Energy Usage Profiling for Virtualized Single Board Computer Clusters

Stefanos Sagkriotis
School of Computing Science
University of Glasgow, UK
s.sagkriotis.1@research.gla.ac.uk

Christos Anagnostopoulos
School of Computing Science
University of Glasgow, UK
christos.anagnostopoulos@glasgow.ac.uk

Dimitrios P. Pezaros
School of Computing Science
University of Glasgow, UK
dimitrios.pezaros@glasgow.ac.uk

***Abstract*—With Network Function Virtualization (NFV) platforms gaining ground, we question the combination of NFV and Single Board Computers (SBCs) in terms of compatibility, reliability, and energy consumption. A mini cluster of SBCs is used to develop a scalable and resilient energy monitoring application. The application is employed to discover the energy demands of a NFV platform in modern SBCs, and build the energy profile of the devices and the deployed services. We use the results and the added knowledge from building the application to strengthen the argument that SBC clusters can support virtualized service deployment. This evidence, alongside the rich gamut of characteristics that SBCs hold, proves that they are a viable option for edge components of a fog network. Our results show that running different virtualised processes offers added functionality, resilience and scalability without heavily sacrificing energy consumption.**

***Index Terms*—Single Board Computer, Energy Monitoring, nergy Profiling, Container, Fog Computing, IoT, Edge Cluster**

I. Introduction

A plethora of modern applications e.g., smart city applications, connected vehicles, smart grids, that require a large number of nodes in combination with characteristics like interoperability, mobility, deployment in distant locations, and integration with sensors, rely on the adoption of fog computing platforms whose capabilities are shaped by among other factors, the specifications of devices located at the edge of the network [1]. Modern SBCs present a wide range of characteristics that satisfy the requirements posed by modern applications, thus classifying them as a choice for the edge components of fog computing networks, either as standalone devices or as clusters of devices. SBCs have been used in smart meter applications, energy management platforms, and as a key-device for e-Health monitoring systems [2] [3] [4].

SBCs are, in most cases, low-cost devices with a small form factor, low energy and passive cooling requirements. They offer connectivity with a wide range of sensors and devices [5]. IoT devices like the Raspberry Pi [6], present enough processing and storage resources to allow formation of small clusters with the potential of processing data as close to the edge as possible, reducing bandwidth consumption and offering a better Quality of Service (QoS) [3] [7].

Virtualization, by offering the necessary level of hardware-abstraction, allows stable development of applications across a variety of devices and is a key enabler for the deployment of multiple services to a cluster [8]. However, full machine virtualization results in high demand for computing resources. This can be a significant barrier for the deployment of virtualized applications in SBCs. A solution that mitigates this problem by reducing resource requirements is containers. Containers enclose only the necessary code dependencies for the execution of a program, omitting the overhead imposed by a complete virtualized OS. The Docker container platform [9] is one of the most renowned container development frameworks and offers a wide set of images designed for the architecture of Raspberry Pi's Central Processing Unit (CPU), the `Arm` architecture, allowing the development of a variety of applications while maintaining compatibility with SBCs.

Virtualized Network Functions (VNFs) can be deployed in the form of docker containers, facilitating a seamless virtualization for the entire network and maximizing the utilization of the various hardware combinations that constitute a Fog cluster. While containers are lightweight, their lack of guest OS results in a lack of resilience [10] and, consequently, they cannot be orchestrated and managed effectively throughout the cluster. Therefore, a platform that ensures a normal life-cycle for VNFs and deals with the networking requirements for VNF scaling is required.

One of the most prominent platforms that uses Software Defined Networking (SDN) to alleviate the issues of VNF management is Kubernetes [11]. Kubernetes supports the deployment and management of pods, a collection of one or more containers that share the same resources. Pods offer the necessary level of abstraction for easy management and resilience of multiple containers. Kubernetes supports the integration of SDN frameworks like Flannel [12] for orchestrating the pods that host the deployed containers.

Contribution: We contribute with a virtualized, scalable, and fault-tolerant energy monitoring application for SBC clusters through which one can derive an energy profile for the deployed services. This evidence can be integrated into network planning or used for further research into optimizing energy consumption. Furthermore, we showcase the adaptability of SBC clusters to versatile computing environments and make their use more appealing as parts of a fog computing network.

The key arguments and motivation on why energy resources monitoring is important for SBC clusters are described in Section II-A. A proposal for a virtualized and scalable monitoring application for small SBC clusters is analyzed in Section II-B. An evaluation of the proposed method of monitoring is reported in Section III. A brief survey on the work conducted on the topic, as well as suggestions for future work can be found in Section IV. Section V concludes the paper.

II. Methodology & Design

A. Rationale

Sensor nodes are typically powered by batteries, which on many applications are not replaceable after their depletion and, thus, can restrict the lifetime of both individual nodes and the network [13]. Hence, the need to monitor the remaining battery capacity to predict node failures due to depletion. It is important to identify and stop the applications or services that increase consumption and can put a node at the risk of energy starvation. Energy draining can be detected through identifying patterns of high consumption and correlating that to a specific application. In that essence, system administrators can be notified about processes that damage the lifetime of nodes and take appropriate action.

An approach to analyze data as close to the harvesting point as possible (as stated in Section I) is the development of edge cloud architectures that accommodate both data collection and data processing needs [14]. On that account, data processing establishes an additional energy requirement for the sensor nodes. The significance of this requirement has to be examined on top of the virtualization framework energy overhead posed to the sensor nodes. To accomplish this, a monitoring application needs to be employed and enable the energy profiling of the processes and the cluster nodes' consumption.

Automating the placement of services in a virtualized network is an upcoming field of research in the networking community. The measurements provided by monitoring applications can be among the various constraints integrated in a mapping policy that will distribute the energy consumption evenly across all nodes, maximizing the network lifetime. To make this possible, the monitoring application should be virtualized in order to be easily distributed across all the nodes. By prolonging the lifetime of the network and reducing failures, the posibility of reducing the Capital Expenditure (CAPEX) of the edge cluster is revealed, making SBC clusters appealing to service operators.

Based on the aforementioned arguments, the development of applications for SBC clusters for the edge of fog networks will benefit from an energy monitoring system. To further investigate this, we have developed a testbed comprised of commodity SBCs (more details in Section II-B) to explore and showcase the functionality behind integrating mini clusters at the edge of the network. A virtualized energy monitoring application is deployed to the cluster, to help us gather measurements for building the energy profile of the devices and the monitoring application itself.

B. Design & Implementation

To explore the feasibility of a virtualized SBC cluster, we have built a testbed of SBCs and deployed a NFV framework. To validate that the cluster is able to support both the gathering of the measurements and their processing, we paired the SBCs with sensors that measure each node's power consumption. A virtualized application is responsible for processing measurements and figure out the State of Charge (SOC) of the battery alongside maintaining a database with the measurements. The details of the testbed as well as the design decisions for the application are described in the sections following.

1) Testbed: To explore the real potential of an edge SBC cluster, we have implemented a testbed comprised of four Raspberry Pi 3 Model B devices [6]. These devices are a typical representation of an IoT device, commonly used as the core devices for many SBC clusters built in the past [5]. By being compatible with a wide range of sensor devices, the Raspberry Pi can function both as a sensor device and a processing node. The Operating System (OS) of choice is Raspbian OS, a variation of Debian that is officially supported by the manufacturer of Raspberry Pis.

The energy source for each device is a battery rated at the capacity of 5200mAh [15]. The sensor device for the electric measurements of the power consumption for each node is the UM24C module [16], which offers bluetooth connectivity with the Raspberry Pi. Among the goals of this paper is to investigate the current state of NFV platforms in terms of compatibility with bluetooth modules and similar external devices. A diagram depicting the connections among the devices can be seen in Figure 1.

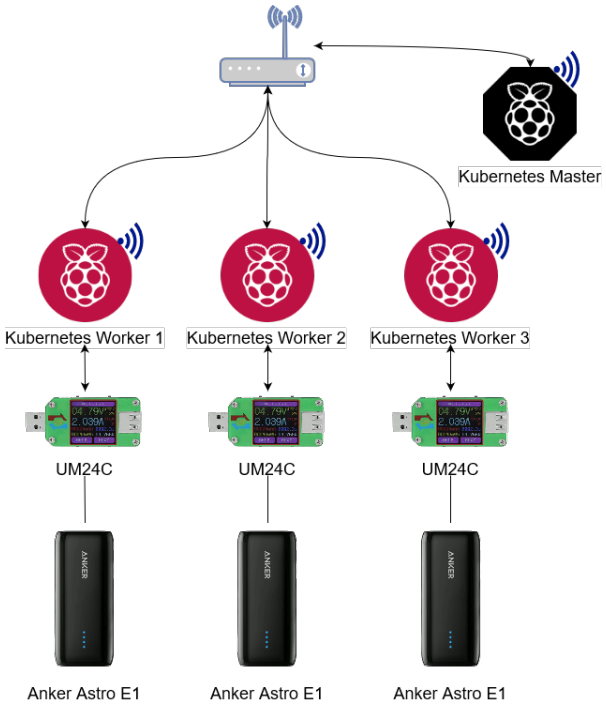

Fig. 1. Overview of the system architecture.

As mentioned in Section I, virtualization is a key-enabler for utilizing every device in the cluster. To satisfy the need for developing a virtualized application, we have selected Docker containers [9]. Even though Docker provides rich documentation that aids in the development process of applications, when these applications are scaled to many devices, their management and orchestration can be a demanding task. Kubernetes [11] can help alleviate the problems that occur when managing scaled containerized applications.

Kubernetes provides an Application Programming Interface (API) that helps customize the distribution of the containers, reschedule them on fail and monitor their behavior and diagnose any failures. An SDN component orchestrates and manages the pods that host the containers. In our case, this SDN component is Weave-net, which was found to require little set up time and presented full compatibility with the `Arm` architecture as well as reliability.

2) Energy Monitoring Application: The energy monitoring must be handled by a virtualized application that will communicate with the UM24C module to obtain the measurements. The application pods should have the ability to recover from errors as pods have the tendency to fail frequently. The application must also process the measurements and calculate an estimation of the remaining capacity of the battery. This metric, called State of Charge (SOC), will be updated on a database table separate for each of the nodes. The database must be maintained in more than one node for data recovery in case of battery depletion or failure of the pods.

Obtaining the values that the UM24C broadcasts was not a trivial task as there is no documentation on how to communicate with the device besides the Windows OS restricted GUI application. As a result, bluetooth communication had to be deciphered through a reverse engineering process. Communication with the UM24C device was achieved through serial-over-bluetooth connectivity. The communication is based on a simple message exchange. There is an initial message which the application must send to UM24C in fixed time intervals to trigger the broadcast of the measurements. The required measurements are acquired by investigating the 130 bytes returned message.

The SOC estimation of a battery, depending on the estimation model in use (e.g., [17]), can be dependent on factors like the state of health of a battery, the specific battery model, etc. These factors are typically hardware related and estimations that integrate them have to be considered coupled with the hardware under use. In such cases, they have to be re-estimated upon hardware change. Such complications are less present in the coulomb counting method [18]. By using the coulomb counting method, the total electric charge that a battery absorbs while being charged is monitored and the same stands for the total charge that is released when the battery is used to depletion. In principle, the SOC is estimated according to the percentage of the electric charge that exited the battery over the electric charge that entered the battery:
With $Q_{releasable}$ denoting the released capacity when the battery is completely discharged, and Q_{rated} depicting the rated capacity of the battery, SOC percentage is given by:

$$SOC = \frac{Q_{releasable}}{Q_{rated}} \tag{1}$$

Because of its simple yet accurate approach, we decided to adopt the coulomb counting method for our application. To get more accurate results, we decided to evaluate the Q_{rated} capacity by examining the actual electric charge that the battery is able to deliver over a number of charge-discharge cycles. We followed a process similar with the coulombic efficiency estimation that authors proposed in [18]. Coulombic efficiency, denoted with η, was found to be $\approx 57.69\%$ of the rated capacity, or around 3000mAh in total capacity. To increase the accuracy of the SOC estimation, Q_{max} was used to denote the maximum releasable capacity and eq. 1 was adjusted to:

$$SOC = \frac{Q_{max}}{\eta \cdot Q_{rated}} = \frac{Q_{max}}{0.576923 \cdot 5200mAh} \tag{2}$$

The formula used in our case is quite similar with the formula used in the original coulomb counting method.

- When fully charged, the SOC is given by eq. 2.
- During discharge for an operating period T, the Depth Of Discharge (DOD) number is used to denote the percentage of the discharged capacity. DOD is measured and then subtracted from the total SOC. I_b states the discharging current in amperes:

$$\Delta DOD(T) = \frac{-\int_{t_0}^{t_0+T} I_b(t)\,dt}{\eta \cdot Q_{rated}} \tag{3}$$

$$DOD(t) = DOD(t_0) + \Delta DOD(T) \tag{4}$$

$$SOC(t) = 1 - DOD(t) \tag{5}$$

Estimated SOC values must be stored in the database after being written in a Redis cache. The pods responsible for these operations are always considered prone to failure. Therefore, the application must be designed according to a resilient architecture. Replicas of the database have to be stored in multiple nodes, so that it can be recovered in case of unexpected failures. Equally, a caching mechanism is used to ensure that services are able to restore their last state before failure. If the pod hosting the cache and the application container fails, the application uses the database to restore its last state. The final architecture of the application can be seen in Figure 2. Each dotted rectangle represents a node. Inside each node there are pods, represented by black rectangles. The collection of pods necessary for the execution of Weave-net are represented with green rectangles. The colored rectangles inside the pods depict the containers running in each pod.

The pod named `double-pod` is the main component of the application. Inside `double-pod` there is a Redis container, which functions both as a cache for the data and a safety measure in case of failure. Data can be requested from the Redis cache of each node in case of high traffic on the database. This can help reduce load from the database at times

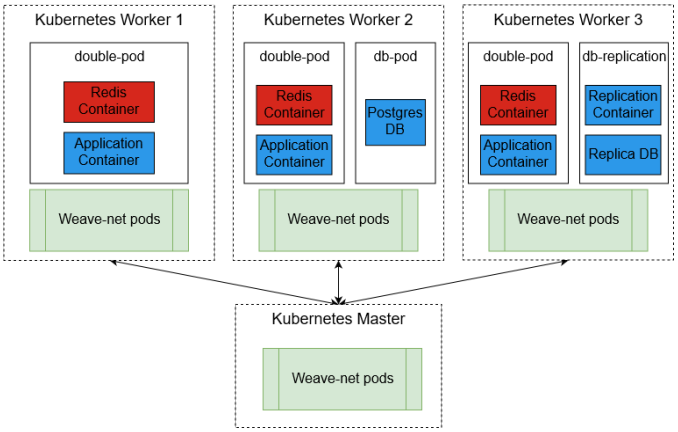

Fig. 2. Overview of application architecture

of congestion, when for example an administrator performs demanding tasks, and save the database pod from failure.

In the `double-pod` there is also the application container which performs data gathering, SOC estimation, and saves the measurements in the Redis container and in the Postgres database (located in the `db-pod`). `Double-pod` is deployed in every worker node and measures their energy requirements. The database is replicated through the `db-replication` pod, which holds the replica database and a container responsible for updating the replica. The `db-replication` pod can easily scale to allow higher safety levels.

The redundancy characteristics of our architecture are not bound to energy monitoring applications and could be equally used for other applications that make use of sensing devices and need redundant storage of the measurements. These fault-tolerant characteristics can be combined with the scalable properties of the design to make it appealing for applications with dynamic node numbers. NFV on top of IoT devices allows altering the processing of the measurements to employ various processing routines. For example, regression models or similar statistical methods can be used. Our design is applicable to cases like that of Unmanned Vehicles to facilitate fault-tolerant data gathering and processing [19].

III. Performance Evaluation

The NFV platform, despite realiably working and supporting the developed application, does not currently provide an official method for mounting host devices like UM24C to pods. The same stands for Docker. Nonetheless, the `volumeMounts` feature of Kubernetes in combination with a privileged execution of the necessary pods were enough to resolve the issue. The fact that the cluster is fully functional proves that such combinations of software and hardware can be used to host virtualized applications in SBC clusters. However, a way to support device mounting in pods without granting generic rights, like rights for manipulation of the network stack, is an addition that will enhance security control.

The application presented a small amount of fails. On more than 40 hours of testing, pods failed 3 times and were able to recover without loosing any data. Its resilient architecture seems mandatory as pod failures are likely to happen in a platform under development, like Kubernetes. The observed failures were not linked to development mistakes. Instead, platform related errors were the main source of pod failures.

To obtain an insight on the energy requirements of a standalone Raspberry Pi, we performed a CPU stress test and measured the voltage, and the current drained from the battery until depletion. The voltage graph of Figure 3 reveals a sawtooth function whose period gets shorter near the depletion time of the battery. This sawtooth behaviour is due to the voltage regulator circuit that is used in the Anker Astro E1, which ensures that the voltage never drops bellow a certain value.

The graph of current, in Figure 4, displays a peak current of $\approx 0.750A$ at the beginning of the measurements subsequently remaining between $0.6A$ - $0.68A$. This comes as a result of the thermal throttling that the Raspberry Pi experiences. As CPU temperatures raise above 83.5 degrees Celsius, the frequency of the CPU drops from 1.2GHz to 600MHz which results in the aforementioned average of power consumption.

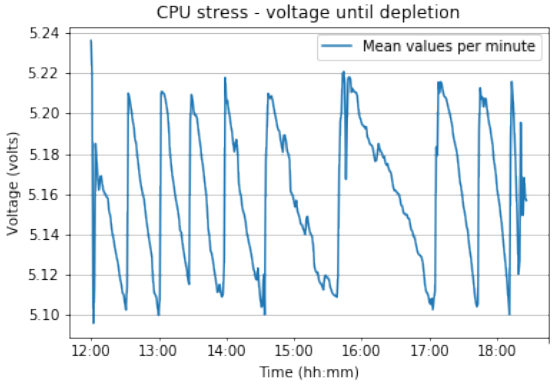

Fig. 3. Voltage of the Raspberry Pi's battery under CPU stress.

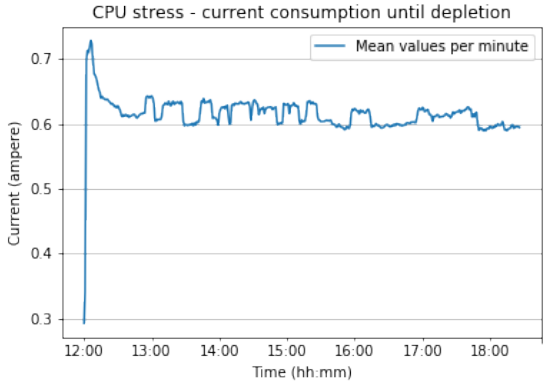

Fig. 4. Current usage by the Raspberry Pi battery under CPU stress.

The SOC estimation method is evaluated by checking for a synchronization between the estimated remaining capacity

and battery depletion. This was found to be accurate in most cases, an example of which is shown in Figures 5 and 6. Differentiation occurred when batteries with different states of health and age were tested.

To demonstrate the SOC estimation method, we test it against unsteady current values, expecting a change of behavior in accordance to the fluctuation of current. The input values of current are shown in Figure 5 and the corresponding output in Figure 6. Opposite to the expected output, the SOC estimation is linear without presenting any major changes that correspond to the varying values of current. The hypothesis behind this observation is that the changes in the current are neither big nor long enough to cause such change when compared with the capacity of the battery.

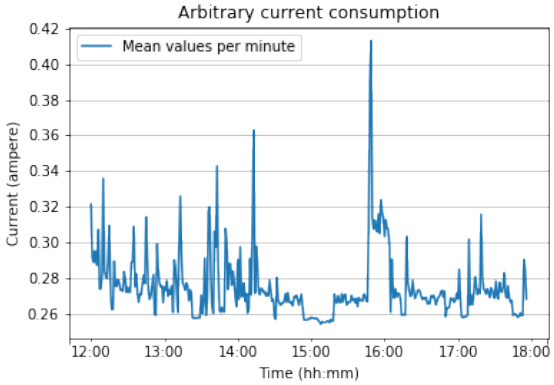

Fig. 5. Input of current values to the SOC estimation algorithm.

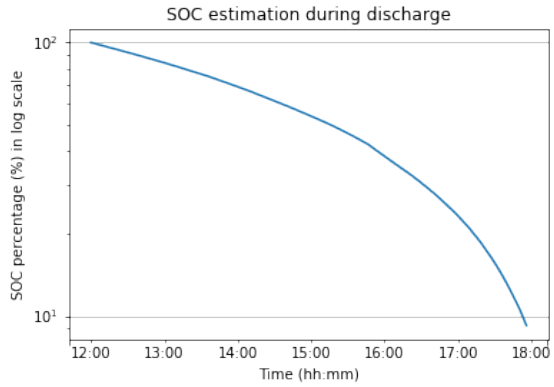

Fig. 6. Response of the coulomb counting method to the input of Figure 5

Another area of investigation is the average energy consumption of nodes depending on the tasks that are executed on them. The measurements for the nodes were obtained while the monitoring application was at full scale for 1.5 hours on the testbed. The task allocation for each node was: the master node executing weave-net and managing the running pods, a worker performing energy measurements and logging, another worker performing measurement and logging while running the database where measures are stored. The results are shown in Figure 7.

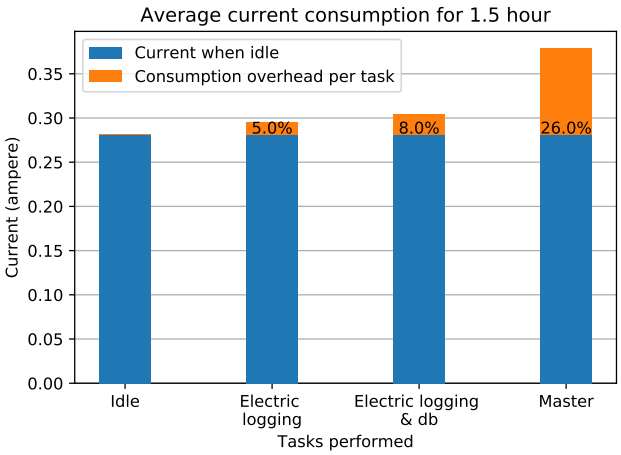

Fig. 7. Average current measurement per activity.

Our comparison basis for energy needs is an idle Raspberry Pi which was found to consume an average of $0.281A$. The node responsible for the logging of energy measurements displayed an increase of $\approx 5\%$, reaching $0.295A$. The combination of database and logging services was found to consume an average of $0.304A$, or $\approx 8\%$ growth from the baseline measurements. The master node was found to be the most energy-demanding node, with an average consumption of $0.379A$ which translates to $\approx 26\%$ increase in power consumption compared to our basis. The explanation behind this result is that the pod orchestration and management for the entire network is mainly executed on the master and this is likely to cause a greater energy demand. What this concludes is that the master node seems to be the most affected node from the execution of Kubernetes in terms of energy consumption.

Overall, the successful deployment of the testbed proved that SBC clusters are ready to be used in conjunction with NFV platforms to host custom-made applications interconnected with sensor hardware. The lack of an official method to mount host devices to VNFs is not a limiting factor, at least for the development stage of applications. Additionally, the overall stability of the hardware, the OS, and the NFV platform is sufficient to claim that SBC clusters have the potential to be considered as trustworthy edge devices when a resilient application design is employed. Even when executing NFV platforms, their small energy footprint remains. This justifies their suitability for applications that run on batteries and have mobility or distant location characteristics. The NFV support makes the management of such networks even easier, especially if resilient and scalable monitoring applications ensure the normal operation of the network. All these contribute towards the use of SBC clusters as the infrastructure to develop fog-oriented applications.

IV. Related Work

VNFs allow migrating functions to other devices on the context of a network-wide policy. This possibility results in a

new area of research aiming at the optimal placement of VNFs according to a certain policy or constraint.

In [20], a heuristic algorithm was used for optimizing VNF placement. Work in this field will enable a certain level of automation in VNF mapping which in turn can reduce Operating Expenditure (OPEX) and improve parameters of the network like energy efficiency, throughput, etc. Our future work aims at integrating electric measurements to various optimization algorithms responsible for VNF mapping and observe whether this type of optimization is feasible and realistic in modern networks.

Another proof of concept for using SBCs with containers to build edge clouds is that of [14]. In this work, the authors have built an edge cloud and verified that it is feasible. Even more, they identified that higher level VNF management platforms still need improvements and further development which comes to support our findings as stated in Section II-B. On the opposite, the authors described that lower level container platforms, like Docker, are fairly well established. The OpenMANO [21] community has been working closely with the ETSI body [22] to improve VNF management and orchestration. Part of our future work will be on examining and working on the OpenMANO framework to discover how higher level VNF management platforms can be improved.

Energy consumption of Raspberry Pis that run containers has been measured and analyzed in [23]. Specific benchmarks have been conducted for CPU, Memory, Network Input/Output (I/O), and Disk I/O. The results verify that the energy requirement of the device is not heavily affected by the execution of the Docker platform. This consolidates our argument about the deployemnt of NFV platforms for low-power IoT applications without the need for excessive power or high-end hardware.

Energy consumption of Raspberry Pis that run containers has been measured and analyzed in [24].

V. Conclusions

We demonstrated a fully functional cluster of SBCs that is running a NFV framework for application deployment. We have developed an application for energy monitoring of the network, and revealed the energy profile of the nodes and the overhead posed by the NFV framework. We have argued that such infrastructures can be considered as an option for placement at the edge of Fog networks, since their compatibility, reliability, mobility and small energy footprint characteristics are now combined with NFV support, making it an appealing option for fog applications. The outcome of our work indicates that virtualized service execution can be handled by SBCs while preserving their low-energy consumption characteristic.

Acknowledgments

This research has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/R511936/1, EP/N033957/1, and EP/P004024/1; by BT (Voucher No. 17000117); by the Huawei Innovation Research Program (Grant No. 300952); by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS – Resilient communication and services; and by the EU H2020 GNFUV Project RAWFIE-OC2-EXP-SCI (Grant No. 645220) under the EC FIRE+ initiative.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[2] S.-Y. Chen, C.-F. Lai, Y.-M. Huang, and Y.-L. Jeng, "Intelligent home-appliance recognition over iot cloud network," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*. IEEE, 2013, pp. 639–643.

[3] M. A. Al Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet of Things*, vol. 3, no. 2, pp. 161–169, 2016.

[4] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[5] S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, and J. Singer, "Commodity single board computer clusters and their applications," *Future Generation Computer Systems*, vol. 89, pp. 201–212, 2018.

[6] "Raspberry Pi 3," https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, accessed: 2018-10-20.

[7] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. Obrien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Computing*, vol. 17, no. 2, pp. 349–358, 2014.

[8] Q. Duan, Y. Yan, and A. V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Transactions on Network and Service Management*, vol. 9, no. 4, pp. 373–392, 2012.

[9] "Docker Containers," https://www.docker.com/resources/what-container, accessed: 2018-10-20.

[10] D. K. Rensin, *Kubernetes - Scheduling the Future at Cloud Scale*. 1005 Gravenstein Highway North Sebastopol, CA 95472: O'Reilly Media, 2015. [Online]. Available: http://www.oreilly.com/webops-perf/free/kubernetes.csp

[11] "Kubernetes," https://kubernetes.io/, accessed: 2018-10-20.

[12] "Flannel," https://coreos.com/flannel/docs/latest/, accessed: 2018-10-20.

[13] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[14] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee, "A container-based edge cloud paas architecture based on raspberry pi clusters," in *Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on*. IEEE, 2016, pp. 117–124.

[15] "Anker Astro E1," https://www.anker.com/products/variant/astro-e1/A1211012, accessed: 2018-10-20.

[16] "UM24C," https://www.mediafire.com/folder/0jt6xx2cyn7jt, accessed: 2018-10-20.

[17] F. Huet, "A review of impedance measurements for determination of the state-of-charge or state-of-health of secondary batteries," *Journal of power sources*, vol. 70, no. 1, pp. 59–69, 1998.

[18] K. S. Ng, C.-S. Moo, Y.-P. Chen, and Y.-C. Hsieh, "Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries," *Applied energy*, vol. 86, no. 9, pp. 1506–1511, 2009.

[19] "GNFUV," https://sites.google.com/view/gnfuv/home, accessed: 2018-10-20.

[20] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*. IEEE, 2015, pp. 1–9.

[21] "OpenMANO," https://osm.etsi.org/, accessed: 2018-10-20.

[22] "ETSI body," https://www.etsi.org/, accessed: 2018-10-20.

[23] R. Morabito, "A performance evaluation of container technologies on internet of things devices," in *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*. IEEE, 2016, pp. 999–1000.

[24] R. Oma, S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa, "An energy-efficient model for fog computing in the internet of things (iot)," *Internet of Things*, vol. 1, pp. 14–26, 2018.