Performance Evaluation of Different Raspberry Pi Models for a Broad Spectrum of Interests

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

Sergio Hernandez
Information Security
Citibank, New York
USA

***Abstract*—Now-a-days, Single Board Computers (SBCs), especially Raspberry Pi (RPi) devices, are extensively used due to their low cost, efficient use of energy, and successful implementation in a wide range of applications; therefore, evaluating their performance is critical to better understand the applicability of RPis to solve problems in different areas of knowledge. This paper describes a comparative and experimental study regarding the performance of five different models of the RPi family (RPi Zero W, RPi Zero 2 W, RPi 3B, RPi 3B+, and RPi 4B) in several scenarios and with different configurations. To conduct our multiple experiments on RPis, we used a self-developed and other existing open-source benchmarking tools allowing us to perform tests that mimic real-world needs, assessing important factors including CPU frequency and temperature during stressful activities, processor performance when executing CPU-intensive processes such as audio and file compressions as well as cryptographic operations, memory and microSD storage performance when executing read and write operations, TCP throughput in different WiFi bands, and TCP latency to send a specific payload from a source to a destination. Our experimental results showed that the RPi 4B significantly outperformed the other SBCs tested. In addition, our research indicated that the RPi Zero 2 W overclocked, RPi 3B, and RPi 3B+ had similar performance. Finally, the RPi Zero 2 W showed a much higher capacity than its predecessor, the RPi Zero W, and seems to be a perfect replacement when upgrading, since they have the same form factor and are physically interchangeable. With this study, we aim to guide researchers and hobbyists in selecting adequate RPis for their projects.**

***Keywords*—Performance evaluation; benchmarks; raspberry pi; single board computer**

I. Introduction

One of the most popular Single Board Computer (SBC) is the Raspberry Pi (RPi), with a vast online community of users around the world. Build on open-source principles and motivated by the non-profit incentive to increase global access to computing and solve a variety of real-world challenges using digital technology, these low-cost SBCs bring together external hardware, sensors and controller interfaces, with user-friendly programming capabilities, high connectivity, and desktop functionality [1].

RPis are being employed in a broad range of projects across diverse topics and research fields, including the Internet of Things (IoT) that has become widely used in recent years in an extensive range of applications from smart cities and industries to water monitoring [2][3]. Examples of successful uses of RPis can be found in the field of Artificial Intelligence (AI) and Machine Learning (ML), where researchers have been highlighting not only a good performance but also a low energy consumption [4][5]. Additionally, the usage of RPis has spread to other relevant areas such as cybersecurity, energy, health, education, to name a few [6].

Considering the frequent usage of RPis in applications, it is essential to deeply analyze how they behave and perform under different conditions for a given period of time, to better understand their capabilities and limitations. In this paper, we carried out a comparative and experimental study of five different models of the RPi family (RPi Zero W, RPi Zero 2 W, RPi 3B, RPi 3B+, and RPi 4B). To do so, we conducted several experiments by using a benchmarking tool that we developed as well as existing open-source benchmarking tools to evaluate the performance of RPis in terms of processor frequency and temperature, CPU use level, RAM and microSD access performance, TCP throughput and latency, among others. We think the results of this study might guide scientists and hobbyists in selecting adequate models of RPi for their projects, according to their budget and performance requirements.

The rest of the paper is structured as follows. Section II discusses the related work. The description of the testbed environment is done in Section III. Section IV reports and discusses the results of our performance evaluation of the different RPi models. Finally, Section V concludes the paper and gives directions for future work.

II. Related Work

Due to the popularity and constant evolution of the Raspberry Pi models, several works about the performance evaluation of these devices have been performed. We reviewed the literature in order to examine and understand areas, methods, and available tools to assess their performance.

Morabito [7] performed an assessment of container-based technologies running on top of a Raspberry Pi 2 Model B using different types of workloads to challenge a specific subsystem of the hardware under test. This study aimed to evaluate the use of Docker containers in constrained environments, providing a detailed performance analysis. Experiments in a testbed were conducted, and metrics such as CPU execution time, power consumption, memory speed, network bandwidth, and protocol overhead for MySQL and Apache were reported.

Other works [8][9] examined the performance of Intrusion Detection Systems (IDS) running on RPis. Kyaw, Chen, and Joseph [8] presented the results of an experiment comparing two open-source IDSs (Snort and Bro) on a Raspberry Pi 2 Model B, with the main goal of determining their performance, efficiency, and efficacy for use in computer network environments, where cost is a determining factor. On the other hand, Aspernäs and Simonsson [9] compared two RPis (Raspberry Pi 1 Model B+ and Raspberry Pi 2 Model B) while examining the traffic for intrusion detection in a home environment.

The authors of [10] carried out a performance analysis of SNMP [11] agents running in three different RPis (Raspberry Pi Zero W, Raspberry Pi 3 Model B, and Raspberry Pi 3 Model B+). Numerous experiments varying different parameters, requests, versions, and security models of SNMP were performed.

Another related work was consulted in [12], where a performance evaluation of RESTful frameworks on two Raspberry Pi 1 Model B was carried out. In this study, experiments involving combinations of factors such as device CPU frequency and message size with different configurations or levels were conducted by comparing two web services frameworks (Axis2 and CXF) to understand better not only the energy consumption, but also the relationship between performance and energy consumption in RPis.

Guamán, Ninahualpa, Salazar, and Guarda [13] did a comparative study between the MQTT [14][15] and the CoAP [16][17] protocols for IoT in an IEEE 802.11 environments, using a Raspberry Pi 3 Model B+. For the analysis of network parameters, traffic injection tests were carried out with specific tools, using different bandwidths and data sizes. Another work in this direction was done in [3], where the authors presented performance measurements of the Raspberry Pi Zero W working as an IoT gateway between local sensors and a public MQTT [14][15] broker running in the cloud. The experimental results demonstrated its performance using the following metrics: CPU utilization, temperature, as well as the rate of received MQTT messages under different levels of Quality of Service (QoS).

Other studies in the area are focused on benchmarking cryptographic algorithms on RPis. For instance, Hawthorne, Kapralos, Blaine, and Matthews [18] compared the performance of three computing systems for three leading cryptographic algorithms (AES, Twofish, and Serpent). The three computing systems considered were: (1) a cluster of Raspberry Pi 3 Model B+, (2) a power-efficient next unit of computing (NUC), and (3) a mid-range desktop (MRD). The metrics reported by this work included encryption/decryption throughput and power consumption. Similarly, the study in [19] aims to analyze the performance of symmetric encryption algorithms (DES and AES) within the PHP programming language using RPis. The authors reported parameters such as the time and memory consumption to encrypt/decrypt messages.

Although multiple studies on performance evaluation have been performed, they mainly analyze how specific technologies (e.g., containers) behave when run on certain RPi models. Thus, parameters that were evaluated are intrinsically related to a particular environment and do not allow a broad understanding of how the main RPi subsystems respond to different conditions, based on CPU workload, I/O requirement, network traffic, amongst others. Since the potential of the RPi has not been broadly analyzed, there is a gap for those who want to have a more general view of the performance of an RPi, for scenarios that were not covered by the actual specialized literature.

Unlike previous works, our paper considers five different models of RPis, and our assessment covers a broad spectrum of interests. Moreover, the Raspberry Pi Zero 2 W was released at the end of October 2021, and no other scientific work to evaluate its performance was found at the time of performing this study. Therefore, with this paper, we aim to guide scientists and hobbyists in their selection of an adequate model of Raspberry Pi according to their budget and performance requirements.

III. Description of the Testbed Environment

A. Models of Raspberry Pi used in the Experiments

For our assessment, we had the following Raspberry Pi SBCs: two Raspberry Pi Zero W, two Raspberry Pi Zero 2 W, two Raspberry Pi 3 Model B, one Raspberry Pi 3 Model B+, and one Raspberry Pi 4 Model B (8 GB of RAM). Some of their technical specifications are presented next:

- Raspberry Pi Zero W (RPi Zero W): It is based on a 32-bit Broadcom BCM2835 single-core ARM1176JZF-S SoC @ 1.0 GHz, 512 MB of RAM, one 2.4 GHz IEEE 802.11b/g/n WiFi interface, one micro USB On-The-Go port, one mini HDMI connector, and one microSD card slot [20].
- Raspberry Pi Zero 2 W (RPi Zero 2 W): It is based on an RP3A0-AU, which consists of the integration of a 64-bit Broadcom BCM2710A1 quad-core Cortex-A53 @ 1.0 GHz and 512 MB of RAM, in a single chip. It also has one 2.4 GHz IEEE 802.11b/g/n WiFi interface, one micro USB On-The-Go port, one mini HDMI connector, and one microSD card slot [21]. It can be easily overclocked to 1.3 GHz with an adequate heat sink.
- Raspberry Pi 3 Model B (RPi 3B): It is based on a 64-bit Broadcom BCM2837 quad-core Cortex-A53 SoC @ 1.2 GHz, 1 GB of RAM, one 10/100 Mbps Ethernet interface, one 2.4 GHz IEEE 802.11b/g/n WiFi interface, four USB 2.0 ports, one full-size HDMI connector, and one microSD card slot [22].
- Raspberry Pi 3 Model B+ (RPi 3B+): It is based on a 64-bit Broadcom BCM2837B0 quad-core Cortex-A53 SoC @ 1.4 GHz, 1 GB of RAM, one Gigabit Ethernet interface over USB 2.0 (maximum throughput 300 Mbps), one dual-band 2.4 GHz and 5 GHz IEEE 802.11b/g/n/ac WiFi interface, four USB 2.0 ports, one full-size HDMI connector, and one microSD card slot [23].

- Raspberry Pi 4 Model B (RPi 4B): It is based on a 64-bit Broadcom BCM2711 quad-core Cortex-A72 SoC @ 1.8 GHz, 1/2/4/8 GB of RAM, one Gigabit Ethernet interface, one dual-band 2.4 GHz and 5 GHz IEEE 802.11b/g/n/ac WiFi interface, two USB 2.0 ports, two USB 3.0 ports, two micro-HDMI connectors, and one microSD card slot [24].

In all the SBCs, we inserted a 64 GB SanDisk Extreme microSDXC UHS-I Memory Card (SDSQXA2-064G-GN6MA) with the operating system preinstalled. It is considered as one of the fastest microSD cards of the market, with up to 160 MB/s and 60 MB/s for the reading and writing speeds, respectively. Also, unless otherwise stated, all the experiments were carried out with no cooling solution for the RPi Zero W, RPi Zero 2 W, RPi 3B, and RPi 3B+. However, for most of the experiments, we chose to control the temperature of the RPi 4B with a small fan since its temperature can dramatically increase, in contrast to the other RPis that we selected for our study.

B. Operating Systems for Raspberry Pi

Several operating systems are available for Raspberry Pi. We opted for the most popular one developed by the Raspberry Pi Foundation, and known as Raspberry Pi OS (32-bit version). It is based on Debian Bullseye. The last version was released in October 2021. Three options of this operating system are available: (1) Raspberry Pi OS Lite, (2) Raspberry Pi OS with Desktop, and (3) Raspberry Pi OS with Desktop and Recommended Software. The "Lite" option is a minimal image consisting of 493 packages without an X-window manager. Hence, it runs fast and is more suitable for a server environment. The "Desktop" option is a superset of the "Lite" option, with a total of 1324 packages. It is more oriented to end-users since it includes Openbox as the window manager and LXDE as the desktop environment. The "Desktop and Recommended Software" option consists of 1944 packages and has all the "Desktop" option features, but also includes an office productivity suite (LibreOffice) and additional supports for developers (Erlang, Node.js, Ruby, Java, Wolfram, Apache Ant, BlueJ, Firebird, and Greenfoot). We chose the "Lite" option for all our experiments since our performance evaluation is more targeted towards a server installation in headless mode.

There is a beta version of Raspberry Pi OS for 64-bit architecture. Hence, it does not work with the RPi Zero W since it is equipped with a 32-bit processor. The Raspberry Pi Foundation released its last version in October 2021, and we could not make it works with the RPi Zero 2 W.

C. Testbed Used for the Network Experiments

We assessed the performance of the networking system of the different RPis. To do so, we utilized the testbed of Fig. 1. It consisted of two network devices connected through a wireless router. In some experiments, the network devices were two identical RPis. In other experiments, one network device was an RPi, while the other one was a PC. The two network devices were placed 4 meters from the wireless router, with no obstacles between them. For the wireless router, we used a NETGEAR AC1200 Smart WiFi Router R6220. It had the following characteristics: an 880 MHz MediaTek processor width two radio bands (IEEE 802.11b/g/n in the 2.4 GHz band and IEEE 802.11a/n/ac in the 5 GHz band), 128 MB of flash, 128 MB of RAM, and five 10/100/1000 Mbps Ethernet ports (one WAN and four LANs).

The PC had the following specifications: Dell OptiPlex 3030 AIO, with a 64-bit Intel quad-core i3-4130 CPU at 3.4 GHz, 16 GB of RAM, a 512 GB SSD, a 1 Gbps Ethernet NIC, and an Intel Wireless 7260 Network Adapter (dual-band WiFi adapter with support to IEEE 802.11a/b/g/n/ac). Debian amd64 11.1 (codename "Bullseye") was installed as the operating system.

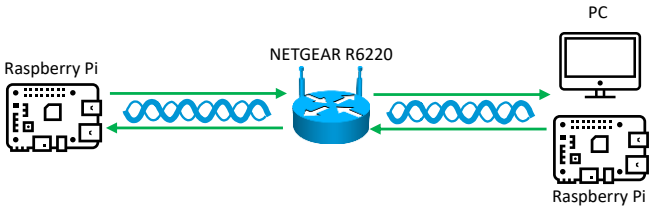

Fig. 1. Testbed for the Network Experiments.

IV. Performance Results and Analysis

In this section, we made a performance evaluation of the considered RPis (RPi Zero W, RPi Zero 2 W, RPi 3B, RPi 3B+, and RPi 4B) in several scenarios, reporting different parameters. Each experiment was executed several times, and the reported results are an average of the repeated experimental runs. By repeating and averaging, we ensure the consistency of our empirical findings.

A. Evaluation of the Temperature with Stressberry Test

The processor of an RPi can overheat if it does not have enough cooling. This overheating problem is more frequent in powerful CPUs, such as the one of the RPi 4B. When necessary, CPU throttling (also known as dynamic frequency scaling) will occur to decrease the electrical energy being consumed, and in turn, to reduce the heat generation. Small heat sinks, specific cases, fans, and other solutions can be utilized to cool down the CPU of an RPi.

The Stressberry test [25] can be used to reveal if the CPU of an RPi can run at maximum load in its case/environment without overheating, and being forced to slow down. It has four phases. In the first phase, Stressberry waits until having a stable temperature. To do so, it lets the CPU idle and just takes a temperature sample every 60 seconds. This initial phase ends when two consecutive samples (previous and current temperatures) are the same. It is noted that no temperature and frequency samples are stored in the resulting file during the first phase. In the second phase, the test lets the CPU idle for 150 seconds, storing samples of the base temperature and base frequency in the resulting file every two seconds. Then, in the third phase, all the cores of the CPU are stressed with a maximum load for 300 seconds. The variation of the temperature and frequency are also saved in the resulting file during this phase. In the final phase (fourth phase), the test lets the CPU idle for another 150 seconds, to have an insight on how fast it can cool down. Here also, the variation of the temperature and frequency are recorded in the resulting file. The entire process takes 600 seconds (10 minutes). When the

test ends, the resulting data (temperature and frequency) that were stored in the resulting file can be processed and plotted.

Fig. 2 shows the instructions that we used to install and run Stressberry. Python 3.x is required, and its version was verified with Line 01. Lines 02 and 03 allow the installation of the dependencies and Stressberry, respectively. The version of Stressberry was obtained in Line 04 (in our case, version 0.3.3), while Line 05 displays some help about the software usage. Stressberry was run with the instruction of Line 06, where option `-i` specifies the idle time in seconds before (phase 2) and after (phase 4) the stress portion (phase 3), while option `-d` indicates the stress test duration in seconds (phase 3). Line 07 generates a graphical representation of the experiment for the temperature, while Line 08 does it for both the temperature and frequency.

```
01: python --version
02: apt-get install stress python3-pip libatlas-base-dev \
 libopenjp2-7-dev
03: pip3 install stressberry
04: stressberry-run -v
05: stressberry-run -h
06: stressberry-run -i 150 -d 300 resFile.dat
07: stressberry-plot resFile.dat -o temp.png --not-transparent
08: stressberry-plot resFile.dat -o both.png --not-transparent -f
```

Fig. 2. Installation and Execution of Stressberry.

Fig. 3 depicts the results that we obtained by running the Stressberry test on the RPi Zero W, RPi Zero 2 W, RPi 3B, RPi 3B+, and RPi 4B. The curves can be divided into three parts: (1) a 150-second idle portion for phase 2, (2) a 300-second stress activity for phase 3, and (3) a 150-second idle portion for phase 4. When running Stressberry without specifying the number of cores, the tools will activate all of them. The RPi Zero W just has one core, and its maximum temperature went barely over 45°C during the stress activity. All the other RPis have four cores, that were activated by Stressberry in this experiment. According to this test, the RPi 3B+ has the highest baseline temperature, while the RPi 3B has the highest temperature during the stress period. It is noted that all the tests of Fig. 3 correspond to naked RPis (no cases), and without any cooling solutions. That is, the small fan was not used for the RPi 4B in this experiment.

The goal of Fig. 4 is to show how a cooling solution can significantly improve the temperature of the CPU. This experiment was conducted with an RPi 4B. The first curve (in orange) corresponds to running Stressberry without any additional heat control system, while a small fan was used for the second curve (in black). The minimum difference is greater than 10°C, and in the best case, it is almost 25°C. It is worth mentioning that the results will significantly vary with the chosen cooling solution (heat sink, fan, a combination of heat sink and fan, specific cases, etc.) and its size.

Fig. 5 aims to determine the impact on the temperature of an RPi, when overclocked or not. The experiment was conducted by running Stressberry on an RPi Zero 2 W. The figure has four curves, two for the RPi not overclocked (temperature and CPU frequency) and two for the RPi overclocked at 1300 MHz (temperature and CPU frequency). In the initial 150-second idle portion, the temperature was the same since the CPU frequency mainly stayed at 600 MHz for both cases. In the 300-second portion of stress, the CPU frequency was steady at 1000 MHz for the RPi not overclocked. However, when overclocked, the CPU frequency started at 1300 MHz and went down to 1000 MHz in the intent to control the rising temperatures. In the final 150-second idle portion, the CPU frequency mainly stayed at 600 MHz for both cases, allowing the temperatures to fall down toward the baseline temperature.

Fig. 6 shows the effect on the temperature of a RPi, when varying the number of cores. The experiment was carried out by running Stressberry on an RPi Zero 2 W not overclocked, activating 1, 2, 3, and 4 cores, respectively. As expected, when under stress, the temperatures get higher with the increasing number of cores.

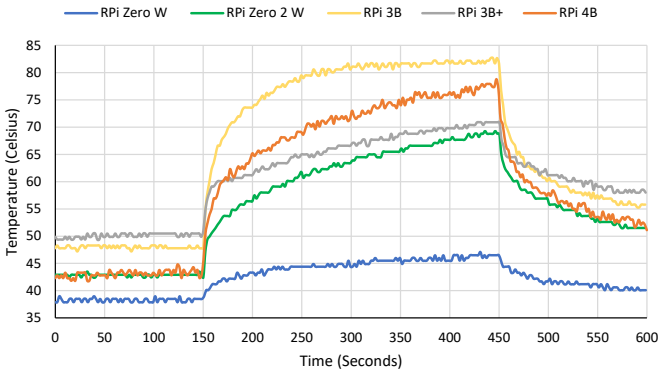

Fig. 3. Results of Stressberry for Different RPi Models.

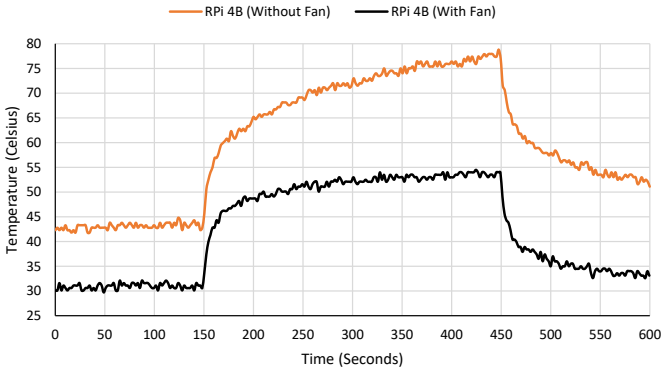

Fig. 4. Results of Stressberry for an RPi 4B with/without Fan.

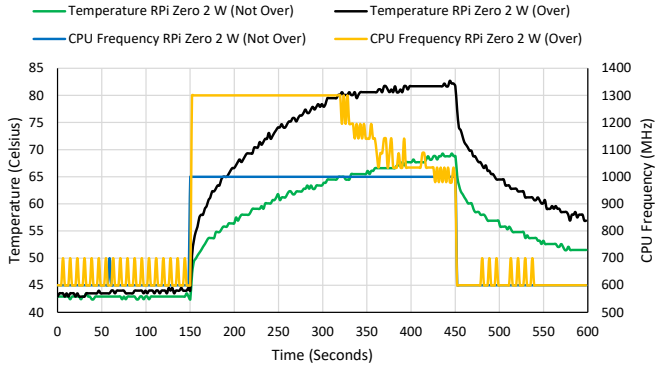

Fig. 5. Results of Stressberry for the RPi Zero 2 W when Overclocked and when Not.

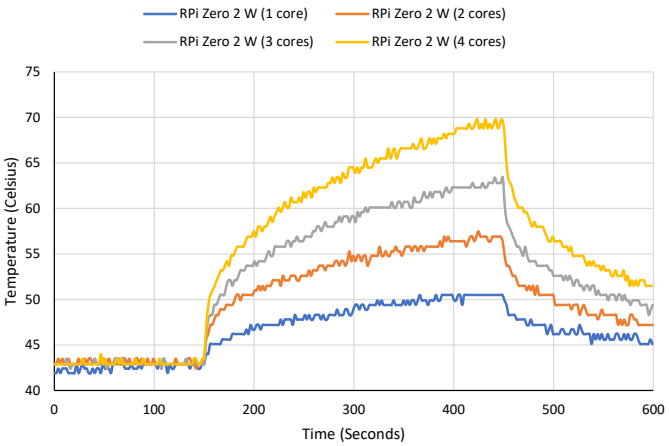

Fig. 6. Results of Stressberry for the RPi Zero 2 W when Varying the Number of Cores.

B. CPU Performance with 7-Zip

The 7-Zip archiving tool [26][27] can be used to pack and compress files into archives, as well as to extract the files from archive formats such as ZIP or 7z. It also has a benchmarking tool built inside it, that assesses the power of a CPU through the LZMA [28] (Lempel–Ziv–Markov chain Algorithm) compression and decompression. The tool reports how fast a CPU processes the compression and decompression instructions over dummy data, displaying the results in MIPS (Million Instructions Per Second). Fig. 7 shows the instructions that we used to install and execute the 7-Zip benchmark. Line 01 installs the tool, while the manual is consulted in Line 02. Lines 03-05 run the benchmark for 1, 2, and 4 threads, respectively. Since the RPi Zero W only has one core, Lines 04-05 were not executed on this RPi.

```
01: apt-get install p7zip
02: man 7zr
03: 7zr b -mmt1
04: 7zr b -mmt2
05: 7zr b -mmt4
```

Fig. 7. Installation and Execution of the Benchmark of 7-Zip.

Fig. 8 and 9 depict the results obtained for the CPU assessment with 7-Zip for compression and decompression, respectively. The dictionary size was 2^{23} = 8 MB. There are two results for the RPi Zero 2 W: one without overclocking, and the other with overclocking the device at 1.3 GHz. It is significantly noticeable how the RPi 4B outperformed all the other RPis. Overclocking the RPi Zero 2 W boots its CPU performance. The RPi Zero 2 W overclocked, RPi 3B, and RPi 3B+ have similar performance.

C. CPU Performance with Sysbench

Sysbench [29] is a scriptable multi-threaded benchmark tool based on LuaJIT. Sysbench has several tests (`cpu`, `memory`, `fileio`, `threads`, and `mutex`) for the CPU performance, memory speed, file I/O access, threads subsystem performance, and mutex performance, respectively.

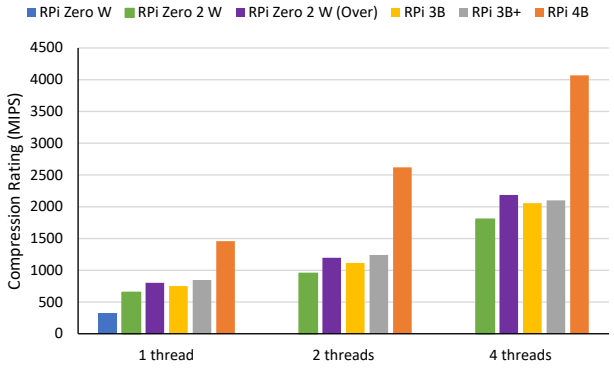

Fig. 8. Compression Rating with 7-Zip for Different RPi Models.

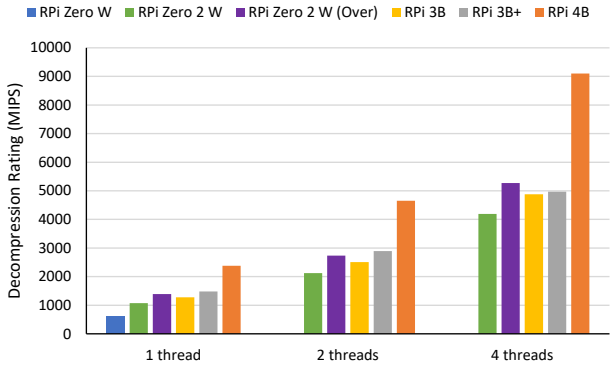

Fig. 9. Decompression Rating with 7-Zip for Different RPi Models.

In this experiment, we evaluated the CPU performance of the different RPis with Sysbench. The test consists in generating random numbers and verifying if they are prime or not, by doing standard divisions of any selected number by all integers between 2 and the square root of this number. If any division gives a remainder of 0, Sysbench starts over by generating a new random number and trying again. The results are reported in events/sec. Fig. 10 gives the instructions that we used to install are run the CPU performance test. Sysbench was installed with Line 01, and the manual was consulted in Lines 02-03. Lines 04-06 run the CPU performance test for 1, 2, and 4 threads, respectively. We limited the total execution time to 20 seconds, and the randomly generated numbers to be tested were inferior to 20,000, as specified by the options of the commands.

```
01: apt-get install sysbench
02: sysbench --help
03: sysbench cpu help
04: sysbench cpu --threads=1 --time=20 --cpu-max-prime=20000 run
05: sysbench cpu --threads=2 --time=20 --cpu-max-prime=20000 run
06: sysbench cpu --threads=4 --time=20 --cpu-max-prime=20000 run
```

Fig. 10. Installation and Execution of Sysbench to Assess the CPU Performance.

Fig. 11 depicts the results that we obtained for this test. It is noted that the RPi Zero W has a result only for one thread. This test confirms the results obtained in Section IV.B with 7-Zip, where the RPi 4B dramatically outperformed the other RPis. Overclocking the RPi Zero 2 W improved its performance. Moreover, the RPi Zero 2 W overclocked, RPi 3B, and RPi 3B+ have a similar performance.

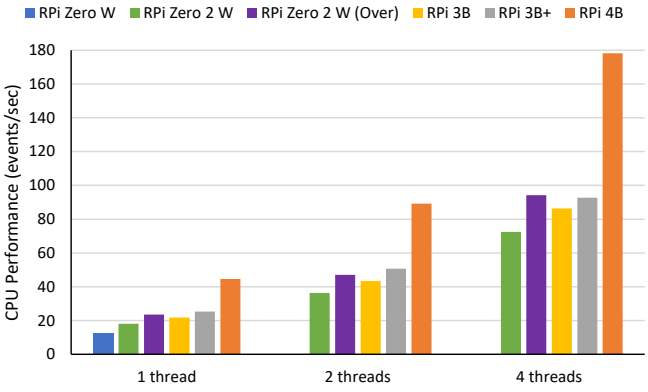

Fig. 11. CPU Performance with Sysbench for Different RPi Models.

D. Memory Speed with Geekbench and STREAM

We initially utilized Sysbench [29] to benchmark the read and write access performance of the memory (RAM) of the RPis. However, we did not get consistent results. Similar problems were reported in [30]. Hence, we opted to use Geekbench [31] and STREAM [32][33].

Geekbench [31] is a cross-platform benchmark program (Windows, macOS, Linux i386/amd64, Android, iOS, etc.) that reports performance related to the integer arithmetic, floating-point arithmetic, and memory. It is a commercial product from Primate Labs; however, a limited version can be downloaded and used for free. At the level of the memory assessment, the older versions of Geekbench [34] has four convenient tests: (1) "Read Sequential" that loads values from memory into registers, (2) "Write Sequential" that stores values from registers into memory, (3) "Stdlib Write" that writes a constant value to a block of memory using functions from the C Standard Library (`memset`), and (4) "Stdlib Copy" that copies values from one block of memory to another using functions from the C Standard Library (`memcpy`). Fig. 12 shows the results that we obtained for "Read Sequential", "Write Sequential", "Stdlib Write", and "Stdlib Copy", respectively, in MB/sec. The RPi Zero W has the poorest performance. The experiment seems to indicate that the RPi Zero 2 W overclocked, RPi 3B, and RPi 3B+ have a similar memory access, for both reading and writing. The RPi 4B significantly outperformed the other devices under test.

Fig. 12. Results of the Memory Performance with Geekbench for Different RPi Models.

STREAM [32][33] is a synthetic benchmark designed to measure sustainable memory bandwidth and the corresponding computation rate for four simple vector kernels: Copy, Scale, Add, and Triad. Copy just transfers data from one memory location to another, i.e., copies it (A[i] = B[i]). Scale takes the value of the first location and multiplies it with a certain constant, before storing it in a second place, i.e., scales it (A[i] = m*B[i]). Add reads data from two different locations, adds them up and writes the result to a third place (A[i] = B[i] + C[i]). Triad reads data from a first location, scales it, then adds data from a second one and writes to a third place (A[i] = m*B[i] + C[i]). Fig. 13 displays the steps that we followed to download, compile, and run STREAM. Notice that STREAM can be compiled with or without OpenMP [35]–[37] support. Lines 01-02 installed Git (a revision control system) and cloned the STREAM repository, respectively. In Line 04, we compiled `stream.c` and specified that the elements of the unidimensional arrays are floating-point numbers in double precisions (`double`), the size of the arrays is 10 MB, each kernel should be run ten times, and activated support for OpenMP. Line 05 is optional. If not specified, one thread is activated for the RPi Zero W, and four threads for the other devices under test.

```
01: apt-get install git
02: git clone https://github.com/jeffhammond/STREAM.git
03: cd STREAM
04: gcc -o stream-bin -O3 -march=native -DSTREAM_TYPE=double \
 -DSTREAM_ARRAY_SIZE=10000000 -DNTIMES=10 -fopenmp stream.c
05: export OMP_NUM_THREADS=<NUM_CPU_CORES>
06: ./stream-bin
```

Fig. 13. Download, Compilation, and Execution of STREAM to Assess the Memory Performance.

Fig. 14 depicts the assessment of the different RPis with STREAM, using one thread for the RPi Zero W and four threads for the other RPis. It is noticeable how the RPi 4B outperformed the other SBCs. The RPi Zero W is much slower. The other devices under test performed similarly.

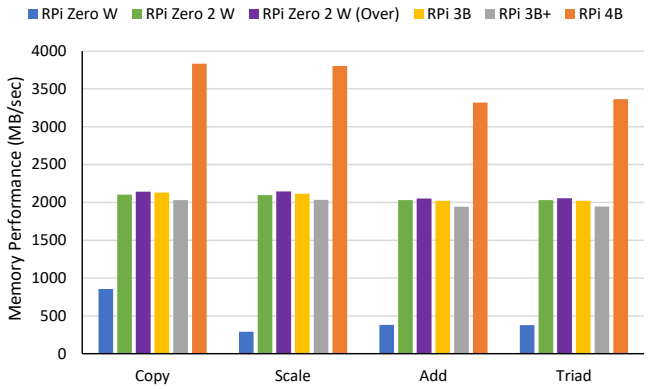

Fig. 14. Results of the Memory Performance with STREAM for Different RPi Models.

E. Sequential Access Performance on the MicroSD Card with Sysbench

The goal of this experiment was to evaluate the sequential read and write access performance on the microSD card for the different RPis with Sysbench [29], for one thread. Fig. 15 shows the instructions that we used to run the file access

performance test, for sequential readings. The manual was consulted in Lines 01-02. Line 03 generated 32 files of size 64 MiB each, for a total of 2 GiB. Those files are the ones from which Sysbench performed the read operations during the test. In Line 04, the test was run for a block size of 1 kiB, for 20 seconds. This line was executed several times, where the block size was varied to 2, 8, 32, 64, 128, 256, 512, 1024, and 2048 kiB, respectively. Finally, Line 05 erased all the 32 temporary files created in Line 03.

```
01: sysbench --help
02: sysbench fileio help
03: sysbench fileio --file-num=32 --file-total-size=2G prepare
04: sysbench fileio --file-num=32 --file-total-size=2G \
 --file-extra-flags=direct --file-test-mode=seqrd \
 --file-block-size=1k --time=20 run
05: sysbench fileio --file-num=32 --file-total-size=2G cleanup
```

Fig. 15. Execution of Sysbench to Evaluate the Sequential Read Assess Performance on the microSD Card.

Fig. 16 presents the instructions used to benchmark the sequential write access performance on the microSD card, for the different RPis with Sysbench, for one thread. In this case, there is no need to pre-generate temporary files since Sysbench will not read, but write. In Line 01 the test was run for a block size of 1 kiB, for 20 seconds, writing files (up to 32 files with a maximum size of 64 MiB each). This line was executed several times, where the block size was varied to 2, 8, 32, 64, 128, 256, 512, 1024, and 2048 kiB, respectively. Finally, Line 02 removed all the files created in Line 01.

```
01: sysbench fileio --file-num=32 --file-total-size=2G \
 --file-extra-flags=direct --file-test-mode=seqwr \
 --file-block-size=1k --time=20 run
02: sysbench fileio --file-num=32 --file-total-size=2G cleanup
```

Fig. 16. Execution of Sysbench to Evaluate the Sequential Write Assess Performance on the microSD Card.

Fig. 17 and 18 depict the results that we obtained for the sequential read and write access performance, respectively, for the different RPis. The RPi Zero W has the poorest performance, while the RPi 4B has the best one. The RPi Zero 2 W (overclocked or not), RPi 3B, and RPi 3B+ have a comparable performance.

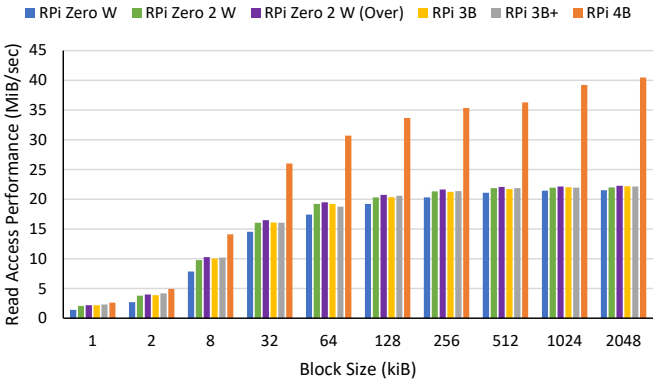

Fig. 17. Sequential Read Access Performance on the microSD Card with Sysbench for Different RPi Models.

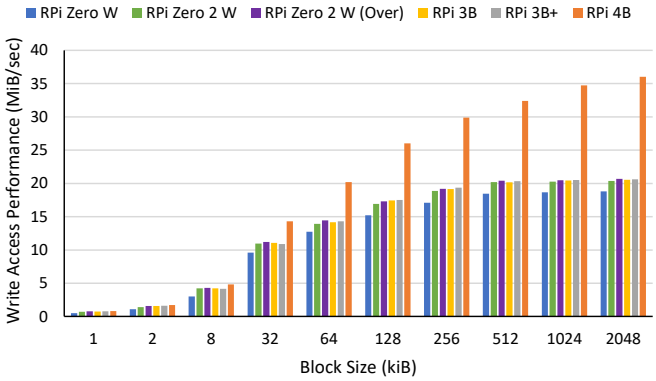

Fig. 18. Sequential Write Access Performance on the microSD Card with Sysbench for Different RPi Models.

F. Sequential Write Access Performance on the microSD Card with dd

This experiment aimed to evaluate the sequential write access performance on the microSD card for the different RPis with "dd", for one thread. The command "dd" is a standard Unix/Linux tool to convert and copy files. Fig. 19 shows the instructions that we used to run the sequential write access performance test with "dd". The manual was consulted in Line 01. In Line 02, the test was run. Here, the tool created a 512 kiB file by making 512 write operations, each one with a block size of 1 kiB. This last line was executed several times, where the block size was changed to 2, 8, 32, 64, 128, 256, 512, 1024, and 2048 kiB, respectively, creating files of 1, 4, 16, 32, 64, 128, 256, 512, and 1024 MiB, respectively.

```
01: man dd
02: dd if=/dev/zero of=/home/pi/test bs=1k count=512 oflag=direct
```

Fig. 19. Execution of "dd" to Evaluate the Sequential Write Assess Performance on the microSD Card.

Fig. 20 depicts the results that we obtained for the different RPis. It is very similar to the performance results of Fig. 18.

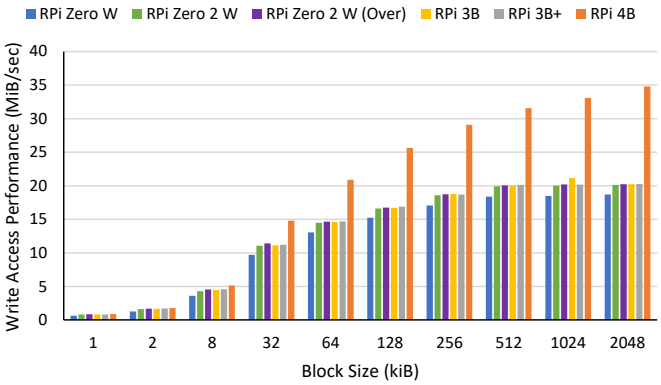

Fig. 20. Sequential Write Access Performance on the microSD Card with "dd" for Different RPi Models.

G. Audio Conversion Performance with Phoronix Test Suite

Phoronix Test Suite [38] (PTS) is a free and open-source framework for conducting automated performance tests. It is multiplatform, and the default version has more than 600 individual test profiles and more than 200 test suites, covering

a wide range of applications such as audio format conversions, encryption and decryption algorithms, compression and decompression algorithms, timed compilation of widely-used open-source software, memory access performance, file access performance, etc. The framework is designed to be extensible so that new test profiles and suites can be easily added.

We used PTS [38] to evaluate the performance of the RPis when converting/encoding sample WAV files to MP3 (encode-mp3), sample WAV files to FLAC (encode-flac), sample WAV files to Monkey's Audio APE (encode-ape), sample WAV files to WavPack (encode-wavpack), and sample WAV files to Opus (encode-opus). Fig. 21 shows the instructions that we used to install and run the audio conversion performance tests with PTS. First, the dependencies were installed in Line 01. In Line 02, PTS was cloned from GitHub. Lines 04-08 run the different conversion tests.

```
01: apt-get install php-cli php-xml git
02: git clone https://github.com/phoronix-test-suite/\
 phoronix-test-suite.git
03: cd phoronix-test-suite
04: ./phoronix-test-suite benchmark encode-mp3
05: ./phoronix-test-suite benchmark encode-flac
06: ./phoronix-test-suite benchmark encode-ape
07: ./phoronix-test-suite benchmark encode-wavpack
08: ./phoronix-test-suite benchmark encode-opus
```

Fig. 21. Installation and Execution of Phoronix Test Suite to Evaluate the Performance of Audio Conversions.

Fig. 22 depicts the audio conversion performance results in seconds. The lower is the conversion time, the better/faster is the RPi. The RPi Zero W had the longest conversion time, while the RPi 4B had the shortest. The RPi Zero 2 W overclocked, RPi 3B, and RPi 3B+ had similar performance.

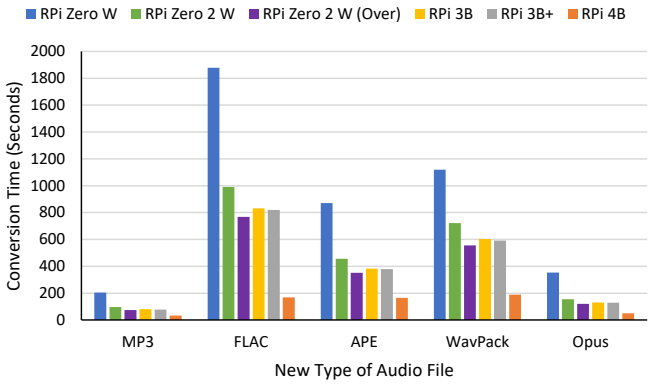

Fig. 22. Audio Conversion Performance with PTS for Different RPi Models.

H. Other Performance Evaluation with Phoronix Test Suite

We also utilized PTS [38] for other performance evaluations. The "`openssl`" test of PTS assesses (1) the number of digital signatures that can be performed per second and (2) the number of verifications of digital signatures that can be processed per second, using RSA with 4096-bit keys. Table I has the results that we obtained for the different RPis. In this test also, the RPi 4B had a much better performance. The RPi Zero 2 W overclocked, RPi 3B, and RPi 3B+ had comparable performances.

TABLE I. SIGNING AND VERIFYING PERFORMANCE WITH PTS

Test	Signing (sign/sec)	Verifying (verify/sec)
RPi Zero W	2.6	176.1
RPi Zero 2 W	39.4	2834.7
RPi Zero 2 W (Over)	47.2	3380.2
RPi 3B	45.9	3329.3
RPi 3B+	46.4	3337.5
RPi 4B	120.0	9126.4

The total time required to compile an open-source software is considered a good benchmark to measure the performance of computers. PTS [38] offers several profiles to get a timed compilation of well-accepted software. Table II shows the results that we obtained to compile ImageMagick [39][40] (an application to create, edit, compose, or convert digital images) and MPlayer [41] (a movie player), using PTS. The big difference between the RPi Zero W and the other RPis can be partially explained by the options used with the "`make`" utility. PTS used "`make -j1`" for the RPi Zero W, and "`make -j4`" for the other RPis. This option specifies the number of jobs (commands) that can be run simultaneously during the compilation process.

TABLE II. TOTAL TIME REQUIRED TO COMPILE OPEN-SOURCE SOFTWARE WITH PTS

Test	ImageMagick (Time in Seconds)	MPlayer (Time in Seconds)
RPi Zero W	7350.6	12990.6
RPi Zero 2 W	825.7	1390.5
RPi Zero 2 W (Over)	758.2	1196.0
RPi 3B	773.9	1241.1
RPi 3B+	762.2	1204.4
RPi 4B	388.4	507.6

I. TCP Throughput with Iperf

The goal of this experiment is to determine the maximum TCP throughput that can be obtained between two end-devices connected as specified in Fig. 1, where at least one RPi is utilized as an end-device. To do so, we used Iperf [42][43], a free, open-source command-line tool for network performance measurement between two network devices. It is based on the client/server model, and reports parameters such as the throughput, delay jitter, and packet loss. Iperf v2.0.14a is available as a pre-compiled package from the Raspberry Pi OS repositories. However, this version has limitations. Hence, we downloaded and installed a newer version (Iperf v2.1.n) for our experiments. We used Iperf to determine the "maximum" TCP throughput between the client and the server. When using this test, the client tries to overwhelm the server by creating a unidirectional TCP flow (from the client to the server) and sending as many segments as allowed by the TCP control mechanism (congestion window). At the end of the experiment, by default 10 seconds, the TCP throughput is displayed.

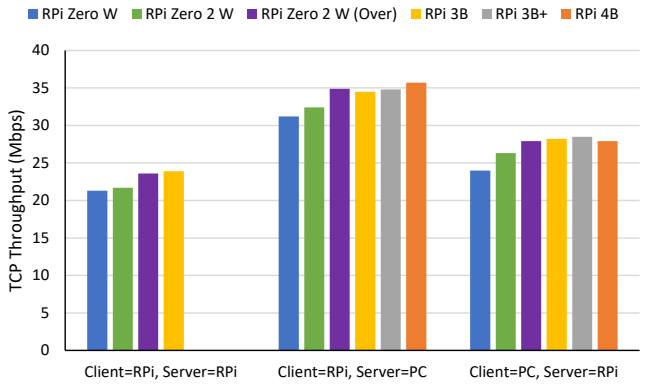

Fig. 23. TCP Throughput with Iperf when Varying the Roles of the RPis for Different RPi Models.

Fig. 23 depicts the TCP throughput that we obtained in three different scenarios: (1) both the client and server were RPis of the same models, (2) the client was an RPi while the server was a PC, and (3) the client was a PC while the server was an RPi. The specifications of the PC were given in Section III.C. It is worth noting that in the first group of bars, there are only four bars, corresponding to two RPi Zero W (blue bar), two RPi Zero 2 W not overclocked (green bar), two RPi Zero 2 W overclocked (purple bar), and two RPi 3B (yellow bar). There are no bars for the RPi 3B+ and the RPi 4B, since we only had one of each of these SBCs. For this experiment, we set up the wireless router to use a maximum bandwidth of 145 Mbps in the 2.4 GHz band. It is noted that all the RPis had a bitrate that capped out at 72.2 Mbps in this band.

The best performance is obtained when the server is a PC (second group of bars), while the worst corresponds to the case of using two RPis of the same models (first group of bars) for the client and server. Since the PC that we used is more potent than the RPis, when used as a server, it is much faster to discard the received segments from the client, and then to reopen its TCP congestion window, allowing the client to send the TCP segments at a higher rate, resulting in a better performance for this case.

The RPi 3B+ and RPi 4B are dual-bands. Hence, we also made some performance evaluations of the TCP throughput in the 5 GHz band, by setting up the wireless router to use a maximum bandwidth of 867 Mbps in the 5 GHz band. It is worth mentioning that the two RPis (RPi 3B+ and RPi 4B) had a bitrate that capped out at 433.3 Mbps in this band. Table III has the results that we obtained. By changing from the 2.4 GHz to the 5 GHz band, the improvement in throughput is noticeable.

TABLE III. TCP THROUGHPUT WITH IPERF IN DIFFERENT BANDS FOR THE RPI 3B+ AND RPI 4B

Test	2.4 GHz	5 GHz
Client=RPi 3B+, Server=PC	34.80 Mbps	102.10 Mbps
Client=PC, Server=RPi 3B+	28.50 Mbps	81.40 Mbps
Client=RPi 4B, Server=PC	35.70 Mbps	104.25 Mbps
Client=PC, Server=RPi 4B	27.90 Mbps	85.70 Mbps

J. TCP Latency with our Own Benchmark

The aim of this experiment is to determine the TCP latency to send a specific TCP payload from a source to a destination. To accomplish this objective, we wrote our own benchmark and used the testbed of Fig. 1, with an RPi as the source and a PC as a destination. The specifications of the PC were given in Section III.C. For this experiment, we set up the wireless router to use a maximum bandwidth of 145 Mbps in the 2.4 GHz band. Fig. 24 shows the results that we got for the TCP latency when varying the payload to be transmitted through TCP from 100 to 10,000 bytes. The RPi Zero W has the longest latency, followed by the RPi Zero 2 W not overclocked. The RPi Zero 2 W overclocked, RPi 3B, RPi 3B+, and RPi 4B had a similar performance.

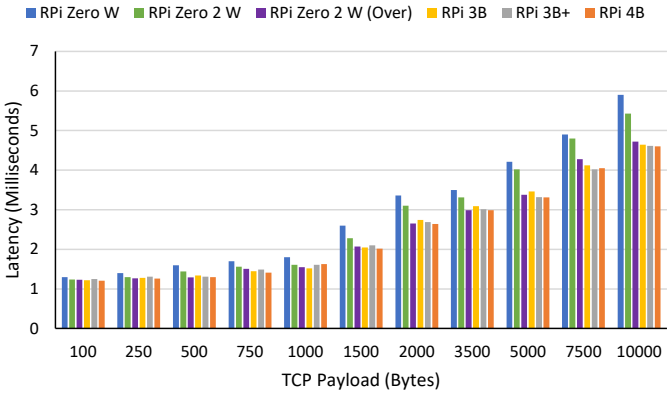

Fig. 24. TCP Latency with our Own Benchmark for Different RPi Models.

V. CONCLUSION AND FUTURE WORK

In this paper, we evaluated the performance of a number of SBCs: RPi Zero W, RPi Zero 2 W, RPi Zero 2 W overclocked, RPi 3B, RPi 3B+, and RPi 4B. The RPi 4B significantly outperformed the other SBCs under test. So far, it is the most potent RPis released by the Raspberry Pi Foundation, with a basic price (board only) of US$35, US$45, US$55, and US$75 for the 1, 2, 4, and 8 GB models, respectively. If used at its maximum power, a cooling solution is recommended; otherwise, the CPU will be throttled, resulting in slowing down the CPU frequency. Our study showed that the RPi Zero, released in February 2017, has limited capacity. It is the only 32-bit processor of the study, with just one core. However, it might still be the solution for many projects with low CPU and RAM requirements. Its basic price of US$10 (board only) is very attractive for projects with a low budget. In general, the RPi Zero 2 W overclocked, RPi 3B, and RPi 3B+ had similar performances. In some tests, the RPi Zero 2 W overclocked had a light advantage, while the RPi Zero 3B+ was slightly better in other tests. The basic price (board only) for the RPi Zero 2 W, RPi 3B, and RPi 3B+ is US$15, US$35, and US$35, respectively. The RPi Zero 2 W is the last SBC released by the Raspberry Pi Foundation, with this amazing price. It has the same form factor as the RPi Zero W, allowing an easy upgrade when required. However, its limited RAM (512 MB) can be a limitation for some projects, compared to the 1 GB of the RPi 3B and RPi 3B+.

When Ethernet is a project requirement, the RPi 3B, RPi 3B+, and RPi 4B have an integrated port. They offer Fast Ethernet (100 Mbps), Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps), and Gigabit Ethernet, respectively. For the other SBCs of this study, Ethernet can still be added through a USB port. However, this solution should be considered only in existing projects as an extension, since new projects will be more cost-effective when using RPis with native Ethernet support.

The RPi 400 [44] was not studied in the paper. It is a keyboard that incorporates an RPi 4B into it, with minor modifications. It is the easiest way to build a desktop computer based on an RPi. Even if they are mostly identical, the Raspberry Pi Foundation releases the RPi 400 with its Broadcom BCM2711 processor clocked at 1.8 GHz, while the RPi 4B was set to 1.5 GHz in previous versions of the Raspberry Pi OS. This is just due to an integrated robust cooling solution in the keyboard of the RPi 400. However, the last version of the Raspberry Pi OS (October 2021) now also sets the Broadcom BCM2711 processor of the RPi 4B at 1.8 GHz. Hence, as specified in Section III.A, the CPU of our RPi 4B was set to 1.8 GHz for all our experiments.

The RPi 3B+ is an improved version of the RPi 3B. The main difference is in the upgraded support for the network connection. According to the Raspberry Pi Foundation website, the RPi 3B will remain in production until at least January 2026 [22]. Both (RPi 3B and RPi 3B+) have the same price: US$35 for the board only. In most of our experiments, they had a similar performance with a slight advantage for the RPi 3B+. Hence, for new projects that are planning to use the RPi 3B, our recommendation is to consider the RPi 3B+ instead.

Even though a powerful processor means more heat and higher temperatures when running intensive CPU workloads, an interesting finding of this study is that the RPi 3B reached the highest temperature (more than 80º C) during the stress activity, which is significantly higher than the temperature reached by the RPi 3B+ and the RPi 4B for the same CPU workload. This shows that these two later models might have been improved in terms of heat management in comparison to their predecessor (RPi 3B).

As future work, we plan to extend our study to other SBCs, such as the ones of BeagleBoard [45]. We are also interested in focusing our efforts on the network performance of IPv4 and IPv6, for both Ethernet and WiFi, in SBCs.

ACKNOWLEDGMENT

We are grateful to “Faculty Commons” and the “College of Science & Mathematics” at Jacksonville State University for partially funding this project.

REFERENCES

[1] W. Jolles, “Broad-scale Applications of the Raspberry Pi: A Review and Guide for Biologists,” Methods Ecol. Evol., vol. 12, no. 9, pp. 1562–1579, 2021, doi: 10.1111/2041-210X.13652.

[2] L. K. Ramasamy and S. Kadry, Blockchain in the Industrial Internet of Things. IOP Publishing Ltd, 2121.

[3] D. B. C. Lima, R. M. B. Da Silva Lima, D. De Farias Medeiros, R. I. S. Pereira, C. P. De Souza, and O. Baiocchi, “A Performance Evaluation of Raspberry Pi Zero W Based Gateway Running MQTT Broker for IoT,” in Proceedings of the 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON 2019), Oct. 2019, pp. 0076–0081, doi: 10.1109/IEMCON.2019.8936206.

[4] A. W. Daher, A. Rizik, M. Muselli, H. Chible, and D. D. Caviglia, “Porting Rulex Software to the Raspberry Pi for Machine Learning Applications on the Edge,” Sensors, vol. 21, no. 19, pp. 1–16, 2021, doi: 10.3390/s21196526.

[5] A. Komninos, I. Simou, N. Gkorgkolis, and J. Garofalakis, “Performance of Raspberry Pi Microclusters for Edge Machine Learning in Tourism,” in Proceedings of the 2019 European Conference on Ambient Intelligence (AmI 2019), Nov. 2019, vol. 2492, pp. 1–10.

[6] H. D. Ghael, L. Solanki, and G. Sahu, “A Review Paper on Raspberry Pi and its Applications,” Int. J. Adv. Eng. Manag., vol. 2, no. 12, pp. 225–227, 2021.

[7] R. Morabito, “A Performance Evaluation of Container Technologies on Internet of Things Devices,” in Proceedings of the 2016 IEEE Conference on Computer Communications Workshops (INFOCOM 2016), Apr. 2016, pp. 1–2, doi: 10.1109/INFCOMW.2016.7562228.

[8] A. K. Kyaw, Y. Chen, and J. Joseph, “Pi-IDS: Evaluation of Open-source Intrusion Detection Systems on Raspberry Pi 2,” in Proceedings of the 2015 2nd International Conference on Information Security and Cyber Forensics (InfoSec 2015), Nov. 2015, pp. 165–170, doi: 10.1109/InfoSec.2015.7435523.

[9] A. Aspernäs and T. Simonsson, “IDS on Raspberry Pi : A Performance Evaluation,” 2015. http://lnu.diva-portal.org/smash/get/diva2:819555/FULLTEXT01.pdf.

[10] E. Gamess and S. Hernandez, “Performance Evaluation of SNMPv1/2c/3 using Different Security Models on Raspberry Pi,” Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 11, pp. 1–9, 2021, doi: 10.14569/ijacsa.2021.0121101.

[11] D. Mauro and K. Schmidt, Essential SNMP, 2nd ed. O’Reilly Media, 2005.

[12] L. H. Nunes et al., “Performance and Energy Evaluation of RESTful Web Services in Raspberry Pi,” in Proceedings of the 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC 2014), Dec. 2014, pp. 1–9, doi: 10.1109/PCCC.2014.7017086.

[13] Y. Guamán, G. Ninahualpa, G. Salazar, and T. Guarda, “Comparative Performance Analysis between MQTT and CoAP Protocols for IoT with Raspberry Pi 3 in IEEE 802.11 Environments,” in Proceedings of the 2020 15th Iberian Conference on Information Systems and Technologies (CISTI 2020), Jun. 2020, pp. 1–6, doi: 10.23919/CISTI49556.2020.9140905.

[14] A. Banks and R. Gupta, “MQTT Version 3.1.1,” OASIS Standard, Oct. 2014. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf.

[15] “MQTT Homepage.” https://mqtt.org.

[16] “CoAP – Constrained Application Protocol Homepage.” https://coap.technology.

[17] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252. Internet Engineering Task Force (IETF), Jun. 2014.

[18] D. Hawthorne, M. Kapralos, R. W. Blaine, and S. J. Matthews, “Evaluating Cryptographic Performance of Raspberry Pi Clusters,” in Proceedings of the 2020 IEEE High Performance Extreme Computing Conference (HPEC 2020), Sep. 2020, pp. 1–9, doi: 10.1109/HPEC43674.2020.9286247.

[19] E. Fernando, D. Agustin, M. Irsan, D. F. Murad, H. Rohayani, and D. Sujana, “Performance Comparison of Symmetries Encryption Algorithm AES and DES with Raspberry Pi,” in Proceedings of the 2019 4th International Conference on Sustainable Information Engineering and Technology (SIET 2019), Sep. 2019, pp. 353–357, doi: 10.1109/SIET48054.2019.8986122.

[20] “Raspberry Pi Zero W – Raspberry Pi.” https://www.raspberrypi.com/products/raspberry-pi-zero-w.

[21] “Raspberry Pi Zero 2 W – Raspberry Pi.” https://www.raspberrypi.com/products/raspberry-pi-zero-2-w.

[22] “Raspberry Pi 3 Model B – Raspberry Pi.” https://www.raspberrypi.com/products/raspberry-pi-3-model-b.

[23] “Raspberry Pi 3 Model B+ – Raspberry Pi.” https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus.

[24] "Raspberry Pi 4 Model B – Raspberry Pi." https://www.raspberrypi.com/products/raspberry-pi-4-model-b.

[25] "Stressberry: Stress Tests for the Raspberry Pi." https://github.com/nschloe/stressberry.

[26] "7-Zip." https://www.7-zip.org.

[27] D. S. N. Nunes, F. A. Louza, S. Gog, M. Ayala-Rincon, and G. Navarro, "A Grammar Compression Algorithm based on Induced Suffix Sorting," in Proceedings of the 2018 Data Compression Conference, Mar. 2018, pp. 42–51, doi: 10.1109/DCC.2018.00012.

[28] A. R. Rosete, K. R. Baker, and Y. Ma, "Using LZMA Compression for Spectrum Sensing with SDR Samples," in Proceedings of the 2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON 2018), Nov. 2018, pp. 282–287, doi: 10.1109/UEMCON.2018.8796574.

[29] "Sysbench: Scriptable Database and System Performance Benchmark." https://github.com/akopytov/sysbench.

[30] "Raspberry Pi 4 B Review and Benchmark - What's Improved over Pi 3 B+," Sep. 2019. https://ibug.io/blog/2019/09/raspberry-pi-4-review-benchmark.

[31] "Geekbench 5 - Cross-Platform Benchmark." https://www.geekbench.com.

[32] "STREAM: Sustainable Memory Bandwidth in High Performance Computers." https://www.cs.virginia.edu/stream.

[33] "STREAM: The de Facto Industry Standard Benchmark for Measuring Sustained Memory Bandwidth." https://github.com/jeffhammond/STREAM.

[34] "GeekBench for Linux ARM - Primate Labs Support." http://support.primatelabs.com/discussions/geekbench/70-geekbench-for-linux-arm.

[35] B. R. de Supinski et al., "The Ongoing Evolution of OpenMP," Proc. IEEE, vol. 106, no. 11, pp. 2004–2019, 2018, doi: 10.1109/JPROC.2018.2853600.

[36] OpenMP Architecture Review Board, OpenMP Application Programming Interface Specification Version 5.1. Independently published, 2020.

[37] T. Katagiri, "Basics of OpenMP Programming," in The Art of High Performance Computing for Computational Science, Vol. 1, M. Geshi, Ed. Springer, 2019, pp. 45–59.

[38] "Phoronix Test Suite: Open-Source, Automated Benchmarking." https://www.phoronix-test-suite.com.

[39] M. Still, The Definitive Guide to ImageMagick, 1st ed. Apress, 2006.

[40] S. ML, A. PJ, and S. DN, "Document Image Analysis Using Imagemagick and Tesseract-ocr," IARJSET, vol. 3, no. 5, pp. 108–112, 2016, doi: 10.17148/iarjset.2016.3523.

[41] "MPlayer: The Movie Player." http://www.mplayerhq.hu/design7/info.html.

[42] "IPerf2: A Tool that Measures Network Performance of TCP/UDP Including Latency." https://sourceforge.net/projects/iperf2.

[43] V. J. D. Barayuga and W. E. S. Yu, "Packet Level TCP Performance of NAT44, NAT64 and IPv6 using Iperf in the Context of IPv6 Migration," in Proceedings of the 2015 5th International Conference on IT Convergence and Security (ICITCS 2015), Aug. 2015, pp. 1–3, doi: 10.1109/ICITCS.2015.7293006.

[44] "Raspberry Pi 400 Personal Computer Kit – Raspberry Pi." https://www.raspberrypi.com/products/raspberry-pi-400.

[45] "BeagleBoard.org - Community Supported Open Hardware Computers for Making." https://beagleboard.org.