., failures resulting from the shutdown or restart of the node for maintenance reasons) compared to the standard method.

To further improve real-time migration performance, the Perf+ Algorithm was developed at the user level, which uses the Hash-MD5 technique to enhance performance by ensuring that the same amount of information is migrated from one virtual machine to another, utilizing less CPU, memory, and bandwidth. This reduces the number of modified pages that need to be migrated, leading to benefits in reducing the downtime experienced by our system due to a controlled failure of one of the network nodes.

The real-time migration, along with the implementation of the Perf+ Algorithm in the KVM full virtualizaton technique proposed for controlled failure recovery, ensures a very minimal service interruption, in the tens of milliseconds, a downtime so small that users do not notice it.

After implementing the Perf+ Algorithm, measurements were taken for: Downtime, CPU performance, Bandwidth, Losses, and Costs during the transfer of modified pages. The conclusion drawn was that the best performance was achieved by virtualization with KVM and Perf+.

The results show that the interruptions are extremely short, and the client-server connection is maintained during migration.

RECOMMENDATIONS

In order to increase flexibility, physical nodes should be equipped with a large amount of RAM. The more RAM the system has, the more reliable and fault-tolerant it becomes.

To achieve more efficient migration, a dedicated KVM network and its virtual machines can be configured. This way, the internal network of a company is not affected by virtual machine migrations. This measure will further reduce the impact that live migration of virtual machines has on critical services.
Finally, to maintain data integrity on the shared disk ("shared storage"), the configuration of "STONITH" and "fencing" is necessary. This prevents the simultaneous access of the same data block by two or more nodes.

REFERENCES

A survey of virtualization technologies with performance testing. (2010). *arXiv e-Prints*. https://arxiv.org/abs/1010.3233

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., & Warfield, A. (2003). Xen and the art of virtualization. In Proceedings of the 19th ACM Symposium on Operating Systems Principles, 164–177. https://doi.org/10.1145/945445.945462

Chiueh, S. N. T. C., & Brook, S. (2005). A survey on virtualization technologies. *Rpe Report*, *142*.

Choudhary, A., Govil, M. C., Singh, G., Awasthi, L., Pilli, E. P., & Kapil, D. (2017). A critical survey of live virtual machine migration techniques.

Daniels, J. (2009). Server virtualization architecture and implementation. *XRDS: Crossroads, The ACM Magazine for Students*, *16*(1), 8-12.

Douglis, F., & Krieger, O. (2013). Virtualization. *IEEE Internet Computing*, *17*(2), 6-9.

DuVisor: A user-level hypervisor through delegated virtualization. (2022). *arXiv e-Prints*. https://arxiv.org/abs/2201.09652

DuVisor: A user-level hypervisor through delegated virtualization. (2022). *arXiv e-Prints*. https://arxiv.org/abs/2201.09652

G-KVM: A full GPU virtualization on KVM. (2016). *IEEE Xplore Digital Library*. https://doi.org/10.1109/ACCESS.2016.7876385

Jayshri, S., & Mehta, R. (2015). A technical review on comparison of Xen and KVM hypervisors. *International Journal of Advanced Research in Computer and Communication Engineering, 3*(D). Retrieved from https://ijarcce.com

KHV: KVM-based heterogeneous virtualization. (2022). *MDPI Electronics, 11*(16). https://doi.org/10.3390/electronics11162631

Kolyshkin, K. (2006). Virtualization in linux. *White paper*, *OpenVZ*, *3*(39), 8.

Miloji, D. S., Douglis, F., Paindaveine, Y., Wheeler, R., & Zhou, S. (2006). Process migration. ACM Computing Surveys, 32(3), 241–299.

Pasunuru, S. (n.d.). KVM-based virtualization and remote management. *Semantic Scholar*. Retrieved from https://www.semanticscholar.org

Performance exploration of virtualization systems. (2021). *arXiv e-Prints*. https://arxiv.org/abs/2103.07092

Portnoy, M. (2012). *Virtualization essentials* (Vol. 19). John Wiley & Sons.

Power consumption of virtualization technologies: An empirical investigation. (2015). *arXiv e-Prints*. https://arxiv.org/abs/1511.01232

Temporal isolation among virtual machines. (n.d.). *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Temporal_isolation_among_virtual_machines

Zhou, F. F., Ma, R. H., Li, J., et al. (2016). Optimizations for high-performance network virtualization. Journal of Computer Science and Technology, 31(1), 107–116. https://doi.org/10.1007/s11390-016-1614-x