Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 1129 – 1139

doi:10.3906/elk-1801-284

Energy efficiency with an application container

Özmen Emre DEMİRKOL[1,*], Aşkın DEMİRKOL[2]
[1]Informatics and Information Security Research Center (BİLGEM), the Scientific and Technological Research Council of Turkey (TÜBİTAK), Gebze, Kocaeli, Turkey
[2]Department of Electrical and Electronics Engineering, Faculty of Engineering, Sakarya University, Sakarya, Turkey

Received: 31.01.2018 • **Accepted/Published Online:** 21.02.2018 • **Final Version:** 30.03.2018

Abstract: Efficient use of energy is an issue that the information technology (IT) world gives prominence to both in academia and industry. Cloud computing and the Internet of things, today's most popular subjects, make the efficient use of resources and energy even more important. Every year, millions of smart devices connected to the Internet increase the demand for data center capacity to provide service to those devices. This increases energy consumption in the IT sector. Thus, more efficient use of energy in these systems is of critical importance. The increase in the migration to cloud computing makes fast and efficient infrastructure-as-a-service and platform-as-a-service services provided by Internet service providers important. On the other hand, virtual machines have been used for a long time as an alternative to physical servers. The application containerization concept is shaping the virtualization world by offering faster deployment, reduced resource consumption, easier manageability, and reduced energy consumption, as demonstrated in this study. Our study shows that this new concept is more energy-efficient than virtualization technologies that are currently being used.

Key words: Virtualization, containerization, Docker, data center, energy efficiency

1. Introduction

In today's global scale, thousands of servers and millions of services are served by many service providers. In recent years, Internet of things and cloud computing technologies motivate servers to be aggregated and services to be offered as infrastructure as a service, platform as a service (PaaS), and software as a service by large data centers. Cloud computing has been expanding under the concepts of private, public, hybrid, and community clouds, as defined by NIST. The organizations researching and working on cloud computing are establishing huge data centers in different regions of the world and manage, or enable users to manage, these centers with software-defined data center logic. This increases the need for fast, scalable, measurable, flexible, and easily expandable infrastructures. However, such a demand inherently increases the need for energy efficiency.

According to Enerdata's 2014 results, the growth in energy demand in the IT sector has increased from 10 TWa to 20 TWa since the 1990s. Additionally, according to the "2015 List of countries by energy consumption" in Wikipedia, the annual energy consumption of countries with big data centers includes 5463.8 TWa for China in 2014, 4686.4 TWa for the USA in 2013, 3037 TWa for the EU in 2009, and 1016.5 TWa for Russia in 2012. These values show the annual energy consumption of countries in all fields. Nevertheless, the total amount of energy consumption in IT around the world is more than the total need of these leading countries. The Energy

*Correspondence: ozmen.demirkol@tubitak.gov.tr

Information Administration estimates that this need will rise to 40 TWa by 2040, with an annual average increase of 2.2%.

In this study, we investigate the effects of application containerization technologies on energy efficiency in data centers. This technology is being accepted as a new generation of virtualization and has become widespread in recent years in the field of virtualization, which is an important and indispensable element of cloud computing. This is an essential study on energy efficiency via application virtualization.

Nowadays, three types of servers are widely used:

BareMetal: Installing an operating system and later an application on a server.

In this work, we took BareMetal as a base line to understand whether hypervisor or container-based virtualization will yield the closest power consumption result. Because BareMetal is an OS that easily accesses hardware, its energy consumption is lower. Therefore, every test was applied on BareMetal and compared hypervisor and container virtualization.

Host virtualization: Installing a hypervisor on a BareMetal server and later on installing other kinds of operating systems and applications as necessary.

Application container: After installing an operating system on BareMetal, serving the application within an isolated package by separating at the kernel level where the package contains the elements of the operating system needed by the application (file system, libraries, and binaries).

In light of this preliminary information, we investigated how these three usages affect the target power usage effectiveness values in data centers and examined the advantages of each one. The rest of the paper is organized as follows:

Related work is given in Section 2. The Linux application container is described in Section 3. The environmental setup of the study and our test techniques are described in Section 4. Finally, the results and our conclusions are given in Sections 5 and 6.

2. Related work

Although power management is the focus of power proportional and green data center approaches in the physical world, it is handled under virtualization and hypervisor resource management in the software environment [4,5].

2.1. Physical world

Power proportional systems provide solutions that transfer the load to a particular source of a system and turn off the power in the remaining parts in order to conserve power. For instance, it distributes the load to certain portions of the disk unit and turns off some other parts in order to gain energy [1,6].

Most power proportional studies focus on turning off the disk and server for energy conservation. However, the reduction of the load from network devices and cooling systems [7], the distance of the power distribution lines, and the losses from AC/DC conversion play important roles in the total power consumption of a data center [2]. One other gain that is modeled by Google and increases energy efficiency from 92% to 100% is to use a small backup power battery for each server, rather than one central auxiliary power unit. A cooling power of 0.5 W is needed for cooling a watt spent by cooling systems. The most effective solution to this issue is using free-cooling systems rather than chiller type cooling systems.

2.2. Virtual world

Koh's [8] coallocation technique, which analyzes a system's characteristics at the disk, memory, and CPU levels; Gupta's approach for performance improvement [9]; and Pu's techniques [3] for decreasing performance interfaces on XEN hypervisors are among the studies that focus on energy efficiency in virtualization.

In addition, Nahomi's [10] virtual power management application is the counterpart of the studies by Deng [11] and David [12] on reducing energy consumption via voltage/frequency scaling optimization in memory. Moreover, the three approaches of Ye [13] towards the energy consumption of virtual hard drives use dynamic memory allocation for running virtual machines.

In the literature, there are no studies focusing on application virtualization. However, Liu and Zhao [14] and Dua et al. [15] study the ease of application, speed, savings in disk space, security, and isolation in the PaaS layer of cloud computing. Rey et al.'s work [16] on fault tolerance in Hadoop MapReduce with Docker and Stubbs et al.'s analysis [17] of serf node project that works on microservice infrastructure are among the previous literature on application containerization. In addition, the work by He et al. [18] proposes to use the application container infrastructure for end users on virtual servers via the elastic application container approach.

In this study, we focus on the effects of Linux application containerization on power consumption compared to server virtualization and BareMetal environments. There are two works in this area related to our study. One of them is Morabito's [19] work on energy consumption in hypervisor and container-based virtualization via benchmarking with some Unix commands. In this work, some tests are applied to CPU, memory, and NICs, but it lacks information about test time and data size that could affect the power consumption results. The second work is Piraghaj's PhD thesis [20] on cloud-based consolidation techniques and their effects on power consumption. As for the difference of these works with ours, we used a hard disk and a common application benchmarking test. The use of a hard disk I/O in our study is an important part of the power consumption. Moreover, all these hardware's power consumption results need to be evaluated using a commonly used application deployment benchmarking test. Additionally, our work also contains deficiencies not included with the related references as [19,20].

3. Linux application container

Operating-system–based virtualization allows users to access all necessary redundant resources. Since an operating system is designed to run all applications on itself, any component that does not lie within the objectives set by the user becomes unnecessary, but still requires management. These components also continue to consume energy. In addition, other factors, such as network settings and scaling, are left to the user most of the time. This approach not only requires the user to have out of focus information, but also makes it difficult for service providers to control. The hypervisor layer, to which virtual operating systems are connected, raises challenges, such as migration difficulties and vendor or hypervisor lock-in. Accordingly, the inefficiency of storage space is repeatedly seen in the form of the operating system or the applications continually creating the same file or block. This inefficiency is one of the main reasons for the extra energy consumed by the same I/O process through CPU and disk space while reading from or writing to disk. Differences between virtual machines and containers are shown in Figure 1.

Although this is not within the scope of this work, file system and block deduplication techniques have an effect on energy efficiency.

Without using all aspects of the operating system, application container is a technique that runs one or

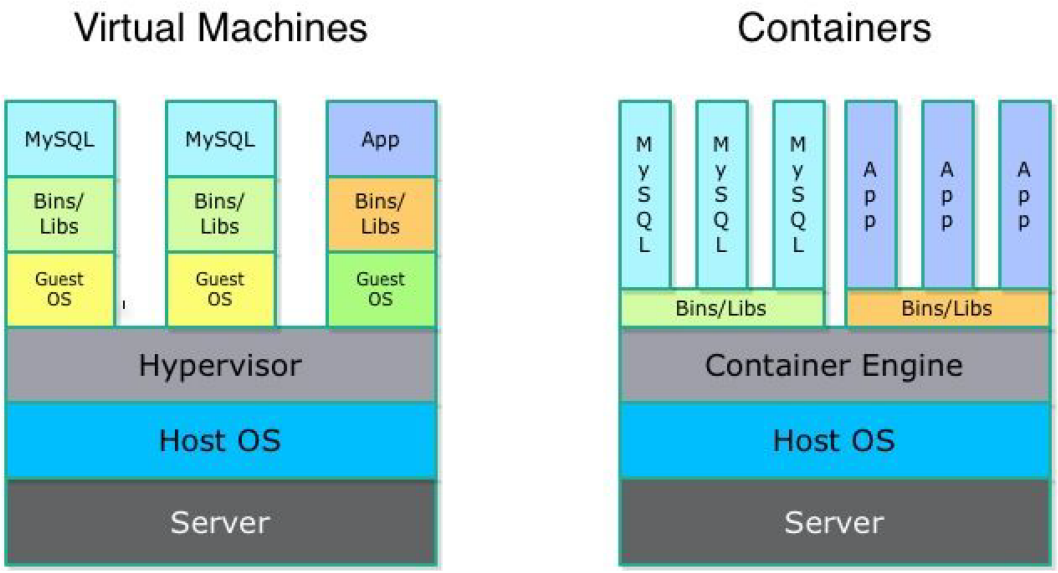

Figure 1. Virtual machines versus application containers.
(http://patg.net/containers,virtualization,docker/2014/06/05/docker-intro/)

more applications in a mold by isolating elements such as CPU, memory, block I/O, and network. Here, there is an isolation at the kernel level over the host operating system. These subsystems are functionally independent from the main system and other systems, and have their own process ID (PID), users, network structure, and file systems. In other operating systems, similar systems are called jails (FreeBSD), workload partitions (AIX), and containers (Solaris, Linux). In this study, we focused on containers that are used in the popular Linux operating system.

LXC was spread with Linux kernel 2.6.24, which uses an isolation system built on Cgroups and namespace functions. While Cgroups make some important features possible (like resource limitation, prioritization, accounting for usage of system resource, freezing on subsystem groups, checkpoint, and reboot), namespace presents an isolated PID poll for containers themselves. Five namespaces can be discussed:

Network namespace: Chance of making routing and firewall rules, special network topology for the container itself.

Unix time-sharing namespace: Chance of assigning hostname and domain name to a container.

Mount namespace: Chance of using a different file system inside of a container.

Inter-process communication namespace: Chance of making a special and isolated communication line between containers.

User namespace: Chance of making different users and groups from the base system to use in containers.

Container infrastructure is shown in Figure 2.

After the acceptance and widespread use of LXC in the virtualization world, approaches that developed this architecture and adapted it to the site started to increase. Most popular of those approaches are Docker, CoreOS, Mesos, and Kubernetes. Docker is in the leading position due to its number of developers, Docker HUB's ready container structure and wealth of content, and the ease of integration by other major architectures. The world's leading and USA-NSA–funded open source cloud management infrastructure (OpenStack) and the top cloud service provider (Amazon AWS) support the integration of Docker container in order to have a faster and easier infrastructure facility that consumes fewer resources.

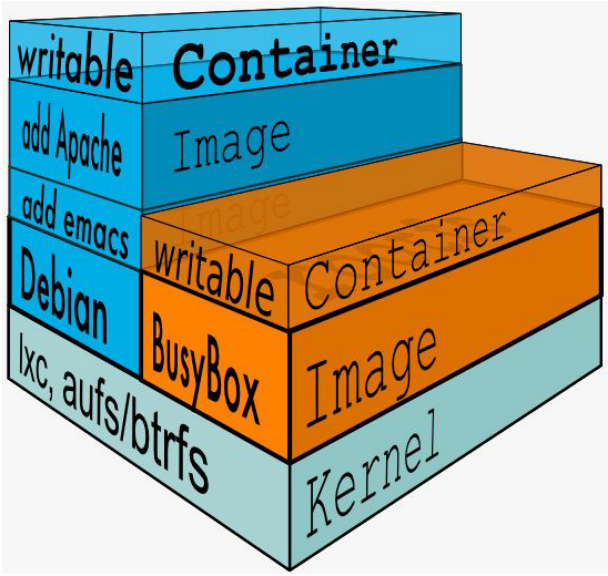

Figure 2. Application container structure. (https://www.docker.com/what-docker)

4. Environmental setup and approach

4.1. Physical environment

An HP Proliant DL380 G7 server was used for the test environment. The server has 2 × Intel Xeon X5650 @ 2.67 GHz (12 cores) processors, 18 × 4096 MB (in total 72 GB) DDR3-1333 MHz memory, 300 GB RAID10 HD, and a Broadcom NetXtreme II BCM5709 1 GB network card.

4.2. Software environment

The CentOS 6.7-final operating system, Linux kernel 2.32-573.8.1.el6.centos.plus.x86_64, libvirt-0.10.2-54, and qemu-kvm-0.12.1.2-2 were used for test virtualization, and container docker-engine-1.7.1-1 was used as the container.

4.3. Benchmark platform

Phoronix Test Suite, a free and open source benchmark software, was used. This software includes over 450 test profiles and over 100 test suites. It also contains important tests for CPU, memory, GPU, hard disk, network, etc. Moreover, the software provides data collected from different sensors on the hardware (such as heat and power) during benchmarking. The results of more than 3,800,000 Phoronix Test Suite tests published on OpenBenchmarking.org were utilized in order to decide on the test suites that would be used in our study. Test suites were chosen according to the recommendations of earlier users of the software and the rankings of the suites in achieving accurate results and full use of the system.

4.4. Used benchmarks

Timed Linux kernel compilation: It is used for CPU benchmarking and it measures how long it takes a Linux kernel to compile.

Aio-stress: It tests the system by asynchronously writing large files of 2048 MB to a hard disk in 64 KB blocks.

RAM speed: It allocates certain memory space and starts either writing to or reading from it using continuous blocks in sizes of powers of 2 from 1 Kb up to the array boundary.

LoopBack TCP network performance: It measures the performance of the system by generating 10 GB network traffic via 1 MB blocks.

Apache benchmark: It simultaneously sends 100 web requests, for a total of 1,000,000 requests, in order to test how many of them can be handled by the system. In contrast to other benchmarks described above, this is chosen to measure the performance of not only one component but all system components working together simultaneously.

All these benchmarks have not been used before in a study on energy efficiency. Here, they were used to test the system performance and produce results in units of req/s, MB/s, and s. Each test was done three times and their average was recorded. In order to measure the energy efficiency of the system, information gathered through energy sensors during testing was used. The average scores were used as the energy required by the system for the corresponding test. All tests were performed three times each for BareMetal, KVM, and Docker.

It is important to provide equal shares of system resources in all test environments. However, it is not clear how this was achieved in some previous BareMetal and virtualization comparisons [13,14,18]. A hypervisor cannot practically use all system resources because the host operating system also requires a portion of these resources in order to maintain the hypervisor and other basic functions. On the other hand, there is no layer on the host operating system and all divisions are done at the kernel level in container systems. Therefore, there is no need for resource allocation in these systems as opposed to hypervisors. We observed that the host operating system consumed 136 W of energy in our tests. KVM incurred a 13.2% system load and 154 W of energy consumption in order to run a host with the same specifications of the host operating system (CentOS 6.7), without any running services or extra work. In contrast, we observed that the system running on the Docker container system without any load required less than 1 W of extra power consumption. In the next section, the test results, and to what extent they are correlated with energy consumption, are evaluated.

5. Results

In this section, the results of five different benchmarks done on three test platforms and the rate of these results to energy consumption are presented.

In three of these five benchmarks, differences were observed in both the energy and performance values. However, the same energy consumption with different performance values is seen in the remaining two benchmarks. In order to interpret these results and to compare the energy consumption of the environments, the test performances for 1 W via the mathematical ratio of the performance values to energy consumption were calculated as follows:

$$\text{Value for } 1W = \frac{\text{Test Result}}{\text{P.C. value for test- P. C. for start}} \tag{1}$$

P.C. = Power Consumption

5.1. Timed Linux kernel compilation benchmark

As shown in Table 1 and Figure 3, BareMetal had the best results according to the CPU performance test, since it consumed 9% less energy than KVM and 1% less energy than Docker. In addition, Docker was 10% more efficient than KVM.

Table 1. CPU-based performance.

Timed Linux kernel compilation	BareMetal	KVM	Docker
Power consumption (W)	302.00	302.00	302.00
Build time (s)	73.95	81.70	73.99
Build time per W	0.445	0.492	0.446

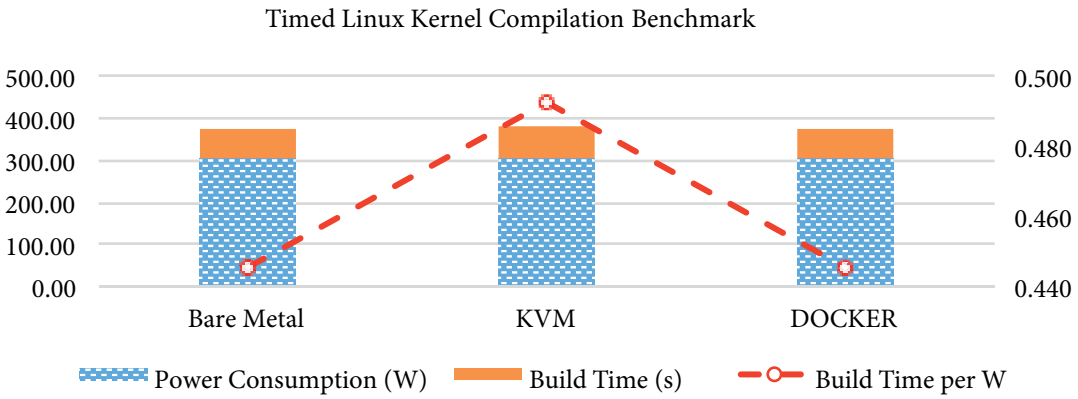

Figure 3. Results of timed Linux kernel compilation benchmarking test.

5.2. Aio-stress benchmark

As shown in Table 2 and Figure 4, based on the results of the HDD performance test, BareMetal uses energy 137% more efficiently than KVM and presents 162% energy efficiency as compared to Docker. On the other hand, KVM has 11% better energy efficiency than DOCKER.

Table 2. I/O-based performance.

Aio-stress	BareMetal	KVM	Docker
Power consumption (W)	146.00	162.00	162.00
HD performance (MB/s)	1706.92	1873.34	1693.78
HD performance per W	170.692	72.052	65.145

5.3. RAM speed benchmark

As shown in Table 3 and Figure 5, according to the benchmark where the systems' performances in reading from and writing to RAM were tested, BareMetal consumed 24% less energy than KVM. BareMetal had a slight (0.1%) energy efficiency in comparison to Docker. Docker also had a 23% energy consumption advantage over KVM.

5.4. LoopBack TCP network performance benchmark

As shown in Table 4 and Figure 6, when these three systems were compared under 10 GB network traffic, BareMetal had a 155% better energy consumption than KVM and a markedly better consumption (466%) than Docker. Moreover, KVM had 121% less energy consumption than Docker. This test demonstrates that although Docker stands out in handling network traffic compared to KVM, it fails in energy consumption.

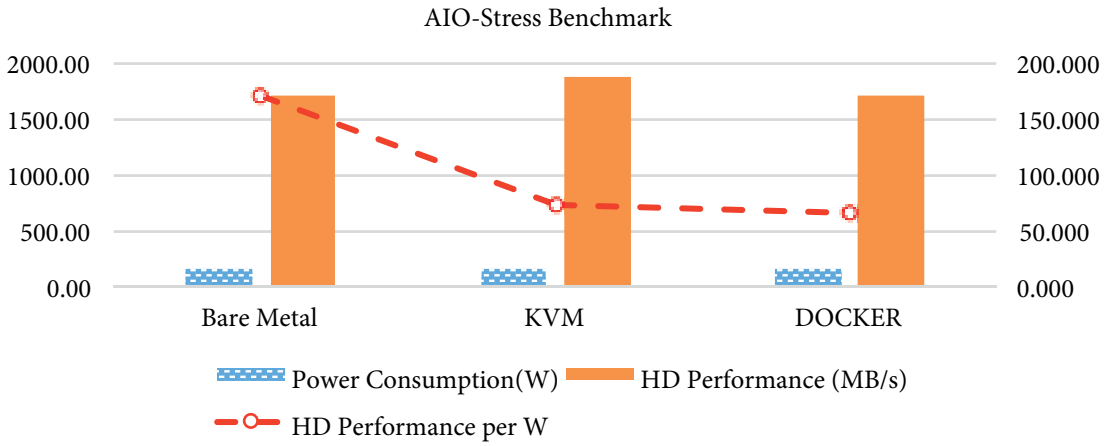

Figure 4. Results of aio-stress benchmarking test.

Table 3. Memory-based performance.

RAM speed	BareMetal	KVM	Docker
Power consumption (W)	232.00	208.00	232.00
RAM usage performance (MB/s)	15,490.68	9335.23	15,296.35
RAM usage performance per W	161.361	129.656	159.337

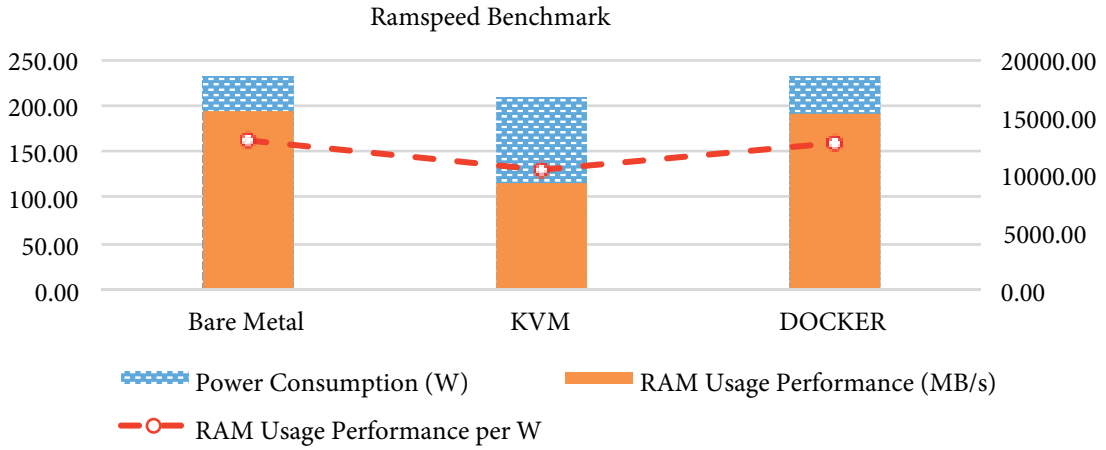

Figure 5. Results of RAM speed benchmarking test.

Table 4. Network traffic-based performance.

LoopBack TCP network performance	BareMetal	KVM	Docker
Power consumption (W)	144.00	174.00	184.00
10 Gb traffic (s)	15.60	29.02	16.55
10 Gb traffic per W	1.950	0.764	0.345

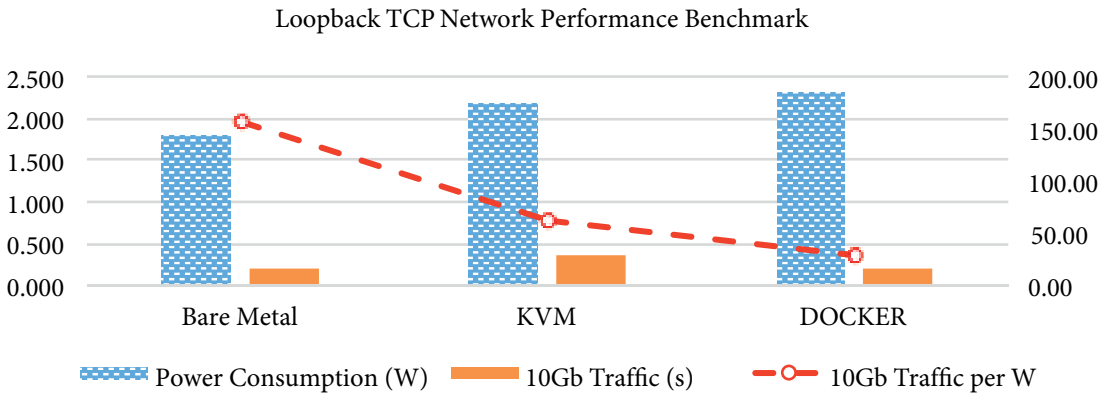

Figure 6. Results of LoopBack TCP network performance benchmarking test.

5.5. Apache benchmark

As shown in Table 5 and Figure 7, this test was designed to compare the energy efficiency of the environments when four main system resources were used at the same time. According to the results, BareMetal had 48% better energy consumption than KVM and 5% better consumption than Docker. Docker used energy more efficiently (41%) than KVM.

Table 5. Request response-based performance.

Apache	BareMetal	KVM	Docker
Power consumption (W)	204.00	204.00	204.00
Request per s	17,802.87	12,025.09	16,982.19
Request/s per W	261.807	176.840	249.738

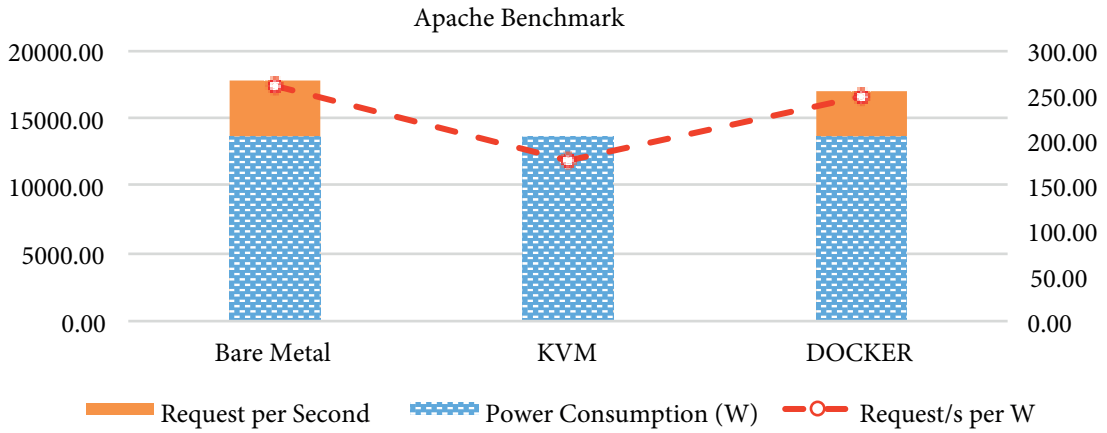

Figure 7. Results of Apache benchmarking test.

6. Conclusion

BareMetal had better results in all benchmarks tests, as expected. This can be explained by the fact that performance and efficiency are expected to improve once the application gets closer to the hardware. KVM,

where tests at the hypervisor layer are performed, had the next best energy consumption during hard disk and network tests. Docker, the application container layer, had better results than KVM in every test except the hard disk test. It is noteworthy to mention that DOCKER required high energy consumption in order to succeed in the network test. In the Apache benchmarking test, we compared the energy consumption rates of the test environments in general use. The results demonstrate that the Docker container system has better performance with less energy consumption than KVM. However, BareMetal beat both DOCKER and KVM in the same test.

Our study reveals that the application container technique is an important competitor against hypervisors from an energy efficiency point of view. If energy efficiency is applied in the IT world as mentioned in the introduction, a significant amount of energy can be recovered and transferred to new or other required fields. We argue that the use of the application container technique is necessary for the expansion of green and cost-effective IT. This study also reveals the requirements for further studies in hard disk usage and networking in application container techniques. Such further developmental studies will lead to improvements in energy efficiency.

References

[1] Narayanan D, Donnelly A. Write off-loading: Practical power management for enterprise storage. ACM T Storage 2008; 4: Article No. 10.

[2] Ton M, Fortenbery B, Tschudi. W. DC Power for improved data center efficiency. Report by Laurence Berkeley National Laboratory (LBNL), 2008.

[3] Pu X. Who is your neighbour: Net I/O performance interference in virtualized clouds' services computing. IEEE T Serv Comput 2013; 6: 314-329.

[4] Colarelli D, Grunwald D, Neufeld M. The case for massive arrays of idle disks (MAID). In: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing 2002; pp. 1-11.

[5] Pinheiro E, Bianchini R. Energy conservation techniques for disk array-based Servers. In: Proceedings of the 18th Annual International Conference on Supercomputing; 2004; pp. 68-78.

[6] Francis QZ, David FM, Devaraj CF, Li Z, Zhou Y, Cao P. Reducing energy consumption of disk storage using power-aware cache management. In: Proceedings of the 10th International Symposium on High Performance Computer Architecture; 2004; p. 118.

[7] Zhu Q, Chen Z, Tan L, Zhou Y. Hibernator: Helping disk arrays sleep through the winter. In: Proceedings of the Twentieth ACM Symposium on Operating Systems Principles; 2005.

[8] Koh Y. An analysis of performance interference effects in virtual environments. In: Proceedings of the IEEE International Symposium on Performance Analysis of Systems & Software; 2007; pp. 200-209.

[9] Gupta D. Enforcing performance isolation across virtual machines in Xen. In: Proceedings of the ACM/IFIP/USENIX International Conference on Middleware; 2006; 342-362.

[10] Nathuji R, Schwan K. Virtualpower: coordinated power management in virtualized enterprise systems. In: Proceedings of the Twenty-first ACM SIGOPS Symposium on Operating Systems Principles 2007; 265-278.

[11] Deng Q, Meisner D, Ramos L, Wenisch TF, Bianchini R. Memscale: active low-power modes for main memory. In: Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems; 2011; 225-238.

[12] David H, Fallin C, Gorbatov E, Hanebutte UR, Mutlu O. Memory power management via dynamic voltage/ frequency scaling. In: Proceedings of the 8th ACM International Conference on Autonomic Computing; 2011; pp. 31-40.

[13] Ye L, Lu G, Kumar S, Gniady C, Hartman JH. Energy-efficient storage in virtual machine environments. In: Proceedings of the 6th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments; 2010; pp. 75-84.

[14] Liu D, Zhao L. The research and implementatıon of cloud computıng platform based on Docker. In: Proceedings of the 11th International Computer Conference on Wavelet Active Media Technology and Information Processing; 2014; pp. 475-478.

[15] Dua R, Raja AR, Kakadia D. Virtualization vs containerization to support PaaS. In: Proceedings of the 2014 IEEE International Conference on Cloud Engineering; 2014; pp. 610-614.

[16] Rey J, Cogorno M, Nesmachnow S, Steffenel LA. Efficient prototyping of fault tolerant Map-Reduce applications with Docker-Hadoop. In: Proceedings of the 2015 IEEE International Conference on Cloud Engineering; 2015; pp. 369-376.

[17] Stubbs J, Moreira W, Dooley R. Distributed systems of microservices using Docker and Serfnode. In: Proceedings of the 7th International Workshop on Science Gateways; 2015; pp. 34-39.

[18] He S, Guo L, Guo Y. Elastic Application Container. In: Proceedings of the IEEE/ACM 12th International Conference on Grid Computing; 2011; pp. 216-217.

[19] Morabito R. Power consumption of virtualization technologies: An empirical investigation. In: Proceedings of the IEEE/ACM 8th International Conference on Utility and Cloud Computing; 2015; pp. 522-527.

[20] Piraghaj SF. Energy-efficient management of resources in enterprise and container-based clouds. PhD, University of Melbourne, Melbourne, Australia, 2016.