Evaluating container-based virtualization overhead on the general-purpose IoT platform

Wagner dos Santos Marques[§], Paulo Silas Severo de Souza[§],Fábio Diniz Rossi[*],
Guilherme da Cunha Rodrigues[†], Rodrigo N. Calheiros[‡],
Marcelo da Silva Conterato[§] and Tiago Coelho Ferreto[§]
Email: wagner.marques.001@acad.pucrs.br
[*]Federal Institute of Education, Science, and Technology Farroupilha (IFFAR)
Alegrete, Brazil
[†]Federal Institute of Education, Science, and Technology Sul-Riograndense (IFSUL)
Charqueadas, Brazil
[‡]School of Computing, Engineering and Mathematics
Western Sydney University, Australia
[§]Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil

***Abstract*—Virtualization has become a key technology that provides several advantages (e.g., flexibility, migration, isolation) for a plethora of computing infrastructures. However, traditional virtualization models are not suitable for embedded IoT platforms due to the virtualization layer overhead. New virtualization proposals such as container-based approaches arise as an option where performance is not impacted. However, when working on general-purpose embedded platforms, some studies have demonstrated that applications on container-based virtualization on embedded devices present considerable performance overhead. Since most performance evaluations on platforms using containers were run on servers, this study expands the testbed scenario by analyzing several metrics that measure the overhead of container-based virtualization layer on embedded IoT devices. Results demonstrated improvements up to 23% in terms of performance and up to 32% in terms of EDP.**

Keywords*—Embedded devices, performance, power consumption, virtualization.

I. Introduction

Currently, there is a great appeal for Internet of Things (IoT) [1]. This growth is due to several factors, such as the reduction in price and popularization of embedded devices, the maturation of data communication technologies, and service-oriented architectures. The core of an IoT ecosystem consists of edge devices, which is the infrastructure that encompasses data acquisition, network transmission, and its transformation into information [2]. Nowadays, many works propose to use general-purpose embedded devices to perform such functions due to the low cost and flexibility.

Therefore, there is a paradigm shift in embedded devices that once performed only one function and now must perform several operations concurrently on the same hardware. The technology that allows concurrent execution of applications on the same hardware is virtualization, implementing features such as isolation and elasticity. However, most virtualization models cause performance overhead. Therefore, it indicates that a large overhead can undermine the good performance of the edge devices and compromise IoT architecture. Traditional virtualization models implement an asynchronous I/O ring structure between Host OS and Guest OS [3]. Such ring is a descriptor queue allocated by domain (Host OS and Guests OS). However, descriptors do not directly contain I/O data. Instead, I/O data buffers are allocated out-of-band by the guest OS and indirectly referenced by I/O descriptors. For this reason, descriptors are accessed using producer-consumer pointers [4], and it causes I/O overhead.

In contrast, container-based virtualization (i.e., Linux Containers) performs this management via operating system kernel, avoiding virtual to real translation mechanisms by an additional layer. Thus, Linux Container presents performance as close as a native operating system [5]. This characteristic becomes vital as edge devices need to respond to geographically distributed sensors requests as fast as possible. From the above, this paper evaluates the performance overhead caused by the use of container-based virtualization in a general-purpose edge computing platform (processor, memory, disk, and network). Besides, we also took into consideration the power consumption during each evaluation performed to calculate the Energy-Delay Product, which considers both performance and energy consumption.

Among the specific contributions of this paper, we highlight the following: (i) we analyze the performance and power consumption overhead caused by containers during the execution of several applications with different behaviors running on an embedded system; and (ii) we verify if the container-based virtualization can bring benefits to edge devices depending on the requirements of the scenario they are inserted. This paper is organized as follows: Section II presents a theoretical framework on Internet of Things, Edge Computing, and Container-based virtualization; in Section III we discuss some studies focused on resource allocation in embedded devices; Section IV shows the evaluation scenario, benchmarks, results, and discussion; Finally, in Section V we present our conclusions and future work.

II. Background

Internet of Things (IoT) has expanded the possibilities of acting to a plethora of diverse environments widening the

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

application of computer science. In this scenario, the increase in capacity (e.g., performance, storage) and the low cost of sensors and prototyping platforms boosted this expansion. Today, sensors allow the management of various metrics, such as temperature sensors and even complex components in a smart environment. Figure 1 illustrates the interaction among the components of an IoT environment, where we highlight IoT in three principal components, namely, sensors, analysis, and the Internet.

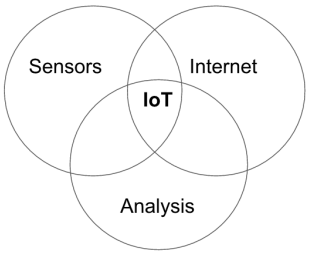


Figure 1. Relationship among IoT components.

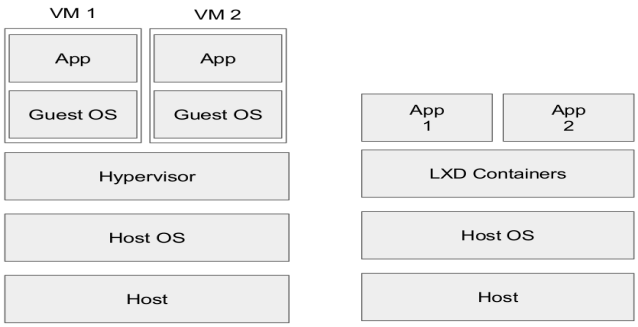
The data obtained from sensors are used to monitor and control the environments in various situations. In other words, such data generates autonomic run-time responses to modify the environment according to the application's metrics and goals. Edge Computing focuses on optimizing the performance and reducing the required bandwidth on the network by bringing the processing to near the source of data (such layer is called the edge of the network). In this context, embedded systems are responsible for the analysis and decision-making tasks, such as anomaly and intrusion detection. Such approach decreases the amount of data that need to be forwarded to external services, increasing the security (since the raw data is treated inside of the network), reducing the transmission costs and latency, and thereby improving the quality of service (QoS).

Performance constitutes a very relevant factor on Edge Computing since edge devices must be able to quickly handle the data coming from the sensors to make decisions. In this context, power saving also emerges as one of the biggest concerns since such devices usually are supplied by batteries [6]. Hence, there is an effort to find balance points between performance and power consumption to allow edge devices to operate with the required performance without changing its batteries constantly. The adoption of virtualization technology generates numerous benefits across multiple computing segments, such as lower acquisition costs, better utilization of computing resources, and higher performance measures. These features are commonly used in servers, computer networks and even on desktops [7].

Moreover, virtualized applications have lower performance than in native systems. The overhead caused by the virtualization layer is a primary concern during the adoption of this technology. Therefore, identifying and reducing the overhead posed by the virtualization layer have been the critical issue to a plethora of studies [8]. Along with the several kinds of virtualization, currently, hypervisor-based virtualization is the most popular. Such virtualization actuates on the hardware components level allowing the virtualization of several devices with different architectures in a single host in an isolated way. However, hypervisor-based virtualization imposes some overhead by emulating both hardware and operating system of each of its virtual machines, requiring more resources from the host device [5].

Some embedded devices have no support for hardware virtualization due to its particular purposes [9]. In this context, container-based virtualization emerges as a viable solution to these devices, since it actuates at the operating system level and requires fewer resources than virtual machines by using features from the host device operating system instead of emulating different systems to each container. Instead of manipulating hardware components directly, container-based virtualization uses kernel-level features of the host device to create the containers, which are isolated environments which have their own processes and libraries. Hence, this kind of virtualization brings several benefits such as smaller images (which do not need to have elements like devices drivers) and the fact that containers can be turned on or off faster than traditional virtual machines [10].

Based on the above, Linux Containers (LXC) is a virtualization technology that has focused on performance. The difference between traditional and container-based virtualization is shown in Figure 2. Container-based advantages regarding performance occur because Linux Containers do not maintain a hypervisor for the management of virtual machines. Instead, the operating system kernel itself performs this control, and for this reason, Linux Container presents performance close to a native operating system.



(a) Traditional Virtualization (b) Container Virtualization

Figure 2. Comparison between different types of virtualization.

Linux Containers is one of the most utilized containers systems used nowadays since it is included in the Linux operating systems and allows the creation and management of groups of processes (e.i., containers). Linux Containers utilizes many Linux kernel features for an operating system level virtualization, such as kernel namespaces and cgroups (control groups) [11].

On the top of Linux Containers infrastructure, LXD has emerged as a lightweight full OS system container solution on top of Linux Containers. Similar to a virtual machine, it provides hosting for multiple OS containers on a single host. Consequently, this architecture provides the highest density of guest OS containers per host among the others container solution as a direct result [11]. Also, LXD provides several benefits, such as verification and restore points [11], and facilitate the use of Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS), since REST API have a suitable support for it.

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

III. Related Work

There is a range of works that use virtualization on devices in fog and edge computing for a variety of purposes, such as security, resource isolation, flexibility, high availability, and centralized management. In this section, we highlight some of them.

Chiu et al. [12] have presented EdgePlex, an approach for Provider Edge Router (PE). The proposal uses a distributed architecture comprised of commodity switches and servers and aims to leverage the advances in virtualization and high-speed packet processing in servers and virtual machines to replicate functionality in edge routers. Authors emphasize the use of one virtual machine per client since it provides significant operational flexibility for providers and isolation for customers. To validate the proposal, a prototype was developed and used in a test bed. Results indicate that the proposed tool is feasible and capable of meeting the performance needs of service providers.

Bellavista and Zanni [13] address how to build a fog middleware support for IoT gateways to perform the decentralization of MQTT broker from the cloud involving edge and exploring industry-mature containerization to facilitate the interoperability and portability via node configuration standardization. For a proposal validation, the authors created and deployed a simple use case application in the domain of Smart Connected Vehicles (SCVs). Results were positive, achieving scalability of multiple container executions over very resource-constrained nodes such as Raspberry PIs.

Pahl et al. [14] performed a review of containers suitability for edge clouds. The authors presented the relevance of the use of such virtualization technology for Platform-as-a-Service concerning application packaging and orchestration. The authors indicated that container technology has the potential to substantially advance PaaS technology towards distributed heterogeneous clouds through a lightweight and interoperable solution.

Ferrer et al. [15] argue that the small amount of data generated by wearables and personal devices can be processed and stored on the edge of the network, i.e., close to the IoT devices. The authors claim that processing and storing a small quantity of data close to the sources has enabled faster control of the data ownership, response time, and semi-autonomy, which are requirements of critical applications. They also have discussed that Fog extends Cloud resources to the edge of the network, such as virtualization, enabling the customers to control their data. The customers can determine the engagement policies of their PHA (Personal Health Assistant) with the purpose to maintain a healthy life.

Jemaa et al. [16] introduce NFV (Network Function Virtualization) placement and provisioning optimization strategies over an edge-central cloud infrastructure. Accordingly, NFV represents a promising solution for wireless network providers to improve business the agility and cope with the continuing growth in data traffic. Also, virtualizing core network functions, as well as radio-access network services, can reduce the cost to deploy and operate large wireless networks. So, the authors proposed QoS models based on QoS-aware VNF placement and provisioning methods for the edge-central cloud system.

Burger et al. [17] describe that the performance of the player and the gaming service highly depend on the distance and latency to the game server. Thus, to reduce latency, servers can be migrated to the edge of the network where they are in closer to the players. To evaluate the impact of migration policies on the latency and load of game servers, the authors used a multiplayer online battle arena. The results show that deploying one additional edge server can already reduce the mean distance to the server by one-half and profoundly reduce the load on the dedicated server.

Based on the above, we can claim that previous work presented in this section did not evaluate the overhead caused by the adoption of container-based virtualization in edge computing, since such technology work with limited resources and aims at rapid responses to the customer. In this sense, we propose to perform the overhead evaluation caused by the container-based virtualization in edge devices.

IV. Evaluation and Discussion

Edge device must be able to run analysis and processing algorithms which may consume several hardware resources in a balanced way or even focus only some components, such as memory, disk, or CPU. In this context, we have decided to compare the performance of several types of applications running natively and on containers.

On embedded systems, the power consumption constitutes a very relevant factor since most of these devices are dependent on battery power [6]. Therefore, we also analyzed the elapsed time and the power consumption of all executed benchmarks to calculate its Energy-Delay Product (EDP). Using EDP, we show the point of balance among energy saving and performance. As a general-purpose IoT embedded platform, we used a Raspberry Pi 2 Model (which hardware specifications are shown on the Table I), running the operating system Raspbian[1], Linux kernel 4.4.50+. The applications were compiled with the GCC 4.9.2 and with the Fortran 4.9.2-10, and OpenMP 4.0. To validate the results, in all test we used the mean of 5 executions. Aiming to meet the aspects exposed above, we used the following benchmarks: IOZone (disk performance), NPB (processor performance), STREAM (memory performance), and AB (network performance).

Table I. Raspberry Pi 2 Model B hardware specifications.

Component	Specification
CPU	Broadcom BCM2837Arm7 Quad-core Cortex-7 900MHz
Memory (RAM)	1GB LPDDR2 SDRAM
Network	10/100 Ethernet Port
Power Supply	+5V Micro USB

A. Disk Performance

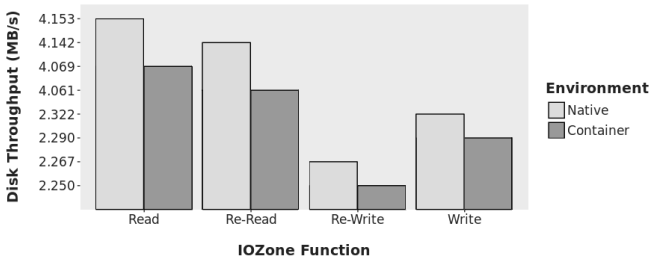
Disk performance was evaluated by using IOZone, a benchmark which analyzes the filesystem's throughput during the execution of several I/O operations. Once the tests finish, IOZone presents accurate information about every task executed, allowing a precise analysis of the disk performance. The benchmark was configured to create and manipulate a 100MB file, with 32k of record length. We ran four operations in the throughput mode with four threads: *Writing*: This operation measures the throughput when writing a new file

[1] Available at <https://www.raspberrypi.org/downloads/raspbian>.

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

on the disk. The writing process does not involve only the file being stored but also some metadata (e.g., information about the storage device); *Re-Writing*: This test measures the disk performance during the writing an already existent file. Such operation usually has a bigger throughput than writing a new file since it doesn't need to add elements like metadata (which already exists on the disk); *Reading*: The reading operation measures the disk performance when reading an existent file. This benchmark usually takes some time since it needs to add the file data into the device's Cache memory; *Re-reading*: Measures the disk performance when reading a file that was recently read. Usually, the Re-Read operation is faster than the Read one, since the operating system usually keeps the data of files recently read into the cache memory.

Results, presented in Figure 3, indicate that the use of container-based approach generated a throughput overhead of approximately 1.53% on all the executed operations. Moreover, such overhead was 54% bigger on the reading and re-reading tests. Nonetheless, the container-based approach showed a better EDP (0.322% smaller than the native's EDP). Results can be attributed to a good use of Cache memory, which serves the data for re-reading and re-writing. It may be a point to be researched in the future, which consists of the overhead of the virtualization layer over misses and hits on Cache, which can influence quite dramatically, even more in multicore embedded architectures.



	Environment	Execution Time	Energy-Delay Product
1	Native	191.492	46447.636
2	Container	189.938 (-0.81%)	46298.265 (-0.322%)

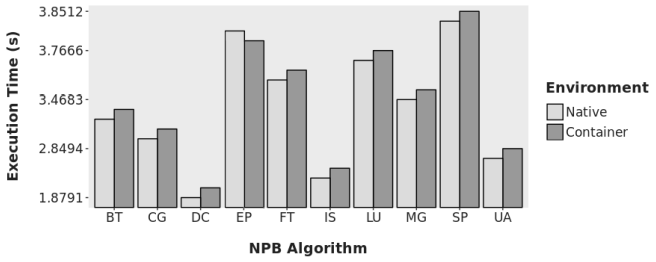
Figure 3. Disk performance comparison using IOZone.

B. HPC Applications Performance

The NAS Parallel Benchmarks[2] (NPB) was the test suite chosen to evaluate the performance and the EDP of native and container-based approaches. The NPB was developed by the Advanced Supercomputing Division of NASA (National Aeronautics and Space Administration), containing algorithms based on Computational Fluid Dynamics applications. The NPB benchmarks are divided into 3 groups: *Kernels:* Integer Sort (IS), Embarrassingly Parallel (EP), Conjugate Gradient (CG), Multi-Grid (MG), and Discrete 3D Fast Fourier Transform (FT); *Pseudo applications:* Block Tri-diagonal (BT), Scalar Penta-diagonal (SP), and Lower-Upper Gauss-Seidel (LU); *Parallel I/O (Input-Output) algorithms:* Unstructured Adaptive (UA), and Data Cube (DC).

Moreover, NPB have several classes, which specify the problems proportions to be solved by the algorithms. In this investigation, we used the S, W, and A classes, which are the smallest classes of the suite. The use of the container-based approach generated a performance overhead on 9 of the 10 executed algorithms. The only case in which the use of container slightly decreased the execution time was on the EP test. Regarding EDP, the evaluation showed balanced results. The container-based approach had better EDP results on kernel algorithms, presenting a gain of approximately 2.1%. On the other hand, the use of containers increased about 4.6% of EDP on the parallel I/O tests and approximately 0.85% during the execution of the pseudo-applications. The bigger overhead caused by the use of container occurred on the execution of the DC algorithm (rise of 8.53%). Graphical representation of both performance and EDP evaluations are presented in Figure 4.

The only case when the use of the container-based approach presented a better performance result than the native one was during the execution of the EP benchmark, which represents applications that can be divided into smaller independent tasks, i.e., can be processed in parallel without any need for communication between them. Such algorithms are very useful on parallel architectures such as the Internet of Things (e.g., depending on the scenario, several edge devices can work together on a single task like clustering-based anomaly detection).

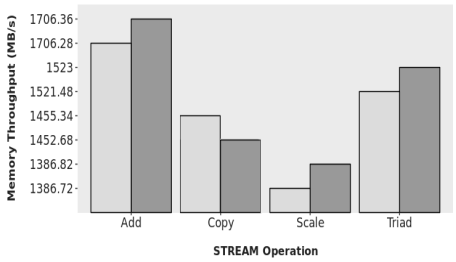


	NPB Algorithm	EDP Impact Caused by Containers Utilisation
1	DC	Loss of 8.53%
2	IS	Gain of 1.95%
3	UA	Loss of 0.67%
4	CG	Gain of 1.50%
5	BT	Loss of 1.10%
6	MG	Gain of 4.71%
7	FT	Gain of 2.63%
8	LU	Loss of 1.79%
9	EP	Loss of 0.56%
10	SP	Gain of 0.35%

Figure 4. Performance and EDP comparisons on HPC applications using NAS Parallel Benchmarks.

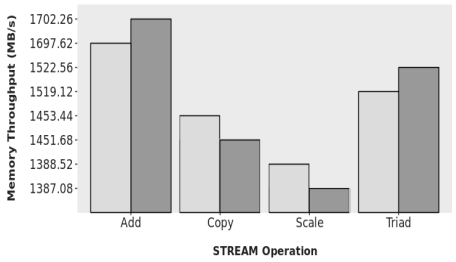
As we can see, applications that perform I/O are more penalized. It is a typical behavior in virtualized data center servers because of ring architecture, and it can also be observed in an embedded environment. Nonetheless, the positive results achieved during the execution of the embarrassingly parallel algorithm showed that there are some real-life embedded scenarios when the use of a container-based approach can bring some benefits (e.g., multiple edge devices can be used on the facial recognition of distributed embedded smart cameras for

[2]Available at: <https://www.nas.nasa.gov/publications/npb.html>.



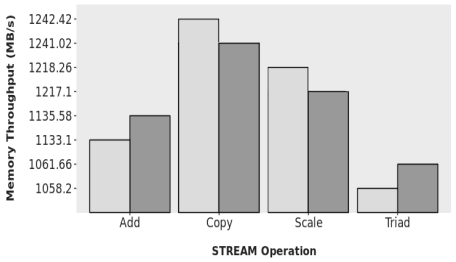
	Operation	Environment	Energy-Delay Product
1	Add	Container	0.3696 (-0.027%)
2	Add	Native	0.3697
3	Copy	Container	0.2254 (+0.399%)
4	Copy	Native	0.2245
5	Scale	Container	0.253 (+0.158%)
6	Scale	Native	0.2526
7	Triad	Container	0.472 (+0.190%)
8	Triad	Native	0.4729

(a) A



	Operation	Environment	Energy-Delay Product
1	Add	Native	0.0409
2	Add	Container	0.0413 (+0.969%)
3	Copy	Native	0.0247
4	Copy	Container	0.0252 (+1.984%)
5	Scale	Native	0.0271
6	Scale	Container	0.0276 (+1.812%)
7	Triad	Native	0.0509
8	Triad	Container	0.0515 (-1.179%)

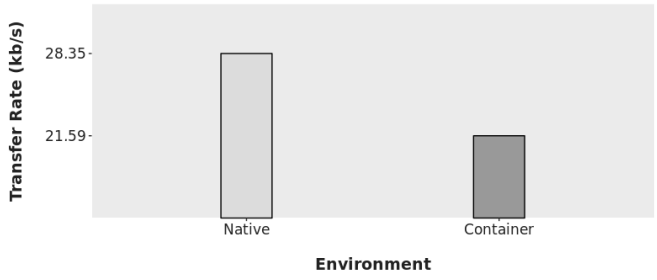
(b) B



	Operation	Environment	Energy-Delay Product
1	Add	Native	0.0103
2	Add	Container	0.0103
3	Copy	Native	0.0038
4	Copy	Container	0.0038
5	Scale	Native	0.0039
6	Scale	Container	0.004 (+2.564%)
7	Triad	Native	0.0118
8	Triad	Container	0.0118

(c) C

Figure 5. Memory performance and EDP evaluations using STREAM benchmark, where [A] represents the 228.9MiB array, [B] represents the 76.3MiB array, and [C] represents the 25.4MiB array.



	Environment	Elapsed Time (s)	Energy-Delay Product
1	Native	250.501	83760.734
2	Container	203.054 (-18.94%)	56558.044 (-32.5%)

Figure 6. Performance and EDP comparisons on network operations using Apache Benchmark.

surveillance).

C. Memory Performance

STREAM[3]: it is a benchmark that analyzes the memory bandwidth during the execution of vector kernels. We executed STREAM in a parallelized way (4 threads) through the OpenMP[4] API (Application Programming Interface). Although performing more than one operation by time, STREAM provides statistical information about every of its operations, which are: *Copy:* it is responsible for analyzing the transfer rate in the absence of arithmetic; *Scale:* it adds a simple arithmetic operation; *Add:* it aggregates multiple loading and storage ports on vector machines to be tested; *Triad:* it allows chained, overlapped, fused multiply, and add operations.

In order to get more accurate results, we modified the STREAM_ARRAY_SIZE constant, which specifies the amount of data which will be manipulated. In this sense, we ran STREAM using arrays which occupy 25.4MiB, 76.3MiB, and 228.9MiB. Moreover, we set the NTIMES constant (which corresponds to the number of iterations of each benchmark execution) to 50.

Results show that the container-based approach achieved higher memory throughput on the Add and Triad operations (rise of 0.17% and 0.29%, respectively), during the execution of the three chosen array sizes. The gain on the Add operation indicates that the container-based approach is effective during operations involving the filling and the manipulation of a fulfilled processor pipeline (e.g., large arrays manipulation tasks). The positive result regarding on the Triad benchmark shows that the use of containers can be a viable solution for the execution of some HPC jobs such as fused multiple-add (e.g., polynomial evaluations, matrix multiplication, etc.).

On the other hand, such approach presented a performance overhead during the execution of Copy and Scale tests (loss of 0.12% and 0.01%, respectively) using 25.4MiB and 76.3MiB arrays. Moreover, of 12 executed operations (4 with each array size), the use of containers increased the EDP on 7, and did not make a difference in 3 operations.

When we used a small dataset, which does not completely stress available memory, the trade-off between performance and energy savings is great. However, the results showed that the greater the amount of data managed and maintained by memory, the better the trade-off, and it occurs because of the proximity of the memory and processor in an embedded device.

[3]Available at: <https://www.cs.virginia.edu/stream>.
[4]Available at: <http://www.openmp.org>.

D. Network Performance

Considering that edge devices perform their functions by collecting data from sensors and sending and receiving information from a remote server, an assessment of their performance while using the internet becomes very important. During this evaluation we use ApacheBench (AB) [5], which can be used to analyze Hypertext Transfer Protocol (HTTP) traffic in web applications. It is a command line program that is part of the Apache web server and is used for the performance of web servers through HTTP requests a URL specified by the user. In this sense, we created an example application using the Ruby on Rails[6] web framework. Results show that the use of the container-based approach showed a performance overhead of 23.9%. Nonetheless, the use of containers decreased the EDP by 32.5%. Such results are graphically presented on Figure 6.

The results indicate that virtual switch maintained by the OS host manages requests coming from containers in order to balance the load, although it has lost performance, it can balance performance and energy savings in a fairer way. Moreover, the network is virtualized in the container, and such network virtualization acts as a second layer. In order to use the network virtualization, several parameters must be specified to define the network interfaces of the container. Several virtual interfaces can be assigned and used in a container even if the system has only one physical network interface. For this reason, we believe that the container's virtual network generated an overhead compared to the native approach.

V. Conclusion and Future Works

This study evaluated the overhead of the container-based virtualization layer using applications in a general-purpose IoT embedded environment by individually testing each hardware component that can influence performance. Results showed that the overhead imposed by such virtualization model could maintain performance close to the native, being an appropriate option to the edge devices.

Despite generating a performance overhead in the majority of the executed applications, in some cases, the use of containers also decreased the power consumption. In this sense, when analyzing both performance and power consumption of each of the executed benchmarks, we realized that using the container-based approach decreased the Energy-Delay Product in some algorithms such as the MultiGrid, whose behavior is similar to tasks involving linear problems like sparse linear systems. In the edge computing context, MultiGrid methods can be implemented in parallel image analysis algorithms which use multiple edge devices to give a fast feedback to the environment. The use of containers during the execution of the MultiGrid algorithm generated an EDP gain of 4.71% when compared to the native approach.

As future work, we intend to evaluate the application performance on container-based virtualization in a cluster of edge devices. Such a computational aggregate will allow evaluating issues of migration of containers among devices, aiming for elasticity.

[5]Available at: <https://httpd.apache.org/docs/2.4/programs/ab.html>.

[6]Available at: <http://rubyonrails.org>.

References

[1] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "Dtls based security and two-way authentication for the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2710 – 2723, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870513001029

[2] M. Medwed, "Iot security challenges and ways forward," in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*, ser. TrustED '16. New York, NY, USA: ACM, 2016, pp. 55–55.

[3] P. "Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 164–177.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Oct. 2003.

[5] M. G. Xavier, I. C. D. Oliveira, F. D. Rossi, R. D. D. Passos, K. J. Matteussi, and C. A. F. D. Rose, "A performance isolation analysis of disk-intensive workloads on container-based clouds," in *Proceedings of the 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, ser. PDP '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 253–260.

[6] A. F. Lorenzon, M. C. Cera, and A. C. S. Beck, "Performance and energy evaluation of different multi-threading interfaces in embedded and general purpose systems," *Journal of Signal Processing Systems*, vol. 80, no. 3, pp. 295–307, 2015.

[7] A. Patel, M. Daftedar, M. Shalan, and M. W. El-Kharashi, "Embedded hypervisor xvisor: A comparative analysis," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2015, pp. 682–691.

[8] X. Ding and J. Shan, "Diagnosing virtualization overhead for multi-threaded computation on multicore platforms," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2015, pp. 226–233.

[9] W. Hajji and F. P. Tso, "Understanding the performance of low power raspberry pi cloud for big data," *Electronics*, vol. 5, no. 2, p. 29, 2016.

[10] M. Eder, "Hypervisor-vs. container-based virtualization," *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, vol. 1, 2016.

[11] S. Gupta and D. Gera, "A comparison of lxd, docker and virtual machine," *International Journal of Scientific & Engineering Research*, 2016.

[12] A. Chiu, V. Gopalakrishnan, B. Han, M. Kablan, O. Spatscheck, C. Wang, and Y. Xu, "Edgeplex: Decomposing the provider edge for flexibilty and reliability," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15. New York, NY, USA: ACM, 2015, pp. 15:1–15:6.

[13] P. Bellavista and A. Zanni, "Feasibility of fog computing deployment based on docker containerization over raspberrypi," in *Proceedings of the 18th International Conference on Distributed Computing and Networking*, ser. ICDCN '17. New York, NY, USA: ACM, 2017, pp. 16:1–16:10.

[14] C. Pahl and B. Lee, "Containers and clusters for edge cloud architectures–a technology review," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*. IEEE, 2015, pp. 379–386.

[15] O. Ferrer-Roca, R. Tous, and R. Milito, "Big and small data: The fog," in *Identification, Information and Knowledge in the Internet of Things (IIKI), 2014 International Conference on*. IEEE, 2014, pp. 260–261.

[16] F. Ben Jemaa, G. Pujolle, and M. Pariente, "Analytical models for qos-driven vnf placement and provisioning in wireless carrier cloud," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '16. New York, NY, USA: ACM, 2016, pp. 148–155.

[17] V. Burger, J. F. Pajo, O. R. Sanchez, M. Seufert, C. Schwartz, F. Wamser, F. Davoli, and P. Tran-Gia, "Load dynamics of a multi-player online battle arena and simulative assessment of edge server placements," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 17:1–17:9.