Assessing LXC Containers on Raspberry Pi 4B/5 Boards in a Proxmox Virtual Environment

Eric Gamess, Deuntae Winston
MCIS Department, Jacksonville State University, Jacksonville, Alabama, USA

***Abstract*—On one hand, containerization is gaining acceptance as a lightweight virtualization alternative to Virtual Machines (VMs). On the other hand, Single-Board Computers (SBCs) are increasingly used due to their affordability, versatility, low energy consumption, and growing computational power. In this work, extensive experiments were conducted to assess the capabilities and limitations of Linux Containers (LXC) when deployed on a Proxmox Virtual Environment (Proxmox VE) cluster, built with Raspberry Pi (RPi) computers. The clusters consisted of either two Raspberry Pi 4 Model B (RPi 4B) or two Raspberry Pi 5 (RPi 5) with identical characteristics, connected through an Ethernet switch. The experiments aimed to determine: 1) the maximum number of containers that can be run simultaneously when varying their operating system, 2) the maximum number of containers that can be executed in parallel when varying their allocated RAM, 3) the maximum number of containers that can be run concurrently under different SBC memory configurations, 4) the time for container migration, and 5) the network performance between two containers. For storage, SATA SSDs were connected to the RPi 4B boards through their USB ports, while the RPi 5 boards used NVMe SSDs connected via their PCIe interfaces. The cluster formed with RPi 5 boards outperformed the one built with RPi 4B boards, showing significant improvements in the migration experiments. In terms of network performance, the results were similar between containers running on different nodes. However, much larger differences were observed between containers in execution on the same nodes. With this study, the authors aim to assist users and researchers in identifying and selecting the technologies and configurations that best meet the performance requirements of their specific study cases.**

Keywords*—Container; virtualization; Proxmox VE; LXC; single-board computers; Raspberry Pi; performance evaluation

I. Introduction

In recent years, virtualization has become the foundation of modern computing infrastructures. It enables the consolidation of multiple servers onto the same hardware by maximizing its utilization, while minimizing energy consumption and physical space requirements. For many years, Virtual Machines (VMs) were the core of virtualization. However, the emergence of container technologies is reshaping the landscape. VMs require a full guest Operating System (OS) for each instance, whereas containers share the host's kernel while maintaining strong process isolation. Owing to this key architectural difference, containerization offers a more efficient approach with significantly lower overhead in terms of CPU, memory, and storage consumption. As a result, a much larger number of containers, compared to VMs, can be instantiated on the same node. Moreover, containers typically exhibit faster startup times because they do not involve booting an entire guest OS, which is particularly beneficial in environments where services must be frequently created, scaled, or updated.

Proxmox Virtual Environment [1-2] (Proxmox VE) is an open-source platform that integrates virtualization technologies and storage management. Users can achieve full virtualization with Kernel-based Virtual Machine [3-4] (KVM) or create Linux Containers [5-8] (LXC) oriented to lightweight operating-system-level virtualization. That is, the same management framework eases the deployment and administration of both VMs and LXC containers, through robust user interfaces (web-based and CLI), allowing users precise control over computing, memory, storage, and networking resources. Additional advanced features include snapshots, clustering, live migration of VMs, high availability, and backup management. Proxmox VE stands out for its open-source nature, strong community support, active development, and abundant instructional material (comprehensive documentation and numerous tutorials) that assist new users in rapidly becoming proficient with the platform.

Single-Board Computers (SBCs) are complete computing systems built on a single circuit board, integrating a CPU, memory, storage interfaces, and input/output ports. Due to their compact size, low energy consumption, affordability, and versatility, they have been increasingly adopted in IoT projects, education, research, and by hobbyists. Several manufacturers produce SBCs, including the Raspberry Pi Foundation, Hardkernel, BeagleBoard.org Foundation, NVIDIA, Intel, and ASUS. However, Raspberry Pi boards dominate the market, due to a plethora of expansion boards, the wide range of supported OSs, continuous development and improvement made by the manufacturer, and a large community that provides tutorials, project guides, forums, and third-party software.

This study evaluates the performance and limitations of deploying LXC containers on the two most powerful RPi models currently offered by the Raspberry Pi Foundation (RPi 4B and RPi 5), when managed by Proxmox VE. To achieve this goal, the authors set up a testbed consisting of two Raspberry Pi boards with identical specifications, interconnected through a Gigabit Ethernet switch, and combined into a Proxmox VE cluster. A series of comprehensive experiments were conducted to determine: 1) the maximum number of containers that can be run simultaneously when varying their OS, 2) the maximum number of containers that can be executed in parallel when varying their allocated RAM, 3) the maximum number of containers that can be run concurrently under different SBC

memory configurations, 4) the time for container migration, and 5) the network performance between two containers.

As expected, the cluster built with RPi 5 boards significantly outperformed the one based on RPi 4B boards, owing to the upgraded hardware of the SBC. The introduction of a PCIe interface in the RPi 5 is a game-changer, as it enables high-speed access to SSDs. In the container migration experiments, the limited network bandwidth of the RPi boards (1000 Mbps) was the primary bottleneck, particularly for the RPi 5. Through this work, the authors aim to support users and researchers in evaluating and selecting technologies and configurations that optimally fulfill the performance objectives of their specific research contexts.

The remainder of the study is organized as follows: Section II presents relevant peer-reviewed studies conducted in this research area. The testbed and the benchmarking tools employed in this work are introduced in Section III. Section IV conducts and analyzes a series of comprehensive experiments. Finally, Section V concludes the study and outlines directions for future work.

II. Related Work

LXC containers have been evaluated in various scenarios; however, most studies in this area focus on the x86/x64 architecture (e.g., Intel and AMD). For instance, Putri, Munadi, and Negara [9] compared the native performance of several applications with their containerized counterparts using Docker, LXC, and LXD. They assessed services such as web, FTP, and mail within a testbed consisting of an HP 1000 Notebook PC (Intel Celeron Dual-Core CPU 1000M @ 1.80 GHz) and a Dell Inspiron One 2020 (Intel Core i3-3240T @ 2.9 GHz), interconnected through a router. The authors of [10] analyzed several performance metrics under three scenarios: 1) bare metal, 2) an LXC container, and 3) a VM managed by XenServer. The tests employed various benchmarking tools, and the evaluated metrics included CPU performance, RAM speed, storage access time, and the maximum number of requests that could be handled by a web server per second. The testbed consisted of a PC with an Intel Core i5-650 @ 3.20 GHz. In [11], Indukuri evaluated container migrations for LXC and OpenVZ, using a testbed composed of three servers equipped with Intel Xeon E5-2420 v2 @ 2.20 GHz. In [12], the authors analyzed the performance of LXC containers and VMs (using KVM as the hypervisor), under High-Performance Computing (HPC) workloads. Their test environment consisted of two HP EliteDesk 800 G1 (F4K58LT) computers, each equipped with an Intel Core i7-4770 @ 3.4 GHz.

Regarding the evaluation of LXC containers on embedded systems, the authors could only find a few works. Manninen [13] compared several container technologies (Docker, balenaEngine, LXC, and Flatpak) against native performance on an RPi 4B. The experiments included two containers running on the same SBC that exchanged random integer numbers through TCP. The author reported metrics such as memory consumption, CPU utilization, and power consumption. In [14], the authors benchmarked multiple container technologies (Docker, Podman, and LXC/LXD) on three RPi models (RPi 3B, RPi 3B+, and RPi 4B). They conducted experiments to assess processing power, container lifecycle times (creation, start, and run times), and basic network performance between containers. Menshchikov [15] examined the overhead introduced by LXC containers for MIPS-based devices.

Unlike the work presented in this study, none of the previous works contemplated the performance evaluation of LXC containers in RPi 4B and RPi 5 boards, when managed by Proxmox VE.

III. Testbed and Benchmarking Tools

For the experiments, the following SBCs were used: two RPi 4B with 2 GB of RAM, two RPi 4B with 4 GB of RAM, two RPi 4B with 8 GB of RAM, two RPi 5 with 2 GB of RAM, two RPi 5 with 4 GB of RAM, and two RPi 5 with 8 GB of RAM. The specifications of the SBCs are summarized in Table I. As presented in [16], there are two main options for local storage for an RPi 4B: 1) a microSD card inserted in the microSD slot and 2) an SSD connected via one of the USB ports (SATA SSD or NVMe SSD). In contrast, the RPi 5 has three main possibilities: 1) a microSD card inserted in the microSD slot, 2) an SSD connected via one of the USB ports, and 3) an SSD connected through the PCIe x1 interface. As shown in [16], and in order to get the highest possible performance, the authors opted for the fastest available storage technology for each SBC under test. Therefore, each RPi 4B was connected to a SATA SSD via one of its USB 3.0 ports (except for the tests in Section IV D), while each RPi 5 was connected to an NVMe SSD through its PCIe Gen 3 x1 interface (except for the assessments conducted in Section IV D).

TABLE I. Specifications of the RPi 4B and the RPi 5

SoC Type	RPi 4B RPi 5	Broadcom BCM2711 Broadcom BCM2712
Core Type	RPi 4B RPi 5	Quad-core ARM Cortex-A72 @ 1.8 GHz Quad-core ARM Cortex-A76 @ 2.4 GHz
GPU	RPi 4B RPi 5	Broadcom VideoCore VI @ 500 MHz Broadcom VideoCore VII @ 800 MHz
RAM	RPi 4B RPi 5	1, 2, 4, or 8 GB LPDDR4-3200 SDRAM 2, 4, 8, or 16 GB LPDDR4X-4267 SDRAM
microSD	RPi 4B RPi 5	microSD microSD with high-speed SDR104 mode
USB Ports	RPi 4B RPi 5	2 x USB 2.0 ports & 2 x USB 3.0 ports 2 x USB 2.0 ports & 2 x USB 3.0 ports
HDMI Ports	RPi 4B RPi 5	2 x Micro HDMI (up to 4Kp60) 2 x Micro HDMI (up to 4Kp60)
PCIe	RPi 4B RPi 5	No PCIe x1 (Generations 1, 2, and 3)
Ethernet	RPi 4B RPi 5	10/100/1000 Mbps 10/100/1000 Mbps
Wi-Fi	RPi 4B RPi 5	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac
Price	RPi 4B RPi 5	US$35, US$45, US$55, or US$75 US$50, US$60, US$80, or US$120

As depicted in Fig. 1, each experiment involved two identical SBCs (same model and RAM size) connected via an Ethernet switch (Cisco Catalyst 2960X-48TS-L). This Layer-2 switch features 48 10/100/1000 Mbps Ethernet ports with PoE/PoE+ (IEEE 802.3af/802.3at) support and four 1 Gbps Small Form-Factor Pluggable (SFP) uplinks. The latest version of the Raspberry Pi OS Lite and Proxmox VE was installed on each RPi, and a cluster was formed with the two nodes. Clustering provides benefits such as centralized management, VM and container migration, resource optimization, and scalability.

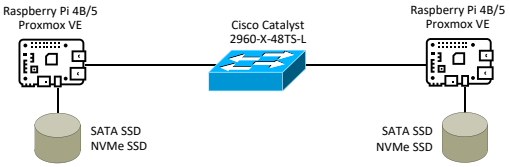

Fig. 1. Testbed for the experiments.

OpenSSL [17-18] is a widely adopted open-source library and toolkit that provides implementations of numerous cryptographic algorithms, as well as the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. OpenSSL includes a benchmarking utility (openssl speed) to evaluate the performance of various cryptographic algorithms, including hashing algorithms (e.g., MD4, MD5, SHA-1, SHA-256, SHA-512, RIPEMD-160), symmetric cipher algorithms (e.g., Blowfish, DES, 3DES, AES, Camellia, SEED, CAST), and public-key algorithms (e.g., RSA, DSA, Ed25519). The utility measures the speed at which a computational system performs cryptographic operations such as hashing, encryption, decryption, and digital signature generation and verification. For hashing and symmetric cipher algorithms, it reports the throughput that corresponds to the number of bytes processed per second. In the case of public-key algorithms, the tool reports two results: the number of signatures generated and verified per second. For the tests of this study, two symmetric encryption algorithms were chosen: 1) 3DES Encrypt-Decrypt-Encrypt (EDE) with three independent DES keys in Electronic Codebook (ECB) mode and 2) AES with a 128-bit key in Cipher Block Chaining (CBC) mode. A cryptographic workload is ideal for container evaluation because it is CPU-intensive, deterministic, and highly sensitive to scheduling and isolation overhead.

iperf3 [19] is an open-source, cross-platform (Linux, Windows, macOS, BSD, and others) network performance measurement tool that operates on the client/server model. It is widely recognized to estimate the maximum achievable UDP, TCP, or SCTP bandwidth under controlled conditions. In this work, iperf3 was chosen to assess TCP throughput. During the execution of a test, the client floods the server with as many TCP messages as possible, which then reports the average TCP throughput received.

sockperf [20] is an open-source network benchmarking tool initially developed by Mellanox for testing and measuring One-Way-Delay (OWD) and throughput for UDP and TCP, using a client/server model. It supports multiple testing modes, such as ping-pong (request–response) and throughput. In the ping-pong mode for TCP, the client sends a TCP payload of a specific size to the server, which immediately returns it. The client then estimates the OWD as half of the measured Round-Trip Time (RTT), that is, the time elapsed between sending the request and receiving the reply, divided by two. This benchmarking utility was selected for the OWD experiments because of its high accuracy, minimal overhead, and widespread adoption in the networking research community.

In this research, unless otherwise stated, all instantiated LXC containers were configured with two cores, 512 MB of memory, a 512 MB swap area, and a 2 GB filesystem.

IV. Experimental Results and Analysis

In this section, several performance evaluation experiments were conducted and analyzed.

A. Maximum Number of Containers When Varying the Operating System

Proxmox has a built-in library of LXC templates that includes Alpine Linux, Arch Linux, CentOS, Debian, Fedora, Rocky Linux, Ubuntu, AlmaLinux, Devuan, Gentoo Linux, openEuler, and openSUSE. With these templates, users can deploy LXC containers with their favorite OS in just a few seconds. Due to their popularity in the community, Debian, Fedora, Alpine, and Ubuntu were chosen for this study. At the level of Debian, two versions were tested: Buster (Debian 10.X) and Bookworm (Debian 12.X). The latter was selected as it was the latest release of Debian at the time of writing this study. Since the authors wanted to illustrate how cryptographic support in the processor can make a huge difference in performance, Debian Buster was also chosen. It is the last release of Debian with a version of OpenSSL that does not leverage hardware cryptography when available in the processor.

Fig. 4 depicts the results of assessing the maximum number of LXC containers that can be run simultaneously on an RPi 4B with 8 GB of RAM, when varying the OS. In each container, the OpenSSL benchmarking tool was executed for 3DES EDE in ECB mode (des-ede3) as specified in the command of Fig. 2, using a 64-byte buffer (option -bytes 64), when running two operations in parallel (option -multi 2) to test multi-core performance.

```
openssl speed -seconds $duration -bytes 64  \
        -multi 2 des-ede3 > /tmp/results.txt
```

Fig. 2. Command executed in each container to evaluate the performance of 3DES EDE in ECB mode.

Fig. 4 consists of seventeen groups of five bars, where the groups correspond to a different number of containers (1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 53, 54, and 63). Within each group, the five bars represent Debian 12, Debian 10, Fedora 38, Alpine 3.18, and Ubuntu 22.04. As can be seen in the figure, the last three groups (53, 54, and 63 containers) have fewer than five bars. That indicates that for the missing OSs, the RPi 4B did not have enough resources to run the specified number of containers and became unstable or crashed. Overall, the performance decreased as the number of containers increased due to contention for processing resources. Across the OSs, performance was generally similar. The maximum number of containers achieved was 54, 50, 50,

63, and 50 for Debian 12, Debian 10, Fedora 38, Alpine 3.18, and Ubuntu 22.04, respectively. Unsurprisingly, it was possible to run more Alpine containers than the other Linux versions, since it was specifically designed to be small and resource-efficient.

The same experiments were repeated on an RPi 5 with 8 GB of RAM, and the results are presented in Fig. 5. As an updated version of the RPi 4B, the RPi 5 delivers better performance. For instance, the throughput of Debian 12 increased from 27.5 B/s on the RPi 4B (see Fig. 4) to 47.4 B/s on the RPi 5 (see Fig. 5) for a single container. It should be noted that neither the RPi 4B nor the RPi 5 provides dedicated hardware support for DES or 3DES. Consequently, the execution of these algorithms relies solely on software implementations, which are considerably slower than hardware-accelerated alternatives.

Fig. 6 and Fig. 7 present the results of assessing the maximum number of LXC containers that can be run simultaneously on an RPi 4B and an RPi 5, both with 8 GB of RAM, when varying the OS. However, in this case, the OpenSSL benchmarking tool was executed in each container for AES-128 in CBC mode (aes-128-cbc) as specified by the command shown in Fig. 3, using a 64-byte buffer (option -bytes 64), when running two operations in parallel (option -multi 2) to evaluate multi-core performance:

```
openssl speed -seconds $duration -bytes 64   \
        -multi 2 aes-128-cbc > /tmp/results.txt
```

Fig. 3. Command executed in each container to evaluate the performance of AES-128 in CBC mode.

As in the previous experiments (see Fig. 4 and Fig. 5), the performance decreases as the number of containers increases. However, this time, for the same OS, a substantial performance difference can be observed between the RPi 4B (see Fig. 6) and the RPi 5 (see Fig. 7). For example, the throughput of Debian 12 increased from 96.3 B/s on the RPi 4B to 2693.7 B/s on the RPi 5 for a single container. That is, this time the RPi 5 significantly outperformed the RPi 4B for all OSs except Debian 10. This improvement is due to the hardware support of the SoC of the RPi 5 for AES (in the RPi 4B, AES is implemented in software). However, the OpenSSL implementation in Debian 10 does not utilize the AES hardware support, resulting in considerably slower performance for this OS on the RPi 5.

The performance tests on the pre-selected OSs showed no significant differences among them. In terms of the maximum number of containers that could be run simultaneously, Alpine 3.18 had a slight advantage. Nevertheless, Debian 12 was chosen for the remaining experiments because it was the second-best OS in this regard and offers several additional benefits: a large and active community, an extensive software repository, exceptional stability, and a reputation for prioritizing reliability over rapid release cycles. Moreover, Debian serves as the base for numerous other popular distributions, including Ubuntu, Kali Linux, ParrotOS, TurnKey Linux, Linux Mint Debian Edition, AntiX, Devuan, PureOS, Raspberry Pi OS, and Proxmox VE, which underscores its proven architecture and enduring relevance within the Linux ecosystem.

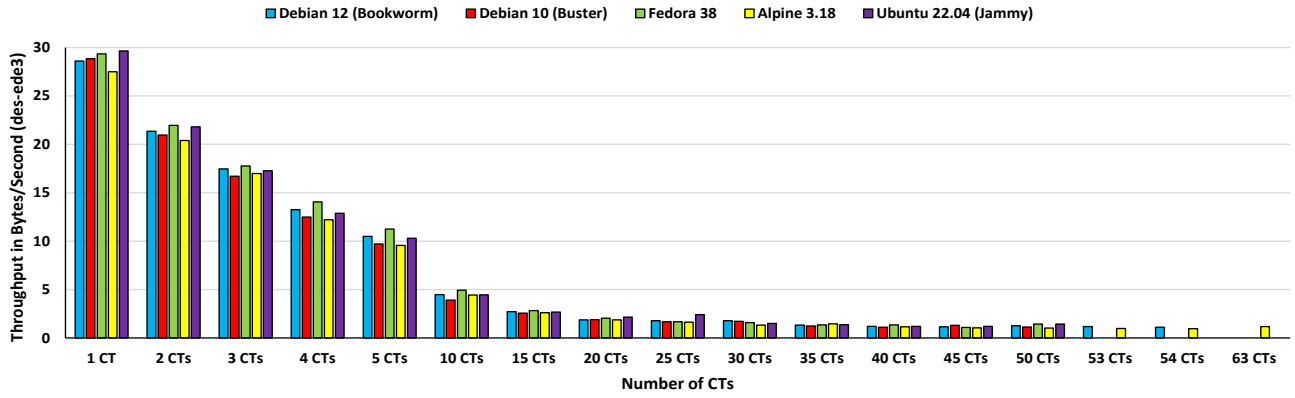

Fig. 4. Benchmarking 3DES EDE in ECB mode on the RPi 4B for several operating systems.

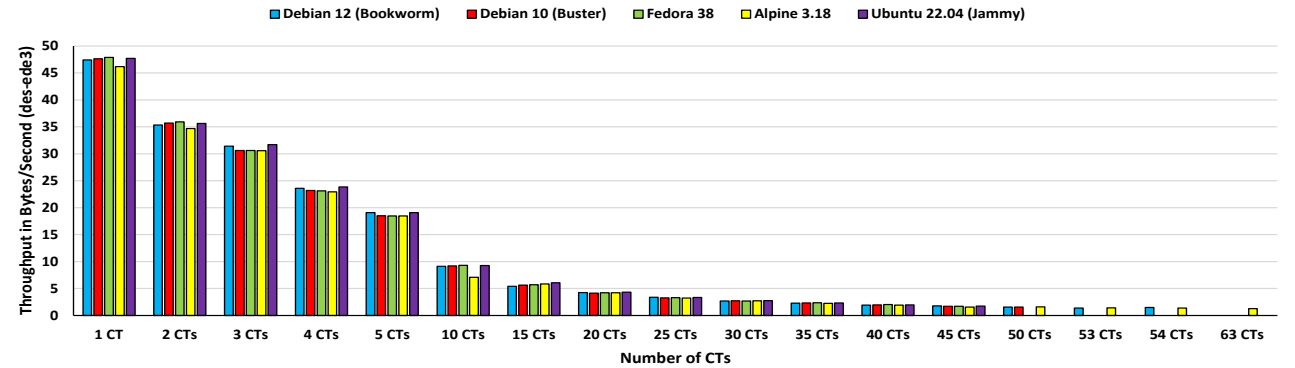

Fig. 5. Benchmarking 3DES EDE in ECB mode on the RPi 5 for several operating systems.

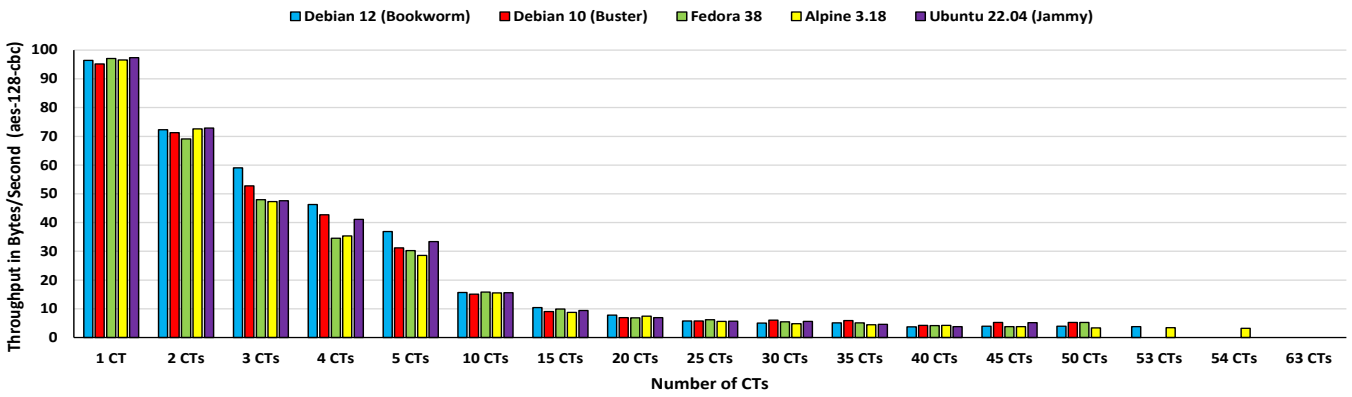

Fig. 6. Benchmarking AES-128 in CBC mode on the RPi 4B for several operating systems.

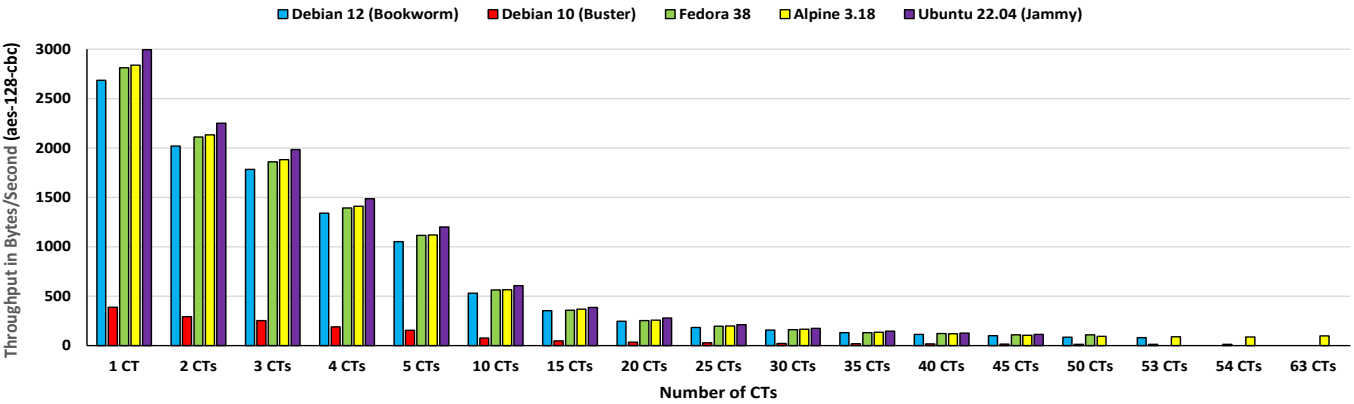

Fig. 7. Benchmarking AES-128 in CBC mode on the RPi 5 for several operating systems.

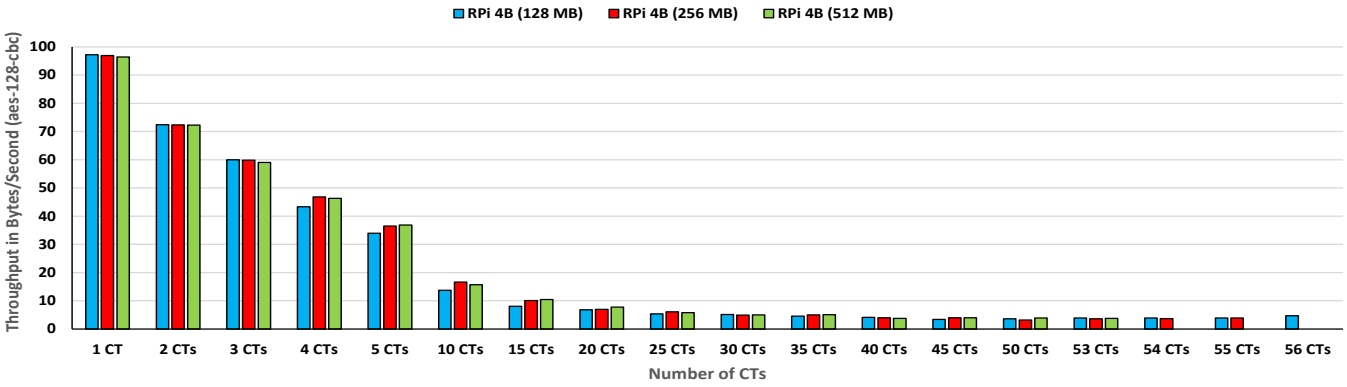

Fig. 8. Benchmarking AES-128 in CBC mode on the RPi 4B for different RAM sizes of the containers.

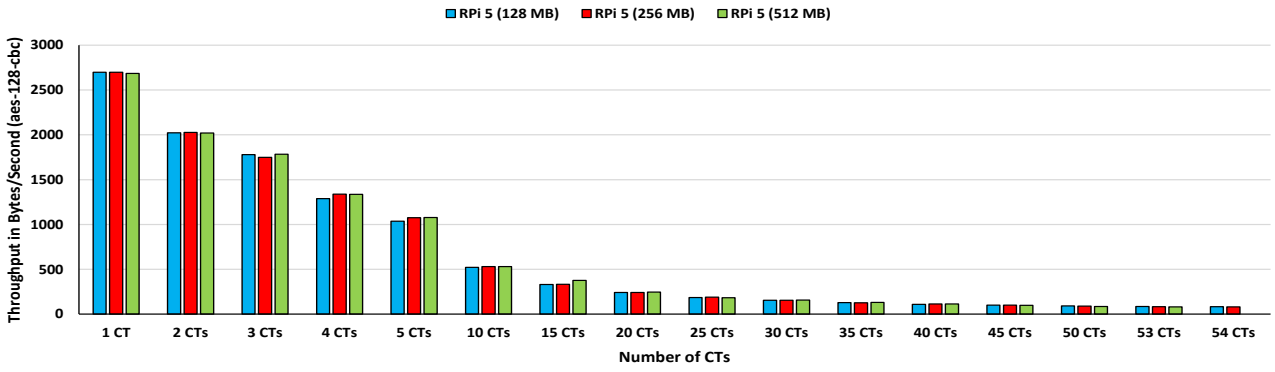

Fig. 9. Benchmarking AES-128 in CBC mode on the RPi 5 for different RAM sizes of the containers.

B. Maximum Number of Containers When Varying their Allocated RAM

In this section, experiments were conducted to analyze the impact of container RAM size on the overall system performance. Similar to the experiments in Section IV.A, Debian 12 containers were deployed on an RPi 4B and an RPi 5 (both equipped with 8 GB of RAM) and executed the command shown in Fig. 3, to assess AES-128 in CBC mode. Three memory configurations were tested for the containers: 128 MB, 256 MB, and 512 MB. Fig. 8 and Fig. 9 present the maximum number of containers that could run simultaneously.

As can be observed in both figures, container performance decreased as their number increased, since more containers had to compete for the available computational resources. In this test, the container RAM size had only a minor impact on performance, as the workload was CPU-intensive and made relatively light usage of memory. For the RPi 4B (see Fig. 8), the maximum number of containers that could be executed simultaneously is 56, 55, and 53 for RAM sizes of 128 MB, 256 MB, and 512 MB, respectively. At the level of the RPi 5 (see Fig. 9), the corresponding values were 54, 54, and 53. It is also noteworthy that the RPi 5 achieved a substantial performance improvement over the RPi 4B, with an increase by a factor of 22 to 41. This improvement is the result of the updated hardware of the RPi 5, particularly its dedicated hardware support for AES, which is absent in the RPi 4B.

C. Maximum Number of Containers Under Different SBC Memory Configurations

As shown in Table I, the RPi 4B and RPi 5 are available with different RAM configurations. This section analyzes the impact of this parameter on the maximum number of Debian 12 containers that could be executed simultaneously, as well as their performance when running AES-128 in CBC mode (i.e., all containers were executing the command of Fig. 3). Three RAM configurations for the SBCs were tested: 2 GB, 4 GB, and 8 GB. The results for the RPi 4B are presented in Fig. 10. As observed, the RAM configuration of the SBC had a minimal effect on performance. However, the maximum number of containers that could be executed in parallel varied considerably, reaching 6, 21, and 53 for the 2 GB, 4 GB, and 8 GB configurations, respectively.

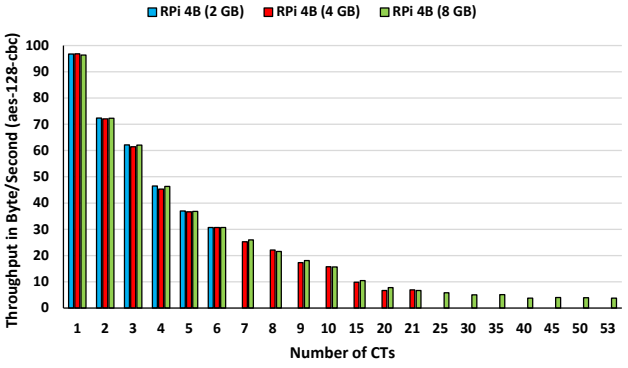

Fig. 10. Benchmarking AES-128 in CBC mode on the RPi 4B under different RAM configurations of the SBCs.

Fig. 11 depicts the results for the RPi 5. Similar to the RPi 4B (see Fig. 10), the RAM configuration of the RPi 5 had a negligible impact on container performance. Regarding the number of containers that could be run simultaneously, the values reached 4, 21, and 53 for the 2 GB, 4 GB, and 8 GB configurations, respectively.

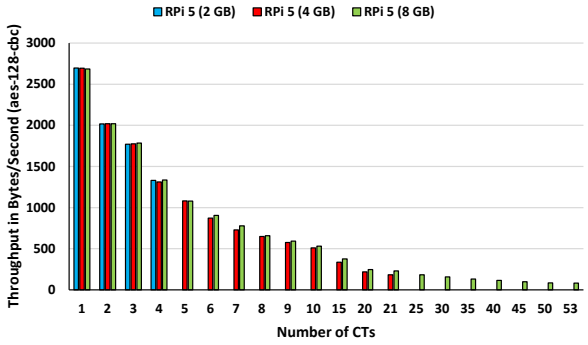

Fig. 11. Benchmarking AES-128 in CBC Mode on the RPi 5 under different RAM configurations of the SBCs.

D. Time for Container Migration

In data centers, container migration is paramount for achieving both load balancing and high availability. It allows containers to be relocated from overloaded nodes to underutilized ones, ensuring optimal utilization of computational resources. Moreover, migration enhances fault tolerance by minimizing service downtime when a host requires maintenance, experiences a failure, or is being decommissioned. The current version of Proxmox VE does not support live migration of containers. Live migration refers to transferring a running container from one host to another without stopping the container. Therefore, this section reports experiments on offline migration, in which the container is stopped, moved, and then restarted.

Unlike the other experiments realized in this study, the tests performed in this section additionally involved connecting the SATA SSD through a USB 2.0 port of the RPi 4B, and the NVMe SSD via the PCIe Gen 2 x1 interface of the RPi 5. Because migration is storage-intensive, this addition at the level of the RPi 4B covered scenarios in which both USB 3.0 ports were already in use. For the RPi 5, the PCIe Gen 2 setup was included to address cases where users prefer to avoid potential stability risks, as the Raspberry Pi Foundation officially supports only PCIe Gen 1 and Gen 2, even though Gen 3 can be enabled experimentally with possible reliability issues.

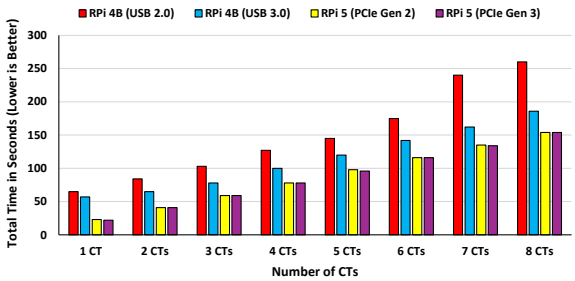

Fig. 12. Container migration times on the clusters built with RPi 4B and RPi 5 boards.

Fig. 12 presents the results of migrating up to eight containers concurrently. For the RPi 4B, total migration times differ significantly for the SATA SSD depending on whether it was connected via USB 2.0 or USB 3.0, reflecting their respective maximum theoretical transfer speeds of 480 Mbps and 5000 Mbps. In contrast, for the RPi 5, the difference between PCIe Gen 2 (4000 Mbps) and PCIe Gen 3 (close to 8000 Mbps) is negligible, as the network bandwidth (1000 Mbps) constrained migration performance in this scenario.

E. Network Performance

This section delves into assessing the network performance between two containers under two scenarios: 1) containers running on the same node of the cluster and 2) containers running on different nodes of the cluster. To benchmark the network, the authors selected to evaluate: 1) the maximum TCP throughput between two containers and 2) the TCP OWD between two containers.

As described in Section III, iperf3 was used to measure the maximum TCP throughput. Fig. 13 presents the commands that were executed on the server and client, with line numbers added for reference. Line 01 started the server (option -s), which waited for client requests. The client command is shown in Line 02, where each test ran for 20 seconds (option -t 20), and the TCP payload size was varied (option -l <size>).

```
01: iperf3 -s -f m
02: iperf3 -c <ip-server> -f m -t 20 -l <size>
```

Fig. 13. Command executed in the containers to evaluate the maximum TCP throughput.

Fig. 14 depicts the results obtained for the cluster built with RPi 4B boards, using TCP payload sizes of 8, 128, 1024, 2048, 4096, 8192, 16384, 32768, and 65000 bytes. For experiments conducted between two containers running on different nodes, the TCP throughput increases with the payload size and eventually plateaus at 936 Mbps (the difference from the theoretical 1000 Mbps is due to Ethernet, IP, and TCP overheads). In contrast, for the experiments between two containers running on the same node, the throughput soars with increasing payload size, reaching 8470 Mbps for a 65000-byte payload. It is worth noting that in the latter case, the network is entirely virtual and data transfer between containers is implemented through memory and/or disk copies.

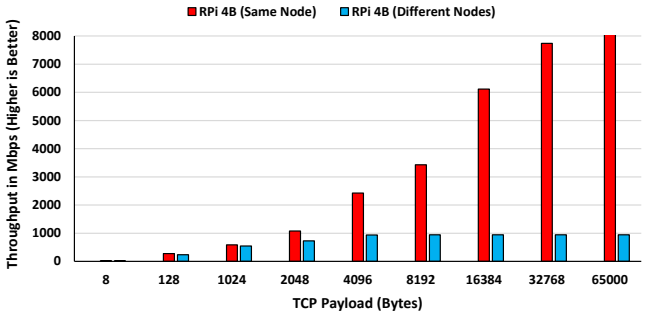

Fig. 14. TCP throughput between containers running in the cluster built with RPi 4B.

Fig. 15 illustrates the results achieved for the cluster built with RPi 5 boards. Similar to the ones with RPi 4B boards (see Fig. 14), the TCP throughput plateaus at 940 Mbps for the experiments between containers running on different nodes. For the tests done between containers in execution on the same node, the cluster based on RPi 5 computers significantly outperformed the one based on RPi 4B computers, achieving a TCP throughput of 27600 Mbps for a 65000-byte payload.

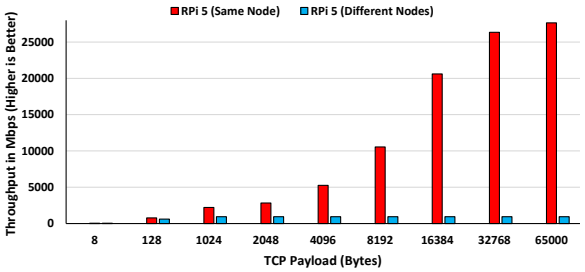

Fig. 15. TCP throughput between containers running in the cluster built with RPi 5.

As specified in Section III, sockperf was used to measure the TCP OWD. Fig. 16 shows the commands that were executed on the server (Line 01) and the client (Line 02). Each test lasted 20 seconds (option `-t 20` in the client) and the TCP payload size was varied (option `-m <size>` in the client).

```
01: sockperf server --tcp
02: sockperf ping-pong -i <ip-server> -t 20  \
    --mps max –tcp -m <size>
```

Fig. 16. Command executed in the containers to evaluate the TCP OWD.

Fig. 17 and Fig. 18 show the TCP OWD for the clusters built with RPi 4B and RPi 5 computers, respectively. As expected, the OWD increases with the payload size. The differences between the experiments conducted on the same and on different nodes are substantial, and are also primarily attributed to the network virtualization.

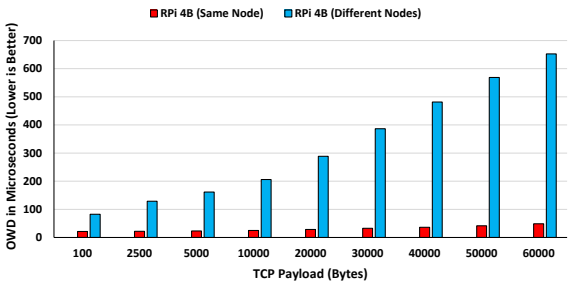

Fig. 17. TCP OWD between containers running in the cluster built with RPi 4B.

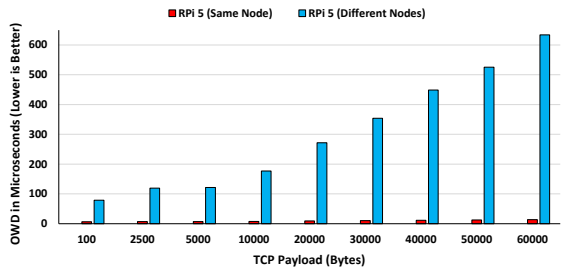

Fig. 18. TCP OWD between containers running in the cluster built with RPi 5.

V. Conclusion and Future Work

This work presented an experimental evaluation of LXC containers deployed on a Proxmox VE cluster built with either two RPi 4B boards or two RPi 5 boards. Several experiments were conducted to analyze the influence of different parameters on container performance, including operating system choice, allocated memory, SBC RAM configuration, migration time, and network behavior.

The results showed that both SBC models can efficiently host multiple LXC containers, although the RPi 5 consistently outperformed the RPi 4B across all tests. The introduction of a PCIe interface in the RPi 5 allowed the use of NVMe SSDs, which considerably improved storage access and migration performance. While the maximum number of containers that could run concurrently depended mainly on the total available RAM of the SBCs, CPU architecture and hardware acceleration for cryptographic operations (such as AES) had a decisive impact on encryption/decryption throughput.

Regarding networking, containers running on the same node achieved significantly higher TCP throughput than those communicating across nodes, highlighting the overhead introduced by physical network interfaces. The measured migration times confirmed that storage access and network bandwidth are the dominant limiting factors in such operations.

Overall, the study demonstrates that Proxmox VE, when combined with modern SBCs such as the RPi 5, provides a practical and cost-effective platform for lightweight virtualization. These findings can assist users and researchers in selecting appropriate SBC hardware and container configurations to meet specific performance requirements.

As future work, the authors plan to include additional SBCs in the study and investigate how different network technologies affect the performance, using multigigabit Ethernet (2.5 Gbps and 5 Gbps) and network bonding.

Acknowledgment

We are grateful to “Faculty Commons” and the “College of Arts, Humanities, and Sciences” at Jacksonville State University for partially funding this study.

References

[1] R.-P. Kleinert, “Proxmox VE 8 Practical Guide: Information, Tips, and Tricks for Proxmox Beginners and Advanced Users”, Independently Published, August 2024.

[2] A. A. Bekroundjo, “Mastering Proxmox VE 8: A Comprehensive Guide to Enterprise Virtualization”, Independently Published, April 2025.

[3] A. A. Bekroundjo, “Mastering KVM Virtualization: The Complete Guide to Enterprise Infrastructure on Ubuntu 24.04”, Independently Published, June 2025.

[4] R. Johnson, “KVM Virtualization Essentials: Definitive Reference for Developers and Engineers”, HiTeX Press, June 2025.

[5] R. Johnson, “Linux Container Essentials with LXC: Definitive Reference for Developers and Engineers”, HiTeX Press, June 2025.

[6] K. Ivanov, “Containerization with LXC: Get Acquainted with the World of LXC”, Packt Publishing, February 2017.

[7] R. Johnson, “Linux Container Management with LXD: Definitive Reference for Developers and Engineers”, HiTeX Press, June 2025.

[8] S. Kumaran S., “Practical LXC and LXD: Linux Containers for Virtualization and Orchestration”, 1st Edition, Apress, September 2017.

[9] A. R. Putri, R. Munadi, and R. M. Negara, “Performance Analysis of Multi Services on Container Docker, LXC, and LXD”, Bulletin of Electrical Engineering and Informatics, vol. 9, no. 5, October 2020, pp. 2008–2011, doi: 10.11591/eei.v9i4.1953.

[10] S. R. Poojara, V. B. Ghule, M. N. Birje, and N. V. Dharwadkar, “Performance Analysis of Linux Container and Hypervisor for Application Deployment on Clouds”, 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS 2018), Belgaum, India, 2018, pp. 24–29, doi: 10.1109/CTEMS.2018.8769146.

[11] P. S. V. Indukuri, “Performance Comparison of Linux Containers (LXC) and OpenVZ during Live Migration: An Experiment”, Master’s Thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 2016.

[12] D. Beserra, E. D. Moreno, P. T. Endo, J. Barreto, D. Sadok, and S. Fernandes, “Performance Analysis of LXC for HPC Environments”, 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2015), Santa Catarina, Brazil, 2015, pp. 358–363, doi: 10.1109/CISIS.2015.53.

[13] H. Manninen, “Evaluation of Container Technologies for an Embedded Linux Device”, Master’s Thesis, July 2020, Aalto University, Espoo, Finland.

[14] E. Gamess and M. Parajuli, “Containers Unleashed: A Raspberry Pi Showdown Between Docker, Podman, and LXC/LXD”, IEEE SoutheastCon 2024, Atlanta, GA, USA, 2024, pp. 34–43, doi: 10.1109/SoutheastCon52093.2024.10500266.

[15] M. Menshchikov, “Design and Testing of LXC-based Virtualization System for Resource-constrained MIPS Devices”, 2018 Third Conference on Software Engineering and Information Management (SEIM 2018), Saint Petersburg, Russia, April 2018.

[16] E. Gamess and G. Hester, “Using a Raspberry Pi as a Network-Attached Storage Device: Performance and Limitations”, 2025 ACM Southeast Conference (ACMSE 2025), April 24–26, 2025, Cape Girardeau, MO, USA, pp. 67–77, doi: 10.1145/3696673.3723073.

[17] R. S. Patil, “Practical Network Security with OpenSSL: Master Cryptography and OpenSSL Techniques for Secure Communications, PKI, and Hardware Integration in Real-World Applications”, Orange Education Pvt Ltd, April 2025.

[18] B. D. Hooper, “Cyber Security Development with OpenSSL and Cryptography: Applied Cryptography and OpenSSL for Cyber Security”, Independently published, June 2025.

[19] ESnet, “iperf3: A TCP, UDP, and SCTP Network Bandwidth Measurement Tool”, Lawrence Berkeley National Laboratory, https://software.es.net/iperf/.

[20] Mellanox, “sockperf: Network Benchmarking Utility”, https://github.com/Mellanox/sockperf.