E-ISSN 2240-0524
ISSN 2239-978X

Journal of Educational and Social Research
www.richtmann.org

Vol 15 No 2
March 2025

Research Article

Received: 17 December 2024 / Accepted: 5 February 2025 / Published: 06 March 2025

Optimizing High Availability in Educational Systems Using Xen Paravirtualization

Eda Tabaku[1]

Ejona Duçi[2]

[1]*Department of Computer Science,*
Faculty of Information Technology,
Aleksandër Moisiu University of Durrës,
Durrës, Albania
[2]*Department of Finance and Accounting,*
Faculty of Business,
Aleksandër Moisiu University of Durrës,
Durrës, Albania

DOI: https://doi.org/10.36941/jesr-2025-0054

Abstract

In today's world, where technology plays a crucial role in development, education is also benefiting from advancements in network infrastructure and virtualized services. Virtualization technology and high-availability (HA) infrastructures usage can transform how educational institutions manage and deliver their services. This is especially important for learning environments that require continuous access to critical systems such as data servers, online learning platforms, and student resources. Network and server administrators in educational institutions must ensure the availability and security of their systems. Building high-availability infrastructures enables the creation of more secure and stable environments, ensuring uninterrupted access to essential resources for students and educators. This paper addresses the use of virtualization technology in managing high-availability services using open-source tools. Educational services can be hosted on virtual machines that are automatically migrated between physical nodes of a cluster system in case of a network failure, ensuring a seamless experience for users. The project utilizes the open-source Heartbeat program to enable real-time migration of virtual machines between physical nodes in a cluster. A solution and algorithm (Perf+) will be developed to improve CPU performance and memory and reduce downtime during migration. The solution will be tested in an experimental HA system, measuring the impact of migration on performance. This approach helps educational institutions adopt innovative technologies to enhance the quality of education, ensuring continuous and reliable access to their services.

Keywords: *Education, High Availability, Open-Source, Virtualization, Xen*

1. Introduction

Virtualization refers to the creation of virtual versions of a computer's hardware or software components. These virtual or logical components, developed using virtualization technology, function just like their physical counterparts, providing the same usability and functionality

(Mansouri & Babar, 2021). Virtualization is an important tool for system administrators, helping improve hardware performance, especially with multi-core processors, allowing support for more memory and the processing of multiple processes (Cinque et al., 2022). The integration of virtualization support in Intel and AMD processors has increased the use of virtualized hardware. Additionally, the wide range of virtualization products, including open-source software like Xen and KVM, has reduced costs and increased system capacities (Gill et al., 2024; Pramanik et al., 2024).

In educational institutions, the availability and reliability of IT systems are critical for ensuring uninterrupted services in many important social and economic systems, such as banking, energy, infrastructure, transport, markets, institutional educations, etc (Tabaku et al., 2024). High-availability (HA) systems guarantee that platforms such as virtual classrooms, digital libraries, and collaborative tools are accessible at all times (Stana et al., 2024). These systems are especially vital as education increasingly relies on online platforms and digital resources (Halidini Qendraj et al., 2023; Sota, 2022).

A system is considered available if it is able to perform the tasks it is assigned. Availability is the probability that the system is operational during a specific time period. The measurement is described as "the ratio of time the system is working to the time it is not" (system uptime and downtime) (Zonta et al., 2022).

Availability = Uptime / (Uptime + Downtime)

The term uptime is often replaced with the term mean time between failure (MTBF), and the term downtime is replaced with the term mean time to repair (MTTR).

Overall, the MTBF for the system is less than the MTBF for any individual component of the system. The lower MTBF of the system is divided by the total number of system components since failures typically occur more often in systems with greater complexity.

If we were asked what high availability is, Table 1 would help us understand it, as it shows the maximum amount of time the system is allowed to be unavailable in a year.

Table 1. Availability Classes

Availability classes	Name	Availability	Non-productive time per year
2	Stable	99	3.7 days
3	Available	99.9	8.8 hours
4	High Availability	99.99	52.2 minutes
5	Very High Availability	99.999	5.3 minutes
6	Fault Tolerant	99.9999	32 seconds
7	Fault Resistant	99.99999	3 seconds

The availability of a system is determined by the services it provides. For systems like banking or ISPs, high availability is a must, ensuring they are online 24/7, while for other types of businesses, the level of availability can be more flexible (Mwiya et al., 2022). The quality and level of service depend on the SLA (Service Level Agreement) between the client and the business (Sissodia et al., 2024; Viegas et al., 2021). Critical services have high availability, but due to the uncertainty in the reliability of system components, achieving 100% availability is difficult. Therefore, availability levels range from 99% to 99.99999%.

The objective of this paper is to develop an algorithm in the "user" mode, which will communicate with the Linux kernel and perform three main tasks: first, real-time migration of virtual machines from one node to another in the case of a controlled failure; second, using the HASH-MD5 technique to transfer the same amount of information with a smaller capacity (fewer bytes); and third, reducing the number of modified pages of processes to be migrated between machines (Tabaku et al., 2018). The project will be implemented using paravirtualization techniques via Xen. The main objectives of this work include providing an overview of virtualization, its types, techniques, and previous work related to the use of virtualization in clusters with critical services (Cinque et al., 2024;

Schulz & Annighofer, 2022). Another goal is to propose and implement a solution that enables the Heartbeat program to migrate virtual machines from one physical node of the cluster to another in real time. The paper also aims to develop an algorithm that improves CPU and memory utilization efficiency, as well as to evaluate the system performance during the migration of virtual machines from one node to another (Jazaeri et al., 2024). By providing an overview of virtualization, its techniques, and its applications in critical systems, this paper highlights the potential of virtualization technology to revolutionize education (Rista et al., 2023; Stana et al., 2019). It demonstrates how HA systems can be employed to support the continuous delivery of high-quality educational services, enabling institutions to meet the demands of modern learning environments effectively (Bushati Hoxha et al., 2024; Javed & Alenezi, 2023).

2. Literature Review

Following this section, several studies conducted by various international authors regarding virtualization processes, live migration, and more will be presented.

2.1 Real-Time Migration of Virtual Machines:

Real-time migration of virtual machines enables the transfer of operating systems between hosts, providing a clear separation between hardware and software while facilitating fault management, load distribution, and system maintenance. This ensures high system availability with minimal downtime (Zolfaghari et al., 2021).

This paper aims to demonstrate the migration of operating systems while the system remains operational and continues to provide services to users, implemented through Xen VMM. The dynamic adaptation of network bandwidth allows migration to be performed with minimal impact on active processes, significantly reducing the system downtime (Helali & Omri, 2021; Singh et al., 2023).

2.2 Comparison of Hardware Virtualization Performance:

Hardware virtualization enables the hosting of virtual servers on a distributed physical infrastructure, widely used in data centers and web hosting services. All platforms implement isolation techniques to limit resource consumption by guest virtual machines, but these techniques also impact the performance of the guest system (Rista et al., 2020).

In the study conducted, the impact of different virtualization platforms (OpenVz, KVM, Xen, VirtualBox, VMware ESXi) on CPU performance and network memory in a virtualized system were compared (Đorđević et al., 2024). The results showed that virtualization has a similar impact across all platforms when only a single network interface is considered. However, as the number of interfaces and network traffic increases, each VMM encounters a throughput bottleneck, which depends on the size of transmitted packets and is significantly lower compared to a non-virtualized system.

2.3 Comparison of the Scalability of Four Host Virtualization Tools:

Virtualization techniques are becoming increasingly popular in grid computing, as they enable multiple operating systems to run on a single host while providing an isolated execution environment. Some network projects require virtualization tools to run multiple virtual machines per host, raising concerns about scalability.

The study by Enes et al. (2020) compares the scalability of four virtualization technologies (Vserver, Xen, UML, and VMware) by evaluating CPU, disk, memory, and network performance across three scalability metrics: overhead, linearity, and isolation. The findings indicate that each tool exhibits different scalability behaviors in terms of resource usage and isolation (Enes et al., 2020).

Results show that Vserver and Xen outperform VMware and UML in overall performance but lack strong network isolation between virtual machines. Xen also suffers from low inter-VM communication performance. Vserver, while memory-efficient due to a shared kernel, cannot run guest kernels different from the host. VMware provides better isolation but at the cost of increased resource consumption. These insights help users choose the most suitable virtualization tool based on their application requirements (McDonough et al., 2022).

2.4 Comparison of Three CPU Schedulers in Xen:

Adopting virtualization technologies helps enterprises build faster, more dynamic IT infrastructures, optimizing resource utilization and reducing costs. However, managing virtualized environments introduces new challenges, such as dynamic VM allocation and migration in response to application demands.

Another study evaluates three CPU scheduling algorithms in Xen and their impact on application performance. It highlights how scheduling choices and parameters significantly affect I/O-intensive workloads in virtualized environments. To assess performance sensitivity, three benchmark tests were conducted, simulating disk-intensive and web server workloads. Additionally, a simple benchmarking tool, ALERT, was introduced to measure CPU allocation errors. The findings emphasize that scheduler selection plays a crucial role in optimizing performance for virtual machines (Nikdel & Neville, 2024).

2.5 Security in Virtual Machines during Live Migration for KVM

Virtualization, supported by the hypervisor, enables more efficient resource utilization and provides flexibility for system maintenance and load balancing. Live migration of virtual machines (VMs) from one host to another is essential in "Grid" and "Cloud Computing" environments for reasons such as scalability, fault tolerance, and hardware maintenance. However, during live migration, two critical issues arise: security (hijacking the hypervisor's processes or device modules may expose VM information) and reliability (a sudden interruption of the connection can prevent the VM's state from being correctly preserved on the destination platform).

In the study by Ramanathan et al., (2021) a method is proposed for migrating guest virtual machines to another host using KVM, along with a discussion of the security and reliability requirements during this process. Live migration is performed using Virsh and Virsh-Manager; however, there is a highlighted risk that an attacker could gain unauthorized access to the VM state or cause disruptions if the connection between hosts fails. These issues underscore the importance of specific security measures and recovery mechanisms to ensure the stable operation of virtual systems (Ramanathan et al., 2021).

2.6 Live Migration of Virtual Machines with Heterogeneous Composition:

A necessary process enabled by system virtualization is the live migration of virtual machines. Moving an entire operating system environment to the cloud—such as from one machine to another within a data center or a computer grid—benefits from workload balancing, online maintenance, and fault tolerance. Ideally, a virtual machine should be able to migrate between any nodes with matching configurations and resources. However, the heterogeneity of multiple VMMs complicates the live migration of VMs from one VMM to another. In an article by Liu et al. (2014), a live migration framework that unifies heterogeneous VMMs at both the abstraction and implementation levels. Vagrant supports the live migration of VMs between heterogeneous VMMs, with a prototype implemented for live migration between a Xen VMM and a KVM VMM. Results indicate that Vagrant achieves acceptable downtime and total migration times, comparable to live migration in a homogeneous environment (Martinez et al., 2022).

2.7 Analysis of Xen and KVM Hypervisor Performance for Hosting the Xen World Project:

According to Aqasizade et al. (2024), it compares two open-source virtualization platforms, Xen and KVM, as hosts for the Xen World Project, evaluating overall performance, throughput, VM isolation, and scalability. The results show that Xen paravirtualization delivers better I/O performance, while KVM provides better CPU process isolation without requiring modifications to the guest operating system (Tabaku, 2025). Migrating the Xen World Middleware to KVM leverages features such as "copy on write images" and "snapshots," significantly reducing laboratory deployment time. Other capabilities, like "kernel samepage merging" (KSM) and using SELinux on the host, give KVM advantages in memory-constrained environments and enhance security. The author also demonstrates the technical feasibility of running a Xen Domo as a guest within KVM (Aqasizade et al., 2024).

2.8 A Critical Review of Live Virtual Machine Migration Techniques:

Live virtual machine (VM) migration enables the transfer of one or more running VMs from one server to another, ensuring zero downtime and continuous service for users. This process encounters challenges such as load balancing, proactive failure handling, management capacity, resource allocation, and online system maintenance. In the paper presented by Haris et al., (2024), the importance of migrating CPU states, memory content, and disk content is highlighted, and pre-copy, post-copy, and hybrid migration techniques are evaluated. The authors emphasize the impact of performance metrics on migration overhead and propose classifying the methods into theoretical models and practical frameworks. These approaches are grouped into three main categories: (i) migrating one or multiple VMs, (ii) replicating, duplicating, reducing, and compressing VM memory pages, and (iii) managing inter-VM dependencies. All these techniques are compared based on the relevant performance metrics (Haris et al., 2024).

3. Methodology

The methodology of this paper focuses on the development and experimental evaluation of an optimized algorithm for resource management in virtualization technologies. The process begins with the development of an advanced algorithm that operates in the user space and aims to improve the utilization of system resources such as CPU, memory, and I/O in order to achieve better performance and higher scalability in virtualized environments. The algorithm will aim to optimize the management of these resources by enabling more efficient sharing between virtual machines and ensuring they operate with a balanced and efficient use of system capacities (Gharehpasha et al., 2021).

After the algorithm is developed, we will move to the experimental phase, where a controlled environment will be used to test the algorithm and measure its performance under different load conditions. This environment will involve the use of the well-known virtualization platform, XEN, which will be used to implement and test the algorithm in real-world conditions. The experiments will include monitoring performance parameters such as processing speed, CPU usage, memory consumption, and I/O speed during the execution of various workloads (Makrani et al., 2020).

At the conclusion of the experiments, the results will be analyzed and compared with existing virtualization technologies to assess the improvements brought by the proposed algorithm in resource management and overall system performance. This analysis will focus on evaluating the impact of the algorithm on increasing the efficiency and stability of the virtualization system while also providing recommendations for further improvement opportunities. This process will help identify potential optimizations for virtualization systems and contribute to the development of more advanced and suitable solutions for use in various IT environments (Tabaku et al, 2025).

4. The Development of the Perf+ Algorithm

4.1 Description

This paper aims to design a high-availability infrastructure using Heartbeat and its implementation on virtual servers. The goal is to develop a user-mode algorithm that will communicate with the Linux kernel to achieve three main objectives: real-time migration of virtual machines from one node to another in the event of a controlled failure, using the HASH-MD5 technique for transferring information with smaller capacity (fewer bytes), and reducing the number of modified pages of processes to be migrated. The project will be implemented using paravirtualization techniques with XEN.

4.2 Implementation

Based on the description above, we will construct a general diagram related to the system design. Figure 1 shows the topology used to implement the practical development of the paper.

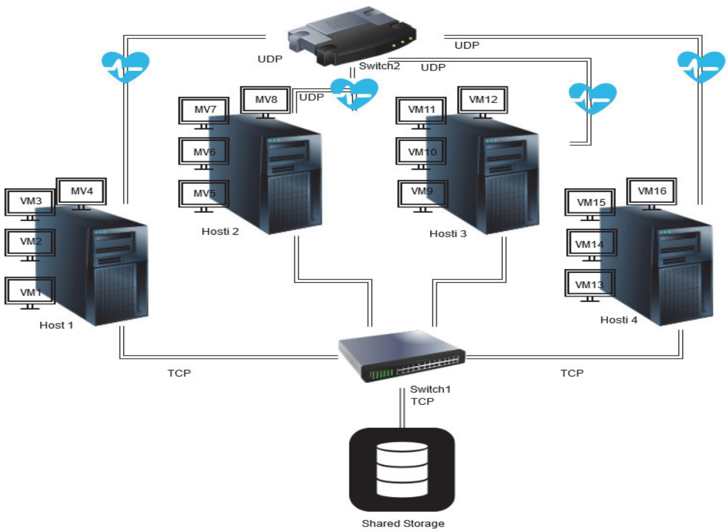


Figure 1: The experimental environment

4.3 Hardware Details

To build the experimental environment shown in Fig.1, five x86-architecture personal computers were used. In the diagram presented in Figure 1, each computer is referred to as host1, host2, host3, host4, and host5.

Each **computer** has the following specifications:

- *Processor:*

Intel® Core™ i7-8565U (1.8 GHz base frequency, up to 4.6 GHz with Intel® Turbo Boost Technology, 8 MB cache, 4 cores)

- *Graphics:*

Integrated: Intel® UHD Graphics 620

- *Memory:*

16 GB DDR4-2400 SDRAM (2 x 8 GB)

- *Storage:*

  512 GB Intel® SSD
- *Wireless technology:*

  Intel® Wireless-AC 9560 802.11 b/g/n/ac (2x2) Wi-Fi® and Bluetooth® 5 Combo (Supporting Gigabit file transfer speeds)

Switch 1

- CISCO Catalyst 3650-48P Layer 3 Switch / 48 Ports - Manageable - 48 x POE+ - Stack Port - 2 x Expansion Slots - 10/100/1000Base-T - Rack-mountable, Desktop / WS-C3650-48PD-S
- Type: Layer 3 Switch
- Stack Port: PO
- Number of PoE+ (RJ-45) Ports: 48
- Port/Expansion Slot Details: 48 x Gigabit Ethernet Network, 2 x 10 Gigabit Ethernet Uplink
- Media Type Supported: Twisted Pair
- Ethernet Technology: 10 Gigabit Ethernet
- Network Technology: 10/100/1000Base-T
- Supported Layer: 4
- Manageable: Yes
- Standard Memory: 4 GB
- PoE (RJ-45) Port: No

Switch 2

NETGEAR 8-Port Gigabit Ethernet Unmanaged Switch (GS308) - Desktop, Sturdy Metal Fanless Housing.

Four of these computers serve as the cluster nodes, while the fifth computer acts as the shared storage. A fifth computer is used for shared storage due to the unavailability of a dedicated storage system. Since this system is built solely for experimental purposes, it is not necessary to ensure high performance and robust data protection, thus avoiding the high costs associated with dedicated storage solutions.

Communication between the physical nodes (host1, host2, host3, host4) and the shared storage is achieved via a gigabit switch (Switch1). The local interface provides sufficient performance for accessing the virtual machine images placed on the shared storage, thanks to the gigabit switch and the local network setup.

On the other hand, to form a cluster between pc1, pc2, and pc3, UDP packets must be exchanged among them to create a second dedicated communication network between the nodes. This requirement explains the inclusion of a second switch in the system.

4.4 Software Details

Each node in the system runs on the Ubuntu Server 16.04 operating system.
Reasons why ise used Ubuntu Server 24.04:

Documentation & Community: Most official docs, tutorials, and forums are in English, simplifying troubleshooting.

Industry Standard: English is commonly used in international and enterprise environments, making collaboration easier.

Tool Compatibility: Many server tools and packages are developed primarily in English, reducing locale issues.

LTS Stability: Ubuntu 24.04 will be a Long-Term Support release (5 years of updates), offering reliability in production.

Training & Certs: Most training materials and certifications are provided in English, streamlining learning.

Detailed implementation guidelines for IT administrators and educators can be found in Appendix A.

4.5 Presentation of the Algorithm Perf +

In this paper, for the implementation of a high availability (HA) system, Heartbeat has been chosen as an open-source tool to maintain continuous contact between network nodes. This chapter focuses on algorithms for improving performance in managing high availability services in virtualized systems. The proposed algorithm, named Perf+, is designed to enhance performance.

The implementation of the HA system will enable real-time migration of virtual machines from one physical node to another. Heartbeat has been used to test high availability in Linux environments, and in this project, the connection between the network nodes is established via switch 2, using the UDP protocol, which ensures good performance for exchanging information between nodes.

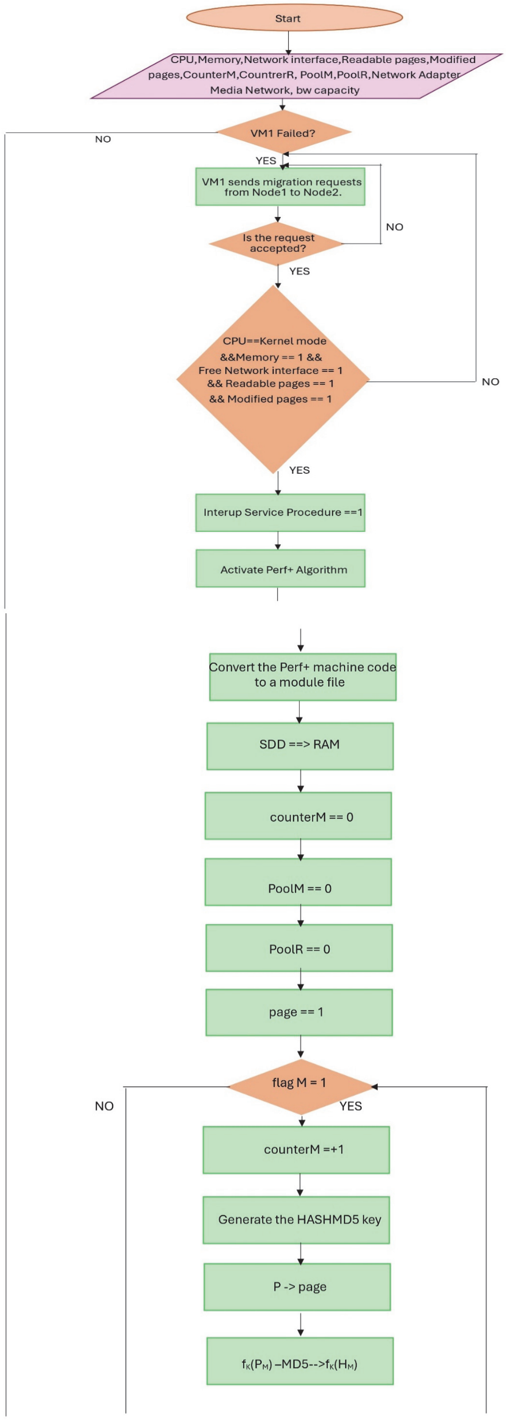
4.6 Algorithm Perf +

The proposed algorithm for modifying the kernel and transferring information with fewer bytes using the Hash-MD5 technique, suggested and developed by me for this project, is called Perf+. The algorithm aims to improve performance during the real-time migration of virtual machines from one physical machine to another. Perf+ uses Hash-MD5 to transfer the same amount of information with fewer bytes, reducing the number of modified pages that need to be migrated and helping to decrease downtime during the migration of virtual machines.

The algorithm starts working when a virtual machine fails and needs to be migrated to another node. Once Heartbeat selects the destination node, the migration begins. The migration process involves transferring both clean and dirty pages. Clean pages are readable and transferred simply, while dirty pages are modifiable and require a short period for migration to minimize modifications and interruptions, thus maintaining system efficiency.

To improve performance, the Perf+ algorithm is implemented, which uses Hash-MD5 to transfer modified pages in a smaller byte size, thereby increasing the efficiency of the high availability (HA) system, reducing downtime, and improving CPU performance and bandwidth. The results will be analyzed in the following chapter.

Below is the flowchart of the Perf+ algorithm.


Start
CPU,Memory,Network interface,Readable pages,Modified pages,CounterM,CountrerR, PoolM,PoolR,Network Adapter Media Network, bw capacity
VM1 Failed?
NO
YES
VM1 sends migration requests from Node1 to Node2.
Is the request accepted?
NO
YES
CPU==Kernel mode &&Memory == 1 && Free Network interface == 1 && Readable pages == 1 && Modified pages == 1
NO
YES
Interup Service Procedure ==1
Activate Perf+ Algorithm
Convert the Perf+ machine code to a module file
SDD ==> RAM
counterM == 0
PoolM == 0
PoolR == 0
page == 1
flag M = 1
NO
YES
counterM =+1
Generate the HASHMD5 key
P -> page
fK(PM) –MD5-->fK(HM)

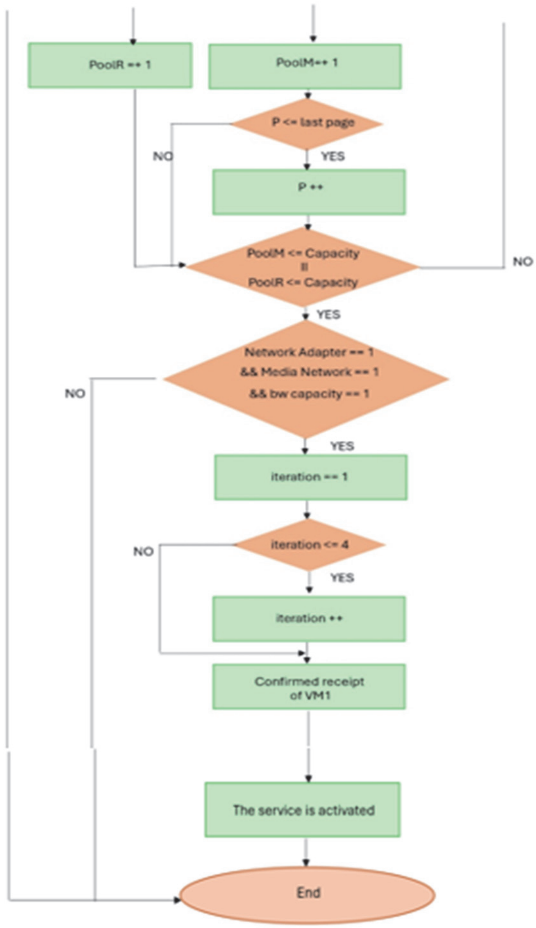


Chart 1. Algorithm Perf +

The algorithm is written in C, the same language used for the Linux kernel. In Appendix **B**, the code in the C language for the Perf+ Algorithm is provided.

5. Measurements, Results, and Discussion

5.1 *Measurements and Experimental Results*

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?

Regarding network services, the experiments are based on the widely used TCP and UDP protocols across the Internet today. Specifically, the heartbeat service will be implemented on the

UDP protocol, while communication between network nodes will be realized through the TCP protocol. This decision is based on the fact that TCP is "connection-oriented" while UDP is "connectionless." For a quick comparison between TCP and UDP, we can refer to Table 2.

Table 2. The most important differences between TCP and UDP

TCP:	UDP:
Reliable Slower than UDP "Connection-oriented""	Unreliable Faster than TCP "Connectionless"

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 Xen, 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 charts:

- Downtime
- CPU Performance
- Bandwidth
- Losses
- SLA

Measuring these metrics (such as total migration time, memory page transfer speed, memory usage, network latency, and communication cost) is incredibly important for several key reasons:

Performance evaluation—Tells us how efficient the migration process is and whether the load is within acceptable levels.

Bottleneck identification—Identifies bottlenecks (e.g., latency, memory, CPU) and helps resolve them.

System optimization—Enables network, memory, or algorithm improvements for better results.

Comparison of methods—Standardized measurement methods allow comparison with other research or solutions.

Resource planning—Helps predict needs for scaling and efficient use of resources.

Increased reliability—Continuous monitoring prevents breakdowns and service interruptions.

Each measurement experiment is structured into three parts: description and results. The description and results of each experiment will be provided in the corresponding experiment charts. The results for each experiment will be chartically compared at the end of each testing chart.

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 in the following tables are presented using a benchmark based on memory-intensive techniques, called LMBench (an open-source benchmark). This tool is installed on all virtual machines at the client level and on the V_Center virtual machine at the server level. LMBench will measure, in real time, the migration process of virtual machines by relying on the technique of physical memory pages. This benchmark is sourced from the internet, is written in C—which is considered the most suitable language for low-level communication with hardware—and its algorithm is based on the TCP protocol. A counter called Counter++ tracks how many times the SYN flag becomes 1; each time this flag becomes 1, a memory page is migrated from one physical machine to another.

The reasons why LMBench was chosen as the main benchmark tool, instead of other alternatives, include:

- Focus on memory performance.

- LMBench is specifically designed for memory-intensive tests, measuring parameters such as memory latency, bandwidth, and the cost of frequent memory accesses. This makes it particularly suitable for a project where memory page migration is critical.
- Credibility and wide usability
- LMBench is an open-source tool with a long history of use in the computer science and engineering community. Opting for a well-known tool helps ensure that the results are more reliable and comparable with similar research.
- Suitability for testing conditions (real-time migration)
- This benchmark allows for real-time measurement of memory page migration, which is essential for evaluating and optimizing the performance of virtual infrastructures.
- Written in C, LMBench interacts directly with hardware and the TCP protocol, enabling high-resolution time measurement.
- Lightweight structure and simple configuration
- LMBench is relatively easy to install and configure across multiple virtual machines, making it ideal for scientific experiments with limited resources.
- Open-source and supportive community
- As an open-source tool, LMBench provides the opportunity to inspect its source code, modify or adapt it to specific experimental needs, and offers a supportive community for technical issues or optimization suggestions.
- Compatibility with migration protocols (TCP)

Migration tests over the TCP protocol are standardized and easily replicable. LMBench provides a solid framework for adhering to this communication model, ensuring systematic and consistent measurement.

For these reasons, LMBench has been chosen as the primary benchmark tool in this research, as it most effectively meets the requirements for measuring performance at the memory-page level and delivers accurate, standardized results for virtual machine migration processes. This facilitates direct comparison of findings with similar work, as well as helping to precisely determine the impact of various migration techniques on the overall system performance.

Measurements will be conducted in scenarios where 4 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 charts.

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

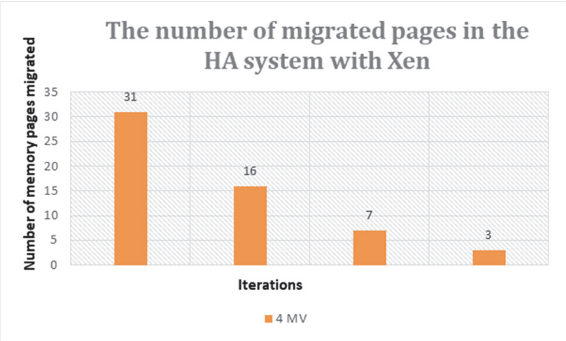


Chart 1. The number of migrated pages in the HA system with Xen

Chart 1 shows the number of memory pages migrated in a Xen-based system across four iterations. From the first column (Iteration I), with 31 migrated pages, it is clear that most of the memory is transferred during the initial phase. In the subsequent iterations (Iterations II, III, and IV), there is a progressive decrease in the number of migrated pages (16, then 7, and finally 3), which is typical for the iterative pre-copy method (or similar approaches).

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

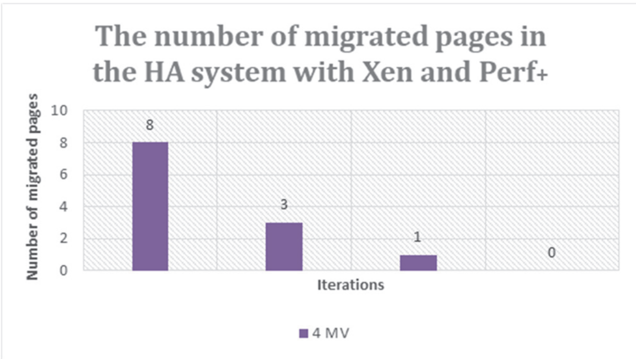


Chart 2. The number of migrated pages in the HA system with Xen and Perf+

The chart shows that in a High Availability (HA) system with Xen and Perf+, the total number of migrated memory pages decreases significantly after each iteration.

5.2 *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 Xen.

Results

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

Table 3. Downtime of the HA system implemented with Xen

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
4 VM	31 pages	16 pages	7 pages	3 pages	210 ms

From Table 3, we derive the following results, which highlight the performance of the Xen 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 Paravirtualization 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 Xen using the Perf+ algorithm

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

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
4 VM	8 pages	3 pages	1 page	0 pages	110 ms

Relying on the advantages of the paravirtualization technique, we implement the Perf+ Algorithm in the user space 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:

- The execution unit
- CPU memory
- Physical memory
- Interrupts and signaling
- Secondary memory
- I/O interfaces

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

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

5.3 *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 Xen, with and without the Perf+ algorithm.

Results

- CPU performance of the HA system implemented with Xen.

Table 5. The CPU performance of the HA system implemented with Xen.

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
4 VM	31 pages	16 pages	7 pages	3 pages	1.55% - 0.41%

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

Table 6. The CPU performance of the HA system implemented with Xen and Perf+.

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
4 VM	8 pages	3 pages	1 page	0 pages	1.36% - 0.29%

a. 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:
The CPU performance in Xen does not have the same elasticity as memory. The CPU load in Xen in this particular case is relatively low (talking about cases in a few KB).

b. CPU utilization when 4 virtual machines fail is 1.72%.

5.4 *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 Xen, with and without the Perf+ algorithm.

Results

- Bandwidth utilization of the HA system implemented with Xen

Table 7. Bandwidth utilization of the HA system implemented with Xen.

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)
4 VM	31 pages	16 pages	7 pages	3 pages	7,8%

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

Table 8. Bandwidth utilization of the HA system implemented with Xen and Perf+.

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)
4 VM	8 pages	3 pages	1 page	0 pages	4,8%

As seen in the tables above, Xen 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 paravirtualization techniques that the Xen hypervisor utilizes.

Bandwidth utilization when 4 virtual machines fail is 34.1%. The best performance is achieved when the HA system is implemented with Xen and Perf+.

5.5 *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 Xen, 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 Xen.

Table 9. Lost data of the HA system with Xen.

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
4 VM	31 pages	16 pages	7 pages	3 pages	1 page

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

Table 10. Lost data of the HA system with Xen and Perf+.

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
4 VM	8 pages	3 pages	1 page	0 pages	0 pages

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

5.6 Costs in terms of SLA (Service Level Agreement)

SLA Calculation for HA System Implemented with Xen

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.

Table 11. SLA Evaluation in the HA System with Xen

Number of failed VMs	SLA %
4 VM	99.99991189814818

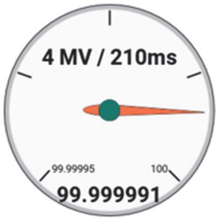


Chart 3. SLA Evaluation in the HA Xen System

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

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

Number of failed Ms	SLA %
4 VM	99.999995757617284

Chart 4. The evaluation of the SLA in the HA system implemented with Xen and Perf+.

If an outage occurs in the time interval from 0 to 400 ms, 4 Virtual Machines in a "faulty" state will be affected, and the availability is expected to be in the Defect-Tolerant class.
In summary, these results are presented as follows:

Table 13. The results of the measurements from the experiment conducted with Xen without and with Perf+

Description	**Results without Perf+ Implementation**	**Results with Perf+ Implementation**
Number of failed machines	4 VM	4 VM
Number of pages migrated in iteration I	31 pages	8 pages
Number of pages migrated in iteration II	16 pages	3 pages
Number of pages migrated in iteration III	7 pages	1 page
Number of pages migrated in iteration IV	3 pages	0 pages
Downtime	210 ms	110 ms
CPU usage	7.80%	1.36% - 0.29%
Bandwidth utilization (Network Adapter)	71%	4,8%
Lost pages	1	0 pages
SLA%	99.999995	99.999991

From the data comparison in the table 13, Perf+ Implementation shows significant improvements in both performance and reliability compared to the setup without this feature:

1. **Number of migrated pages**

Without Perf+: A total of 57 pages across four iterations (31 + 16 + 7 + 3).

With Perf+: A total of 12 pages (8 + 3 + 1 + 0).

This indicates that Perf+ greatly reduces the data volume to be transferred, typically resulting in faster or more efficient migration.

2. **Downtime**

Decreases from 210 ms to 110 ms, almost halving the offline period.

This directly benefits service availability.

3. **CPU usage**

Without Perf+: Approximately 7.80%.

With Perf+: Ranges from 1.36% down to 0.29%.

Such a drastic reduction suggests more efficient resource management, freeing the CPU for other processes.

4. **Bandwidth utilization** (Network Adapter)

Without Perf+: 71%.

With Perf+: 4.8%.

A noteworthy decrease, indicating that Perf+ either minimizes unnecessary traffic or compresses data more effectively.

5. **Lost pages**

Without Perf+: 1 page lost.

With Perf+: 0 pages lost.

This indicates a safer or more reliable migration process using Perf+.

6. **SLA%**

Without Perf+: 99.999995%

With Perf+: 99.999991%

The slight difference has virtually no noticeable impact on user experience or service commitments.

By reducing downtime, drastically lowering CPU and network overhead, and maintaining nearly the same SLA level, Perf+ Implementation proves to be a superior solution for VM migration/management in terms of speed, resource efficiency, and reliability.

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.

6. Conclusions

The standard method for failure recovery in virtualized environments involves shutting down a virtual machine and restarting it on another node. This method leads to significant disruptions in service, which can be particularly problematic in educational environments where students and educators rely on uninterrupted access to online learning platforms, virtual classrooms, and other digital resources. Furthermore, the standard approach results in users losing their connection to the virtual machine and needing to reconnect, causing extended downtime that hinders productivity and the learning process. To address these challenges, an algorithm was developed and implemented to enable Heartbeat to manage virtual machines in high-availability clusters in real-time (live). The solution involves modifying the init. d/xo script and creating a process that operates on standby in each cluster node to handle migration requests. This innovation allows the system to prioritize real-time migration of virtual machines rather than restarting them, thereby minimizing interruptions caused by controlled failures, such as maintenance-related node shutdowns. To enhance the efficiency of real-time migration further, the Perf+ Algorithm was developed. This algorithm utilizes the Hash-MD5 technique to optimize resource usage by transferring the same amount of data with reduced CPU, memory, and bandwidth consumption. By minimizing the number of modified pages requiring migration, the algorithm significantly reduces system downtime during controlled failures, achieving interruptions in the range of milliseconds—imperceptible to end users. For educational institutions, this capability ensures that digital learning environments remain operational even during system maintenance or unexpected node failures. Students and educators benefit from uninterrupted access to essential resources, fostering a seamless learning experience and reducing disruptions to academic activities. Performance measurements conducted after implementing the Perf+ algorithm demonstrated its effectiveness in reducing downtime, improving CPU performance, and optimizing bandwidth usage. When combined with the Xen paravirtualization technique, this approach provides an efficient and cost-effective solution for high-availability systems in education.

By adopting real-time migration and the Perf+ Algorithm, educational institutions can ensure reliable and resilient IT infrastructures, enabling them to meet the growing demands of modern digital learning environments while minimizing service disruptions. The consequences of controlled failures on the network (TCP protocol) were measured experimentally in real services (file transfers, web server, and database server). The results show that the interruptions are extremely short, and the client-server connection is maintained during migration.

7. Recommendations

To ensure seamless operation of digital learning platforms and virtual environments in educational institutions, the following measures are recommended:

1. **Increase Physical Node RAM**

Educational institutions should equip physical nodes with a large amount of RAM to enhance system reliability and fault tolerance. The additional RAM ensures that virtual machines hosting critical services such as virtual classrooms, learning management systems, and student portals can handle higher loads and recover efficiently from failures. This added flexibility supports uninterrupted educational processes, even during peak usage.

2. **Dedicated Xen Network Configuration**

To improve the efficiency of virtual machine migrations, a dedicated Xen network and its virtual machines can be configured. This separation prevents the migration process from impacting the institution's internal network, ensuring that critical services like online assessments, real-time collaboration tools, and digital libraries remain unaffected. By isolating migration traffic, educational institutions can maintain the quality and reliability of their learning environments.

3. **Configure "STONITH" and "Fencing" for Shared Storage**

For educational systems utilizing shared storage solutions, configuring **STONITH** (Shoot the Other Node In The Head) and **fencing** mechanisms is essential to maintain data integrity. These configurations prevent simultaneous access to the same data block by multiple nodes, safeguarding critical information such as student records, academic content, and research data. Ensuring data integrity is vital for maintaining trust in the institution's digital infrastructure and protecting sensitive educational resources.

By implementing these recommendations, educational institutions can build a more robust, reliable, and efficient IT infrastructure, enabling them to deliver uninterrupted, high-quality digital learning experiences.

8. Acknowledgment

This research was conducted with the support of Aleksandër Moisiu University, Durrës, which provided the necessary facilities and resources for the study. The authors also express their gratitude for the financial and technical support received from international projects and partner institutions that contributed to the successful completion of this research. Special thanks are extended to the academic staff and collaborators for their valuable expertise and guidance throughout the process.

References

Aqasizade, H., Ataie, E., & Bastam, M. (2024). Experimental assessment of containers running on top of virtual machines. *IET Networks*. https://doi.org/10.1049/ntw2.12138

Bushati Hoxha, S., Bushi, F., & Gjecka, A. (2024). Application & Challenges Of Big Data Technologies In Albanian Health Care System. *Interdisciplinary Journal of Research and Development*, *11*(3), 120. https://doi.org/10.56345/ijrdv11n316

Cinque, M., Cotroneo, D., De Simone, L., & Rosiello, S. (2022). Virtualizing mixed-criticality systems: A survey on industrial trends and issues. In *Future Generation Computer Systems* (Vol. 129). https://doi.org/10.1016/j.fu ture.2021.12.002

Cinque, M., De Simone, L., & Ottaviano, D. (2024). Temporal isolation assessment in virtualized safety-critical mixed-criticality systems: A case study on Xen hypervisor. *Journal of Systems and Software*, *216*, 112147. https://doi.org/10.1016/j.jss.2024.112147

Đorđević, B., Kraljević, N., & Davidović, N. (2024). Performance Comparison of CPU Hardware-Assisted Features for the Type-2 Hypervisors. *2024 23rd International Symposium INFOTEH-JAHORINA (INFOTEH)*, 1–5. https://doi.org/10.1109/INFOTEH60418.2024.10495932

Enes, J., Expósito, R. R., & Touriño, J. (2020). Real-time resource scaling platform for Big Data workloads on serverless environments. *Future Generation Computer Systems*, *105*, 361–379. https://doi.org/10.1016/j.future .2019.11.037

Gharehpasha, S., Masdari, M., & Jafarian, A. (2021). Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm. *Cluster Computing*, *24*(2), 1293–1315. https://doi.org/10.1007/s10586-020-03187-y

Gill, S. S., Wu, H., Patros, P., Ottaviani, C., Arora, P., Pujol, V. C., Haunschild, D., Parlikad, A. K., Cetinkaya, O., Lutfiyya, H., Stankovski, V., Li, R., Ding, Y., Qadir, J., Abraham, A., Ghosh, S. K., Song, H. H., Sakellariou, R., Rana, O., … Buyya, R. (2024). Modern computing: Vision and challenges. In *Telematics and Informatics Reports* (Vol. 13). https://doi.org/10.1016/j.teler.2024.100116

Halidini Qendraj, D., Xhafaj, E., Kosova, R., & Gjikaj, N. (2023). Factors Affecting the Adoption of Cloud Based Learning Management System: A Decision Approach via Fuzzy Z-AHP. *Interdisciplinary Journal of Research and Development*, *10*(2), 82. https://doi.org/10.56345/ijrdv10n212

Haris, R. M., Khan, K. M., Nhlabatsi, A., & Barhamgi, M. (2024). A machine learning-based optimization approach for pre-copy live virtual machine migration. *Cluster Computing*, *27*(2), 1293–1312. https://doi.org/10.1007/s 10586-023-04001-1

Helali, L., & Omri, M. N. (2021). A survey of data center consolidation in cloud computing systems. *Computer Science Review*, *39*, 100366. https://doi.org/10.1016/j.cosrev.2021.100366

Javed, Y., & Alenezi, M. (2023). A Case Study on Sustainable Quality Assurance in Higher Education. *Sustainability*, *15*(10), 8136. https://doi.org/10.3390/su15108136

Jazaeri, S. S., Asghari, P., Jabbehdari, S., & Javadi, H. H. S. (2024). Toward caching techniques in edge computing over SDN-IoT architecture: a review of challenges, solutions, and open issues. *Multimedia Tools and Applications*, *83*(1). https://doi.org/10.1007/s11042-023-15657-7

Makrani, H. M., Sayadi, H., Nazari, N., Dinakarrao, S. M. P., Sasan, A., Mohsenin, T., Rafatirad, S., & Homayoun, H. (2020). Adaptive Performance Modeling of Data-intensive Workloads for Resource Provisioning in Virtualized Environment. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, *5*(4), 1–24. https://doi.org/10.1145/3442696

Mansouri, Y., & Babar, M. A. (2021). A review of edge computing: Features and resource virtualization. *Journal of Parallel and Distributed Computing*, *150*. https://doi.org/10.1016/j.jpdc.2020.12.015

Martinez, H. F., Mondragon, O. H., Rubio, H. A., & Marquez, J. (2022). Computational and Communication Infrastructure Challenges for Resilient Cloud Services. *Computers*, *11*(8), 118. https://doi.org/10.3390/comp uters11080118

McDonough, K., Gao, X., Wang, S., & Wang, H. (2022). Torpedo: A Fuzzing Framework for Discovering Adversarial Container Workloads. *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 402–414. https://doi.org/10.1109/DSN53405.2022.00048

Mwiya, B., Katai, M., Bwalya, J., Kayekesi, M., Kaonga, S., Kasanda, E., Munyonzwe, C., Kaulungombe, B., Sakala, E., Muyenga, A., & Mwenya, D. (2022). Examining the effects of electronic service quality on online banking customer satisfaction: Evidence from Zambia. *Cogent Business and Management*, *9*(1). https://doi.org/10.108 0/23311975.2022 .2143017

Nikdel, Z., & Neville, S. W. (2024). Exploring Emerging Industry Trends for Large-Scale Software System Performance Predictability: The Role of Apache and Xen. *2024 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 1–8. https://doi.org/10.1109/PACRIM61180.2 024.10690225

Pramanik, P. K. D., Pal, S., & Choudhury, P. (2024). Mobile crowd computing: potential, architecture, requirements, challenges, and applications. *Journal of Supercomputing*, *80*(2). https://doi.org/10.1007/s11227-023-05545-0

Ramanathan, S., Kondepu, K., Razo, M., Tacca, M., Valcarenghi, L., & Fumagalli, A. (2021). Live Migration of Virtual Machine and Container Based Mobile Core Network Components: A Comprehensive Study. *IEEE Access*, *9*, 105082–105100. https://doi.org/10.1109/ACCESS.2021.3099370

Rista, A., Ajdari, J., & Zenuni, X. (2020). Cloud Computing Virtualization: A Comprehensive Survey. *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, 462–472. https://doi.org/10.23919/MIPRO48935.2020.9245124

Rista, A., Toti, L., & Xhaferra, E. (2023). Design and Implementation of an Alumni Management System. *2023 4th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*, 1–4. https://doi.org/10.1109/CIEES58940.2023.10378784

Schulz, B., & Annighofer, B. (2022). Evaluation of Adaptive Partitioning and Real-Time Capability for Virtualization With Xen Hypervisor. *IEEE Transactions on Aerospace and Electronic Systems*, *58*(1), 206–217. https://doi.org/10.1109/TAES.2021.3104941

Singh, R., Sukapuram, R., & Chakraborty, S. (2023). A survey of mobility-aware Multi-access Edge Computing: Challenges, use cases and future directions. *Ad Hoc Networks*, *140*, 103044. https://doi.org/10.1016/j.adh oc.2022.103044

Sissodia, R., Rauthan, M. M. S., & Barthwal, V. (2024). Service Level Agreements (SLAs) and their role in establishing trust. In *Analyzing and Mitigating Security Risks in Cloud Computing*. https://doi.org/10.4018/ 979-8-3693-3249-8.ch009

Sota, J. (2022). Technology and Virtual Reality in Teaching - The Challenge of the New Century. *Interdisciplinary Journal of Research and Development*, *9*(3), 11. https://doi.org/10.56345/ijrdv9n302

Stana, A., Kosova, R., & Rista, A. (2024). Benefits of using ICT Tools in Higher Education Institutions in Albania. *2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*, 1–6. https://doi.org/10.1109/CIEES62939.2024.10811268

Stana, A., Toti, L., Kosova, R., & Prodani, F. (2019). FUTURE OF DURRES: SMART CITY AND SMART UNIVERSITY. *European Journal of Engineering and Technology*, *7*(6).

Tabaku, E. (2025). Improving High Availability Services Using KVM Full Virtualization. *European Journal of Computer Science and Information Technology*, *13*(1), 1–15. https://doi.org/10.37745/ejcsit.2013/vol13n1115

Tabaku, E., Duçi, E., & Lazaj, A. (2024). From Physical Stores to Virtual Marketplaces: The Evolution of Shopping. *Interdisciplinary Journal of Research and Development*, *11*(3), 175. https://doi.org/10.56345/ijrdv11n324

Tabaku, E., Shapllo, K., & Bushati, S. (2018, October 27). Multithreading technique improvement in processing Client Server communication by utilizing object oriented language. *2018 UBT International Conference*. https://doi.org/10.33107/ubt-ic.2018.98

The Evolution of Technology in Accounting and Corporate Finance: Implications for Business Adaptation and Competitiveness. (2025). *International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/IRJMETS66318

Viegas, E., Santin, A., Bachtold, J., Segalin, D., Stihler, M., Marcon, A., & Maziero, C. (2021). Enhancing service maintainability by monitoring and auditing SLA in cloud computing. *Cluster Computing*, *24*(3). https://doi.org/10.1007/s10586-020-03209-9

Zolfaghari, R., Sahafi, A., Rahmani, A. M., & Rezaei, R. (2021). Application of virtual machine consolidation in cloud computing systems. *Sustainable Computing: Informatics and Systems*, *30*, 100524. https://doi.org/10.1 016/j.suscom.2 021.100524

Zonta, T., da Costa, C. A., Zeiser, F. A., de Oliveira Ramos, G., Kunst, R., & da Rosa Righi, R. (2022). A predictive maintenance model for optimizing production schedule using deep neural networks. *Journal of Manufacturing Systems*, *62*. https://doi.org/10.1016/j.jmsy.2021.12.013

Appendix:

A (here)

B (here)