Assessing the Power Cost of Virtualization Through Real-world Workloads

Senay Semu Tadesse
Politecnico di Torino
Email: d037964@polito.it

Francesco Malandrino
Politecnico di Torino
Email: francesco.malandrino@polito.it

Carla-Fabiana Chiasserini
Politecnico di Torino
Email: chiasserini@polito.it

Claudio Casetti
Politecnico di Torino
Email: casetti@polito.it

***Abstract*—Next-generation mobile networks will be heavily based on virtualization and their pervasiveness raises many questions regarding the energy efficiency of an architecture that requires distributed computing resources at the network edge. In this paper, we focus on the two main virtualization approaches, i.e., virtual machines and containers, which play a primary role in the provisioning of MEC-based services for mobile users. Specifically, we compare the two approaches from the viewpoint of the power consumption they are associated with – a metric significantly affecting the network provider's costs as well as the ICT environment footprint. Through a set of real-world experiments, using real-world video streaming and gaming applications, we assess not only the magnitude of the power consumption we incur, but also how it evolves as the workload increases. Our results show that containers are both more power-efficient and more scalable than virtual machines.**

I. Introduction

One of the most notable differences between next-generation ("5G") mobile networks and present-day ones is the former's ability to serve user request within the network itself. Instead of traveling all the way to an Internet-based server, requests will be processed at computation-capable nodes located at the network edge, i.e., as close as possible to users. This paradigm is known as multi-access edge computing (MEC).

Virtualization is a key enabling technology of MEC, as it allows general-purpose network hardware to process requests of any type. At the same time, it is associated with an overhead in terms of CPU usage and memory occupation; such overhead in turn translates into additional power consumption. Power consumption is an increasingly important key performance indicator (KPI) for all types of networks, as it affects their profitability as well as their environmental sustainability. In view of populating the network edge with computational resources to realize cloud and fog computing scenarios, it is important that virtualization techniques are chosen accounting for the associated power consumption.

The traditional approach is virtual machine (VM)-based virtualization: as summarized in Fig. 1, a hypervisor software emulates a whole virtual machine, running the guest operating system and applications. The main advantage of VM-based virtualization is the high level of separation between guests and host: guest machines can have different architecture (e.g., x86 and ARM) and different operating system (e.g., Windows or Linux) from their host; furthermore, the same host can run multiple guests with different architectures and operating systems at the same time. On the negative side, the tasks of emulating guest hardware and running the guest operating system translate into a significant CPU and memory overhead.

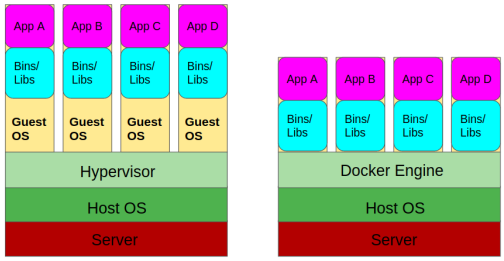


Fig. 1. High-level architecture of virtual machine-based virtualization (left) and container-based virtualization (right).

More recently, container-based virtualization has emerged as a lighter-weight alternative to VMs. As shown in Fig. 1, both the hardware and the operating system kernel are shared between host and guest applications. Isolation is obtained through operating system-level primitives such as namespaces and cgroups, controlling the parts of the execution environment (running processes, daemons...) and system resources visible to guest applications running within containers. The main advantage of container-based virtualization is its lower overhead compared to VMs. The main drawbacks include a lower level of isolation between host and guests (which can lead to security issues) and the impossibility of emulating a guest operating system different from the host one.

In this paper, we embark on the twofold task of (i) measuring the power consumption associated with VM- and container-based virtualization under a variety of workloads, and (ii) modeling how such consumption depends on the workload at hand. By fulfilling the first task, we can check to which extent the lower overhead touted by containers translates into lower power consumption. Perhaps more importantly, studying the link between workload, chosen virtualization technique and power consumption allows us to estimate the power consumption associated with a generic workload, beyond those we directly test.

The rest of this paper is organized as follows. We begin by describing our methodology in Sec. II, including our testbed

and the workloads we use. Then, in Sec. III we summarize the results we obtain from our real-world experiments. Finally, after summarizing related work in Sec. IV, we conclude the paper in Sec. V.

II. Methodology

We measure the power consumption associated with virtualization through a small-scale testbed. Whenever possible, we use commodity hardware and open-source software, so as to make our results as easy to reproduce as possible. This section describes the hardware (Sec. II-A) and software (Sec. II-B) we use in our testbed, as well as the real-world workloads we take into account (Sec. II-C).

A. Hardware

The host machine we use for both containers and virtual machines is an HP EliteBook 820 G3 laptop, equipped with a Intel Core i5-6200U processor (two cores, 2.3 GHz, 3 MByte cache) and 8 GByte of RAM. When testing client-server applications, we run the servers (within containers or VMs) on the laptop, and the clients on a separate computer, namely, a ThinkCenter M93p desktop. By doing so, we prevent client and server applications from interfering with one another, e.g., triggering thread preemption. The laptop and desktop are connected through a portable Gigabit Ethernet switch, and that switch is not shared with other computers. This ensures that the connection between clients and servers is consistently fast, and never represents a bottleneck for the application running.

Finally, we measure the power consumed by the laptop through an RCE PM600 power meter, displaying the total power the laptop draws from the grid. By using a power meter instead of estimates from the operating system, we obtain very precise and reliable measurements. However, this also means that special care is needed to tell the power consumption due to the workload apart from other contributions, e.g., the power consumed by the host operating system. To this end,

- before running any workload, we measure the idle power consumption of the computer;
- we subtract such value from the power consumption measured under each workload.

Additionally, we remove the battery from the laptop to ensure that no power is drawn to charge it.

B. Software

There are two main decisions we need to make concerning the software to use in our testbed: the operating system to use, and which VM- and container-based virtualization solutions to adopt.

The natural choice for the operating system is Linux, due to its broad support for all virtualization technologies. Among Linux distributions we chose Ubuntu, which offers the best balance between up-to-date packages, documentation, and overall system stability; specifically, we use the 16.10 LTS version, with kernel version 4.8.0-46-generic. As both the i5 processor and this kernel support hyper-threading, four threads can be run at the same time.

VM-based virtualization is a mature technology, with several available options, both commercial and open-source. We choose VirtualBox, on the grounds that it is the most popular among open-source ones. Both commercial (most notably, VMWare) and open-source (namely, QEMU) alternatives are sometimes reported to be marginally faster than VirtualBox; however, VirtualBox is more widely available than the other two solutions (VMWare requires a license, and QEMU – branded a "processor emulator" – is suited for narrower set of scenarios).

Choosing the reference container-based solution is easier. Docker is indeed the de facto standard, and arguably the primary reason why container-based virtualization attained its current level of popularity. It is an open-source application, and Linux support is its primary (though not exclusive) focus.

C. Workloads

We measure the power consumption under a set of real-world workloads, allowing us to get a glimpse of how popular, present-day applications would perform in a virtualized environment, and the power consumption we can expect from them. Specifically, we select two types of application that play a crucial role in the traffic of next-generation networks: video streaming and on-line gaming.

Video steaming. We employ the FFServer open-source streaming server, and the VLC free client. A varying number of servers, each running in its own VM or container, stream a video to a varying number of clients, running on the desktop computer as standalone applications. This setup, depicted in Fig. 2, allows us to assess how both the number of servers and the number of clients per server affect the power consumption.

On-line gaming. We select the popular on-line game Minecraft, as it offers an open-source server. In selecting the client, we need to ensure that (i) it can run on a headless machine, i.e., without graphical interface, and (ii) we can reproduce multiple times the same input, i.e., the same game. To this end, we combine the Java-based Minecraft client with the xdtool keystroke emulator, both running on the desktop computer.

III. Results and models

We now present the power and resource consumption associated with the real-word workloads described in Sec. **??**, i.e., the FFServer streaming application and the Minecraft game. As summarized in Fig. 2, we run the servers on our laptop, each in its own VM or container, and the clients on a separate desktop computer.

A. Video streaming: FFServer

In our first test, we deploy one FFServer server on our laptop (running within a VM or container) and use it to stream the same video to a varying number of VLC clients. It is important to highlight that we are modeling on demand streaming (à la YouTube) as opposed to real-time streaming (à la AceStream): each client plays an independent stream, and all traffic is unicast.

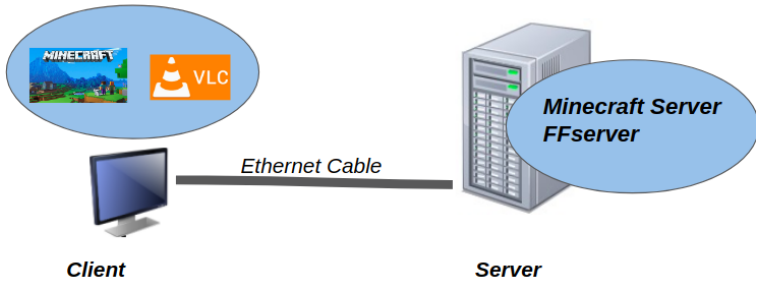
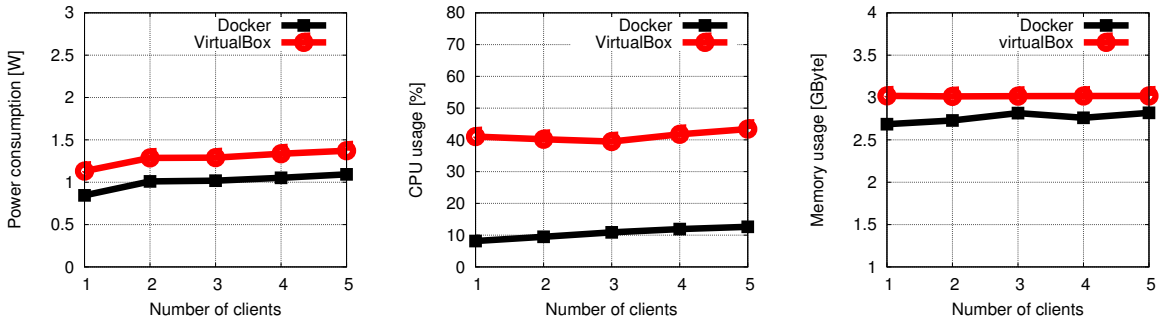


Fig. 2. Testbed setup for real-world workloads.

None the less, as we can see from Fig. 3, the number of clients only has a small impact on the resources consumed by the server; indeed, neither the CPU usage (Fig. 3(b)) nor the memory consumption (Fig. 3(c)) substantially increases with the number of clients. As a consequence, the power consumption (Fig. 3(a)) is essentially always the same. This effect is due in large part to FFServer's own scalability. Additionally, the native coding of the video (namely, H.264) was also supported by the client, and therefore no transcoding – a CPU-intensive operation – was necessary in our case.

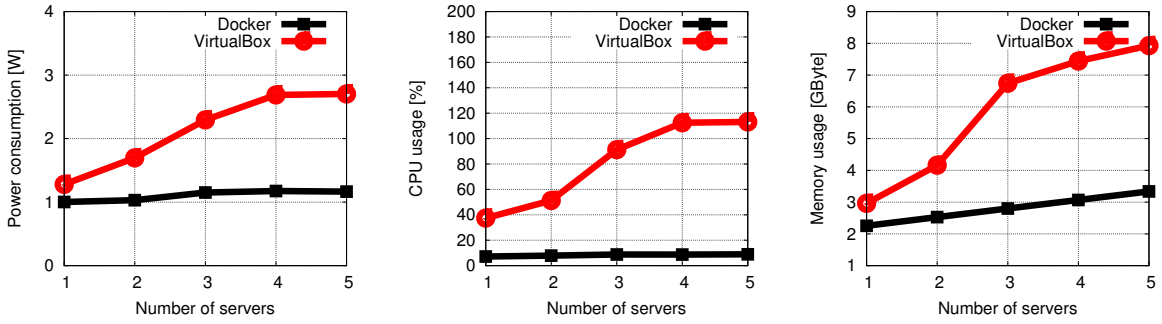
We now study the effect of having multiple servers running in parallel on the laptop, each in its VM or container, and each serving one client. This is relevant, for example, in MEC scenarios, where virtual (network) functions belonging to different services are often run separately, even if they perform the same task. The corresponding resource consumption is shown in Fig. 4.

Once more, we observe differences between Docker and VirtualBox that are not only quantitative, but qualitative. With Docker, five servers serving five clients consume approx-



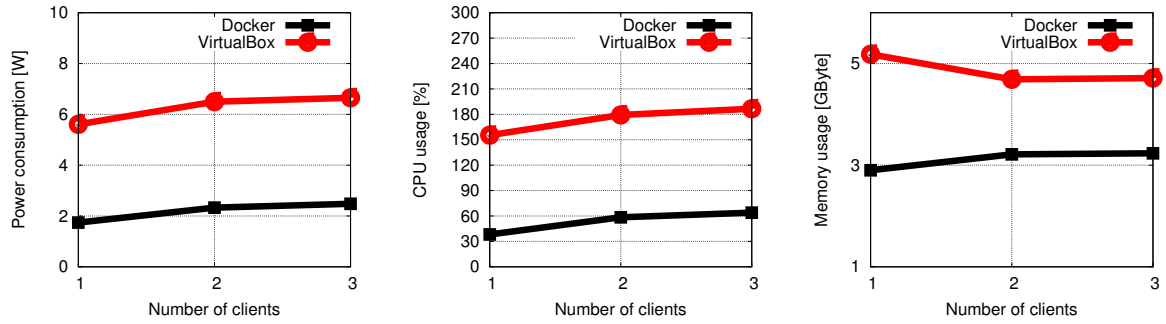
(a) (b) (c)

Fig. 3. FFServer, single-server setup: power consumption (a), CPU usage (b) and memory usage (c) as a function of the number of clients, for VirtualBox (red lines) and Docker (black lines).



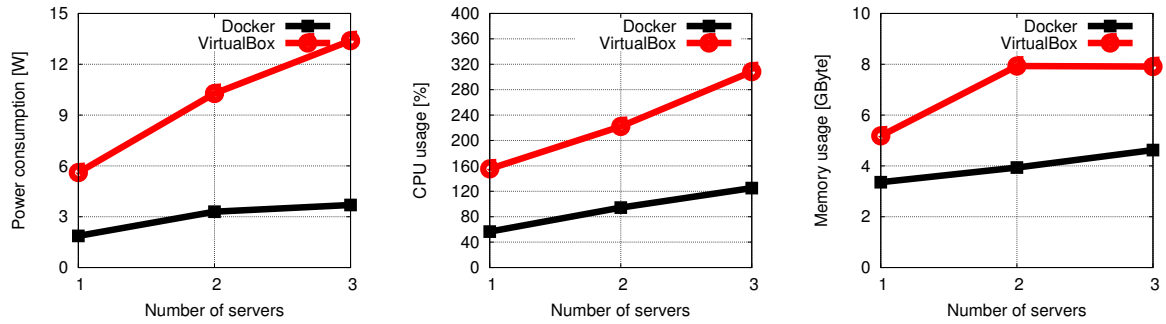
(a) (b) (c)

Fig. 4. FFServer, multiple-server setup: power consumption (a), CPU usage (b) and memory usage (c) as a function of the number of clients, for VirtualBox (red lines) and Docker (black lines).



(a) (b) (c)

Fig. 5. Minecraft, single-server setup: power consumption (a), CPU usage (b) and memory usage (c) as a function of the number of clients, for VirtualBox (red lines) and Docker (black lines).



(a) (b) (c)

Fig. 6. Minecraft, multiple-server setup: power consumption (a), CPU usage (b) and memory usage (c) as a function of the number of clients, for VirtualBox (red lines) and Docker (black lines).

imately the same amount of CPU (Fig. 4(b)) and power (Fig. 4(a)) of one server serving five clients; only the memory usage shown in Fig. 4(c) grows with the number of servers. Using VirtualBox, on the other hand, means that CPU usage and power consumption grow linearly with the number of servers: this behavior stems from the need to emulate separate virtual hardware and run separate operating systems for each server instance.

We can conclude that the overhead incurred by container-based solutions like Docker is not only lower than that of VMs, but also grows more slowly with the number of containers being run. Such a scalability justifies the interest that container-based virtualization has attracted as a key technology of MEC, and indeed, one of its enablers.

B. Gaming: Minecraft

We now move to the Minecraft game. Fig. 5 refers to the setup with one server and a varying number of clients, similar to Fig. 3. We can observe a steep increase in the utilization of resources as the number of clients increases, for both Docker and VirtualBox; this is due to the different nature of the application being run. Indeed, unlike video servers like FFServer, game servers like Minecraft have to:

- keep a detailed description of the world users are set in, and update it according to the user's actions;
- compute view of the world to serve to the user as it moves;
- transfer the updated view to the user's client.

The two tasks above imply a higher consumption of (respectively) memory and CPU compared to the FFServer case, and therefore a higher power consumption.

As far as the difference between VirtualBox and Docker is concerned, Docker still exhibits a substantially lower power consumption, (Fig. 5(a)), CPU usage (Fig. 5(b)), and memory usage (Fig. 5(c)). It is also interesting to notice, from Fig. 5(b), how the CPU overhead due to virtualization is higher than the load from the game server itself, intensive as it is.

Fig. 6 summarizes the resource consumption in the multi-server setup, when each user is served by its own server. Similar to Fig. 4, we can see a higher resource consumption than in the single-server setup; furthermore, the qualitative behavior is now the same for both VirtualBox and Docker. The latter has, however, a much lower overhead than the former, suggesting that the scalability advantage of container-based virtualization is present under all types of workloads.

IV. Related work

Measuring the power consumption of virtualization environments and applications running on them is the subject of several existing works. Kansal et al. [1] have developed Joulemeter, a tool to measure the energy consumption of a virtual machine and break it down as the sum of the individual power consumption of CPU, memory and disk. Krishnan et al. [2] have modeled the power consumption of virtual machines as a linear function of the number of CPU instructions and the number of Last Level Cache (LLC) memory misses.

Another VM power modeling technique is VMeter by Ata et al. [3], which is based on online monitoring of CPU, cache, disk and RAM. The model predicts instantaneous power consumption of an individual VM hosted on a physical node in addition to the full system power consumption. Yet another tool to measure power consumption of virtualized applications is presented by Comant et al. [4]. This paper [4] introduces a fine-grained monitoring middleware, which automatically learns an application-agnostic power model, which can be used to estimate the power consumption of applications. BITWATTS instances use high-throughput communication channels to spread the power consumption across the VM levels and between machines.

Dhiman et al. [5] have presented a system for online power prediction in virtualized environments based on Gaussian mixture models that use architectural metrics of the physical and virtual machines collected dynamically by our system to predict both the physical machine and per-VM level power consumption. Bertran [6] has proposed a system based on CPU and memory power models, relying on Performance Monitoring Counters (PMCs), to perform energy accounting in virtualized systems. Morabito [7] has presented an empirical investigation of different virtualization technologies (including containers) from the viewpoint of power consumption.

Kritwara et al. [8] have investigated power consumption and performance issues concerning memory and disk I/O in Xen and KVM virtualization environments. Avino et al. [9] have addressed the suitability of Docker in MEC scenarios by quantifying the CPU consumed by Docker when running two different containerized services: multiplayer gaming and video streaming. The have done tests by varying the number of clients and servers for both services. For the gaming service, the overhead logged by Docker increased only with the number of servers; conversely, for the video streaming case, the overhead is not affected by the number of either clients or servers. Fan et al. [10] have proposed the Energy driven AvataR migration (EARN) scheme to reduce the total on-grid energy consumption of GCN by considering the energy consumption of Avatar migrations.

V. Conclusion

We aimed to assess the power consumption due to VM- and container-based virtualization. To this end, we performed an extensive set of real-world measurements, using VirtualBox and Docker as reference technologies and a set of real-world workloads.

Through our measurements, we found that CPU usage is the main driver of the global power consumption, and the extra power consumption of container-based virtualization is not only lower than that of VM-based virtualization, but also grows more slowly as the workload increases. Additionally, we observed that that there is a penalty in energy consumption associated with creating a VM and then leaving it unused, while containers only consume system resources if the application they host is active. These findings suggest that container-based virtualization is an attractive technology for MEC scenarios, when large numbers of virtualized applications run on the same physical hardware and some of them may be inactive for some time periods, e.g., during user handover.

References

[1] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *ACM symposium on Cloud computing*, 2010.

[2] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, "VM power metering: feasibility and challenges," *ACM SIGMETRICS Performance Evaluation Review*, 2011.

[3] E. Ata, H. Bohra, and V. Chaudhary, "Vmeter: Power modelling for virtualized clouds," in *IEEE/ACM GRID*, 2010.

[4] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe, "Process-level power estimation in vm-based systems," in *IEEE/ACM GRID*, 2015.

[5] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using gaussian mixture models," in *IEEE/ACM GRID*, 2010.

[6] R. Bertran, Y. Becerra, D. Carrera, V. Beltran, M. Gonzalez, X. Martorell, J. Torres, and E. Ayguade, "Accurate energy accounting for shared virtualized environments using pmc-based power modeling techniques," in *IEEE/ACM GRID*, 2010.

[7] R. Morabito, "Power consumption of virtualization technologies: an empirical investigation," in *IEEE/ACM UCC*, 2015.

[8] R. Kritwara and P. Tandayya, "Comparison of disk i/o power consumption in modern virtualization," in *Computer, Communications, and Control Technology (I4CT), IEEE*, 2015.

[9] G. Avino, M. Malinverno, F. Malandrino, C. Casetti, and C.-F. Chiasserini, "Characterizing docker overhead in mobile edge computing scenarios," in *IEEE/ACM GRID*, 2016.

[10] Q. Fan, N. Ansari, and X. Sun, "Energy driven avatar migration in green cloudlet networks," in *IEEE/ACM GRID*, 2016.

Acknowledgement

This work is supported by the European Commission through the H2020 5G-TRANSFORMER project (Project ID 761536).