Monitoring the Energy Consumption of Docker Containers

Mehul Warade[1], Kevin Lee[1], Chathurika Ranaweera[1], and Jean-Guy Schneider[2]
[1]*School of Information Technology, Deakin University, Geelong VIC 3220, Australia.*
[2]*Faculty of Information Technology, Monash University, Clayton VIC 3168, Australia.*
mehul.warade@research.deakin.edu.au

***Abstract*—Containers are an increasingly used mechanism for providing low-cost, lightweight, portable, standalone application deployments, particularly for service orchestration. Docker provides container technology that enables a single host to isolate several applications and deploy them rapidly in different environments. The increasing demand for container applications and the growing popularity of Docker has motivated extensive research into evaluating the performance, energy consumption, and running cost of Docker-based computation. This paper investigates the energy footprint of Docker containers and workloads. We take a practical approach to measuring the energy consumption in common web and database Docker containers under various workloads. We also investigate the factors that affect the energy consumption of different Docker containers. The study aims to motivate research in energy-efficient container development.**

***Index Terms*—Virtualization, Containers, Docker, Energy**

I. Introduction

Docker containers provide an infrastructure for lightweight, deterministic, and manageable isolated containers enabling agile computing resources. Containers are a form of virtualization where applications can run in isolated environments with pre-determined dependencies [1]. Docker containers can launch quickly and with lower overhead as compared to Virtual Machines. It is common for Docker containers that contain services to be deployed multiple times daily in support of thousands of users and on hosts with very different hardware [2]. The computing resources available for these containers can be easily managed. This leads to the highly scalable nature of the containers. Being lightweight, Docker containers can be used in conjunction with one another without a huge impact on their performance.

Docker allows for the seamless development and deployment of applications as it ensures the dependencies and makes sure that of consistency of the development environment will be the same as the deployment environment. According to a Docker report in 2018, users were running around 154 individual Docker containers on a single host which was 50% higher than that of the previous years [3]. 83% of the production environments tested in the report of 2018 made use of Docker for deployment [3]. Although virtualization has been accepted for many years, the ease of use of Docker containers has led to massive growth in its usage.

Docker has been gaining popularity due to its multiple benefits over traditional deployments or virtualization techniques. The increasing use of Docker for deployments has led to research in performing measurements, optimizations, and improvements of Docker workloads. Docker containers can be very efficient in terms of resource utilization, but they can also consume a significant amount of energy if not properly managed [4]. Much of the research until recent years looked at measuring the overheads of Docker, its impact on performance, and the comparison between bare-metal deployment versus Docker deployment. The research was aimed towards improving the performance of Docker containers with not much notice given to their energy consumption [5]. In the recent past, the energy consumption or carbon footprint of computing systems has started to gain the attention of researchers as we are gearing towards global energy crisis [6]–[9].

Despite Docker becoming one of the most used modern deployment technologies; there is very little information on the energy consumption of Docker containers and Docker workloads that can help in developing or making informed decisions about how to optimize the Docker deployments [10], [11]. There is a need for a standardized way to measure, analyze and improve the energy consumption of Docker containers. Measuring the energy consumption of the Docker workloads can also help in understanding how energy consumption correlates to performance [12].

The aim of this study is to begin to understand and motivate research into analyzing and optimizing the use of Docker containers with respect to their energy consumption. This would promote energy-efficient container deployments and a large proportion of computation infrastructure. It also aims to provide important additional context metrics for deployment managers to make decisions that include energy. The key contributions of the research presented in this paper are: i) proposing an approach for measuring the energy consumption of Docker containers. ii) providing in-depth analysis of different factors that affect the energy consumption of Docker usage iii) providing the motivation for improving the energy efficiency of Docker containers.

The remainder of this paper is structured as follows. Section II provides background on container virtualization, Docker, and energy-aware research in this area. Section III presets the experimental setup used to conduct this study. Section IV presents experiments that observe the energy impact of a range of standard docker containers and varying workloads. Section V presents an in-depth analysis of different factors

that affect the energy consumption of the machine. Finally, Section VI provides conclusions and future work.

II. Background

Virtual Machines provide virtualization and resource isolation at the cost of being resource-intensive and high overheads. Containerization helps solve the issue as it allows organizations to quickly, efficiently, and successfully deploy their applications. As the world becomes increasingly digitized, organizations are deploying applications rapidly when needed. This has raised the need to monitor and optimize the energy consumption of their deployments [13]. This can help to develop more sustainable applications that use Docker to have less energy footprint. The remainder of this section provides a background on (i) Containers, (ii) Docker, and iii) Related work in Energy-Aware Docker Computation.

A. Containers

Containers are a type of virtualization technology that isolates an application or process from the rest of the operating system (OS) so it can run without affecting other parts of the system [1], [2], [10]. A container can be considered a package of multiple processes that are running in an isolated environment along with all the code and the dependencies. This isolation makes containers an attractive option for running multiple applications or processes on a single OS instance, as it ensures that any issues with one container will not impact the others.

Containers have been around for many years, but they have only recently gained popularity due to the rise of microservices and container orchestration tools like Docker Swarm and Kubernetes [14], [15]. Containerization offers many benefits, including portability, isolation, and ease of use. They are extremely lightweight and can be instantaneously started, managed, or stopped with a single command.

These containers are managed and controlled by container engines such as Docker, rkt, runC, Containerd, LXC, etc. These container engines allow users to easily start multiple containers with a single command. They also provide other features such as statistics, resource management, removal, and deployment of the container to the cloud.

B. Docker

Docker is a container virtualization technology [16]. Docker is a tool that enables developers to easily create, deploy, and run applications in a container. A container is a self-contained, isolated environment that contains all the necessary files and dependencies for an application to run.

As seen from Figure 1, Docker allows multiple applications to run seamlessly on a single server. Each application along with its dependencies is isolated in its own container, which makes it easy to manage and update them. Docker is popular because it makes it easy to package and ship applications. Developers can simply create a container, add their application code, and then ship it off as an image that can be instantiated on another server. This makes it easy to deploy applications in a consistent and repeatable manner.

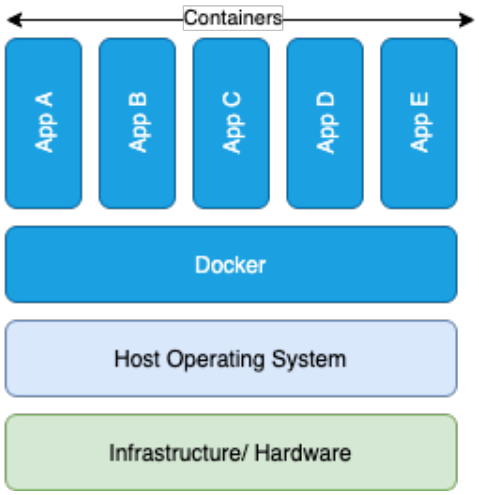

Fig. 1. Docker: Container-based architecture

A Docker container is a custom isolated system including all the base requirements of an Operating System such as filesystem, libraries, and dependencies. A Docker image contains a filesystem, which contains everything needed to run an application - all dependencies, configurations, scripts, binaries, etc. The image also contains other configurations for the container, such as environment variables, a default command to run, and other metadata.

The Docker daemon (dockerd) is a container management engine. It is responsible for starting, running, managing, and removing the containers and the system's resources. Docker is being increasingly used in industry as it provides highly unique benefits such as 1) Fast and Easy Configuration; 2) Security Management; 3) Easy resource management; 4) Application isolation; 5) Development of Swarm Applications; 6) Rapid Scaling and Deployment of Systems, and 7) Reduced Deployment Size [17].

A Docker container can be easily deployed using a single configuration file called a Dockerfile. A Dockerfile is a text file that contains instructions for how to build a Docker image. It contains all the information such as the location of the code, its dependencies, the base image, the default command to run, etc. A docker image is dependent on the Host Operating System on which it is realized or executed.

C. Energy Aware Docker Workloads

As Docker has been gaining popularity in the field of industry as well as research, multiple attempts at optimizing the Docker workloads have been conducted. These optimizations are in both - the performance and the energy consumption of the Docker containers. This section focuses on the literature in the field of Energy Aware Docker Workloads.

Docker containers consume a lot of energy if not properly configured or managed. Even though the main contributor towards the energy consumption of a Docker container is its

CPU load, [18] identified that there are other components such as the strain on the host Operating System that can contribute towards the increased energy usage of a container. These components need to be considered during the analysis and optimization of Docker containers.

According to a recent study, the average energy consumption of a Docker container is about 0.15 kWh [12]. This does not seem like a lot but it can add up quickly as usually multiple Docker containers are used together. A task scheduling algorithm that takes into consideration the current energy consumption of the containers has been developed to handle requests in real-time and schedule the requests in an energy-balanced manner [19].

A Workload aware Energy Efficient Container (WEEC) brokering system is introduced for Docker containers with the aim to reduce their energy consumption in Docker-based cloud data centers [20]. A Docker-based energy management system (DEMS) architecture is developed as an improvement to the traditional energy management system (TEMS) in order to fix the issues of slow deployment and low flexibility [21]. The DEMS reduces the number of web releases of a container by 3 times and the workload by 2 times.

An approach called brownout is proposed to dynamically activate or reactive optional containers in data centers [22]. This approach was able to reduce about 10% - 40% energy consumption of the micro-services hosted by the data center. Similarly, an Energy-Aware scaling algorithm was developed to dynamically load balance the requests [23], [24]. The algorithm is able to spawn new containers during heavy loads and kill existing containers in order to save energy based on certain thresholds [23], [25].

Energy consumption increase due to the Docker containerization was compared between different Docker workloads [12]. A comparison between the energy consumption of different virtualization technologies such as Virtual Machine, Docker, and Kubernetes has been conducted [26], [27]. The energy consumption of a single web app container has been captured in response to the scaling and balancing of the load [28].

Background and Literature in the field of containers, Docker, and energy-aware Docker Computing have been presented in this section. The majority of the energy-aware work has been conducted in relation to a single Docker workload and/ or Docker workloads deployed in the cloud. The current research in the field of energy-aware Docker Computing is generic in nature and tries to reduce or monitor the energy consumption of the whole Docker as a whole. They do not profile or analyze the energy footprint of common Docker workloads. The study presented in this paper addresses this issue and provides a comparison between multiple Docker workloads and investigates the underlying factors that affect the energy consumption of the workloads.

III. Experiment Setup

For the purposes of this study, the energy consumption of multiple Docker containers with different workloads running on a single computer has been measured. An initial experiment to showcase the energy monitoring technique used in this study has also been presented here. In this section, the experimental setup and the components used to generate, collect, store, and analyze the energy and performance data are presented.

A. Hardware

1) Host Computer: In order to translate the results of this study to real-world data centers, an Intel-based x86 NUC was chosen as the host machine to run the Docker containers with varying workloads. The Operating System installed on the Intel NUC is Debian-based Linux Mint 20 (https://linuxmint.com/, accessed on 4 December 2022). The configuration of Intel NUC is 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz with 4 cores or CPU and 8 threads. Intel NUC corresponds to computers that are normally used to run Docker containers.

Fig. 2. Energy monitoring plugs

2) Energy Data Collection using Smart Plugs: The host machine is powered using the main power plug via a smart energy monitoring plug. The smart plug has an integrated ESP8266 [29] chip that allows the smart plug to connect to the local network and enable the collection of energy data using application programming interface (API) calls. Internally, the smart plugs are collecting energy consumption using the HLW8032 energy meter sensor. The smart plug is running the latest version of ESPHome Firmware. The firmware has been modified to measure and deliver the energy data every 500ms. This data is collected at a set interval and stored in the database.

There are different endpoints that can be queried to get different measurements from the smart plug. The API endpoint `smart_plug_v2_current` provides us with the current in Ampere. Similarly, the endpoints `smart_plug_v2_power`, `smart_plug_v2_energy`, `smart_plug_v2_voltage` provide the consumption data such as the power (in Watts), Energy (in kWh) and Voltage (in volts) respectively. These endpoints return the data in a JSON format which is then parsed and stored in the database.

B. Software

1) Energy Collection: FEPAC [13] was modified to collect energy data from the Smart plugs using API calls. FEPAC

identifies the smart plug using the IP address of the smart plug when connected to the local network. The energy data is then collected and stored in a database every 500ms. The data in the database is synced with the computation on the host machine via the timestamp field. The synced data is then analyzed and important conclusions are made.

2) Docker: The experiment platform is running Docker Daemon version 19.03.8, build afacb8b7f0 [16]. For the experiments presented in this paper, the Docker daemon uses the default configuration. The versions of the Docker images used in this study, use the most recent stable versions from the DockerHub repository [30].

IV. Energy Consumption of Docker Workloads

Section III presented the experimental setup for the experiments conducted throughout this study. The section also confirmed the accuracy of the energy monitoring device (energy-monitoring power plugs) being used in this study. Docker containers are being used everywhere, right from small applications to hosting full-fledged enterprise solutions.

Dockers are highly scalable and can perform really well under stress [31]. Docker containers are also able to utilize the full computing resources of their host [32]. In this section, a number of workloads and experiments were conducted on different web and database Docker containers to test their performance and measure their impact on energy consumption.

A. Measuring the energy consumption vs CPU load

An initial experiment to measure the change in energy consumption of the host machine with respect to varying workloads was conducted to confirm the proper and accurate functioning of the energy monitoring device. Energy consumption data was collected every 500ms and the load on the host machine was gradually increased and decreased by maxing out the threads using the command '$yes > /dev/null$'. Every time this command is executed, a thread on the host machine is maxed out that outputs the string 'yes' continuously.

The execution data of the experiment is shown in Figure 3. The X-axis indicates the execution time (in seconds). The left Y-axis denotes the instantaneous energy consumption (in watts) of the host machine at that particular execution time. The right Y-axis shows the CPU usage (in percentage). The orange lines denote the number of threads being maxed out (also denoted by T0-T8 at the top).

Based on the data from Figure 3, the energy consumption of the host machine increases gradually until the 4 threads are maxed out on the host machine. After that, even though the CPU usage is increasing as we max out more threads, the energy consumption is stable. This is expected as the host machine is quad-core (i.e. 4 cores and 8 threads). This means that 4 CPUs can perform 8 tasks parallelly. The base energy consumption of the host machine is around 3.75W and the maximum it consumes is around 47W when the load is maxed out. The increase in energy can be seen before the increase in CPU usage with a delay of around 2 seconds. This experiment successfully shows that smart plugs can be used to accurately

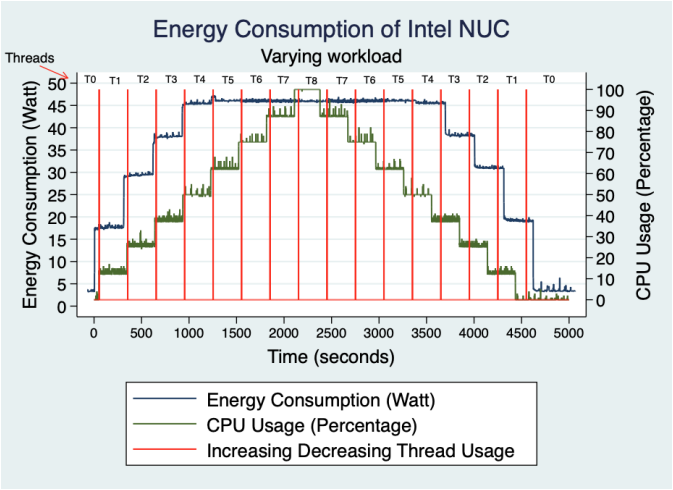

Fig. 3. Energy consumption of Docker container under stress increasing decreasing threads

measure the energy consumption of any device connected to it. Also, it is recommended to use all 8 threads on the host computer as compared to just using 4 threads.

B. Energy Overhead of Docker

Standard execution of any workload on Docker containers includes starting the Docker daemon and the service in the background and then building and/or running the containers to perform a certain task. This experiment denotes the initial overhead of starting the Docker daemon and service in the background. The data from this experiment showcases the initial impact of Docker on the energy without running any specific workload on the computer. Based on the data from this experiment, if the initial energy overhead of Docker is too high then it might be considered during the calculation of any future experiments.

Figure 4 illustrates the energy consumption of the host machine before and after turning the Docker daemon on. The X-axis indicates the execution time (in seconds) and the Y-axis denotes the instantaneous energy consumption of the host machine (in watts).

The energy consumption of the host machine was collected for a few minutes before and after turning the Docker daemon on. It can be seen that the base energy consumption of the host machine is around 3.75 watts. As soon as the Docker daemon is turned on (denoted by the Orange line), the energy consumption of the host machine increased to around 18.25 watts for a few seconds and then it came back to the mean average energy consumption. The difference in the idle energy consumption of the host machine before and after the Docker daemon is minimal. This is expected as once the Docker daemon is turned on, it does not actually use any resource of the host until a container is activated. Due to the minimal and short-lived increase in energy consumption of the computer when turning the Docker daemon on, this energy is not considered for any future experiments.

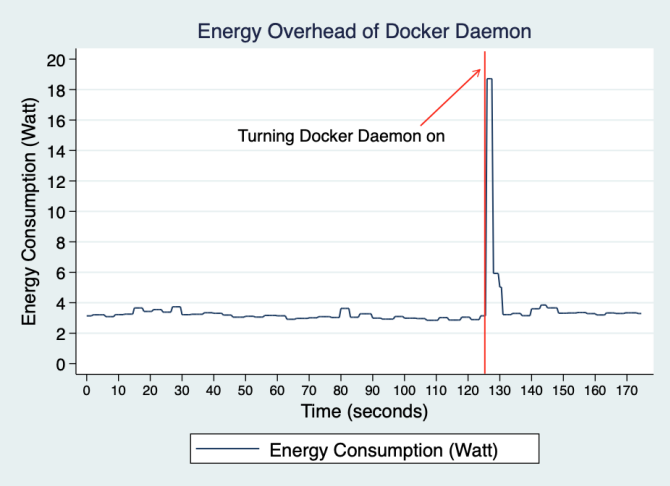

Fig. 4. Energy Consumption of Docker daemon turning on

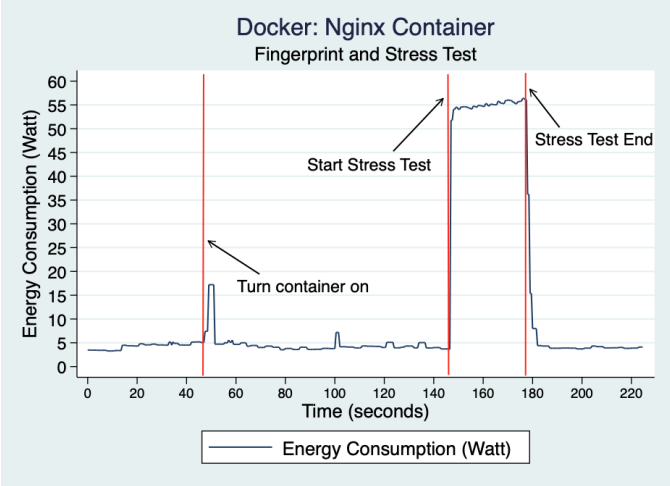

Fig. 5. Energy Consumption of Nginx Docker container turning on and under stress

C. Web server container

A study showed that a few of the top most used technologies running using Docker are web servers and database servers [33]. Out of the top 10 technologies deployed using Docker, 5 are database-related and 3 are web hosting-related technologies [33].

In this Section, two web-based Docker technologies (Nginx and Apache) are compared. A number of different experiments were conducted to measure the performance and energy impact of these containers. A simple HTML web page of size 615 bytes is being served using these web technologies. The web servers are stress tested. The stress test includes sending a huge amount of requests to the web servers and measuring the latency by which the web page or the request is served. This test also keeps the connection between the old requests open to further stress the web server.

1) Web technology fingerprinting and stress test: Fingerprinting web technologies include measuring the energy consumption of starting the containers. This is done mainly to check the overhead of the container. The containers are then subjected to a standard stress test using ***wrk2*** benchmarking tool [34]. For the purposes of this study, the benchmarking tool is requesting 500,000 requests to the web server every second and keeping 500 concurrent connections alive.

Figures 5 and 6 illustrate the energy consumption of the host machine during the fingerprinting and stress test of the Docker web technologies. The X-axis indicates the execution time (in seconds) and the Y-axis denotes the instantaneous energy consumption of the host machine (in watts).

The energy consumption of the host machine was collected for a few minutes before and after the experiment. The time when the Docker containers were started and the stress test was conducted is marked in Figure 5 and 6 (denoted by the Orange line). A spike in the energy consumption of the host machine can be seen which is distinctly greater than the base energy consumption. The energy overhead of the Nginx container is greater than that of the Apache Container as an increase in

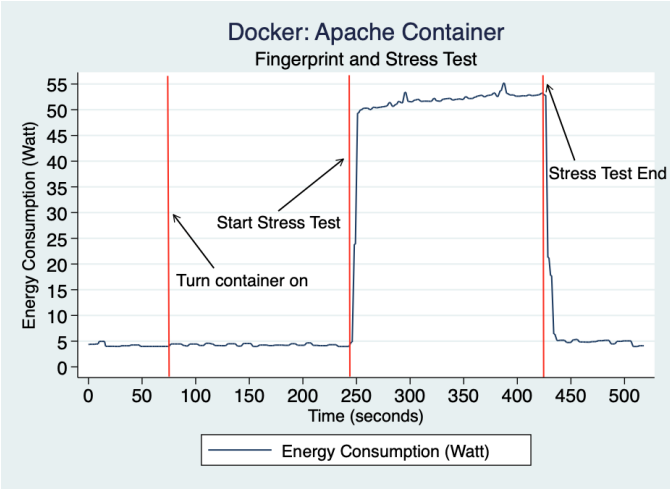

Fig. 6. Energy Consumption of Apache Docker container turning on and under stress

energy consumption can be when the Nginx container is turned on as compared to the Apache container.

Both containers consume around the same amount of energy during the stress test. The energy consumption range of both containers is between 50 to 56 watts. The Nginx server served a maximum of 114,717 requests per second and Apache served around 38,636 requests per second. Nginx container can be considered more energy efficient as it consistently and without any latency served more requests made by the benchmarking tool in contrast to Apache.

2) Varying workloads on Web Docker Containers: In this experiment, the same benchmarking tool is used to stress test the servers using varying workloads. The only change in the configuration is the number of requests requested per second from the server. The number of requests gradually increase from 10,000 requests per second to 100,000 requests per second. The data from this experiment will help co-relate energy with the performance of the Docker containers. This

is done to check the impact of the workload on the energy consumption of the Docker web server.

Figures 7 and 8 illustrate the energy consumption of the host machine during the experiment. The X-axis indicates the execution time (in seconds) and the left Y-axis denotes the instantaneous energy consumption of the host machine (in watts). The right Y-axis denotes the requests-per-second requested by the benchmarking tool. It should be noted that the benchmarking tool requests more than what the web server can process just so as to stress it to its maximum.

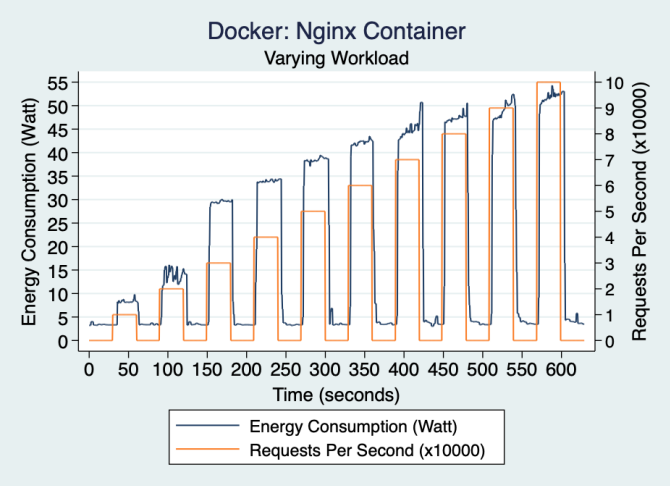

Fig. 7. Energy Consumption of Nginx Docker container under varying workloads

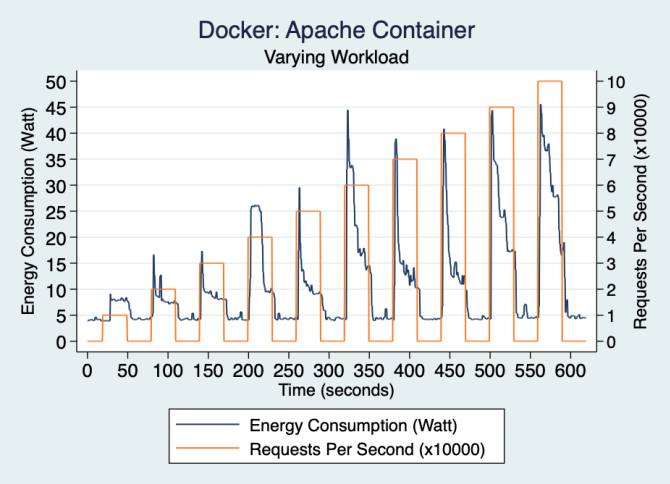

Fig. 8. Energy Consumption of Apache Docker container under varying workloads

Figures 7 and 8 present distinct differences between the energy consumption of the Nginx and Apache Docker containers under varying workloads. The Nginx container has consistent behavior in terms of energy consumption and is consistently serving the requests made by the benchmarking tool. A clear increase in energy consumption can be seen during each benchmark experiment. Also, the peak energy consumption gradually increases as the workload increases.

In Apache containers, many of the requests were of high latency and some even failed. This behavior is confirmed by the energy consumption of the Docker container during the benchmark test. The energy consumption during each test spiked instantaneously and then gradually decreased during the benchmark indicating that the container was not under stress for the whole duration of the experiment. Due to this, multiple requests were failing or not being served consistently. A more in-depth understanding of the Apache web server and its internal load balancing is required to deduce why this behavior was observed.

3) Nginx vs Apache: Inconsistent behavior in terms of energy consumption and performance was observed between the Nginx and Apache Docker containers in the previous Section. Apache server drops more requests than Nginx and due to that the energy utilization data is skewed. A fair comparison between the energy consumption of the containers/ servers can be done by considering the energy consumption of the containers when their successful performance (successful completion of requests) is compared.

Stress tests similar to the previous Section were conducted. The stress tests provided the number of successful requests being served by the server and the number of failed/ dropped requests. This number of successful requests served by both servers was compared for varying loads of stress.

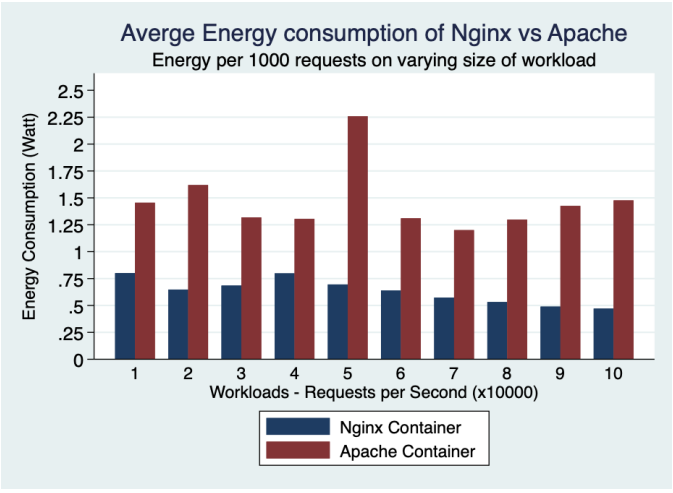

Fig. 9. Averge Energy consumption of Nginx vs Apache

Figure 9 illustrate the energy consumption of the host machine during varying workloads as compared with every 1000 successful requests. The X-axis indicates the requests per second requested by the benchmarking tool (in multiples of 10,000 requests) and the Y-axis denotes the average energy consumption of the container per 1000 successful requests (in watts).

As seen from Figure 9, the Nginx Docker container is more energy efficient as compared to Apache for different workloads. Nginx has also outperformed the Apache web

server in terms of memory, latency and CPU load due to its inherent architecture [35], [36]. A study showed that Nginx is around 2.5 times faster than Apache web server [37]. Similar, results can be seen in terms of the energy consumption of the Docker containers. Apache Docker containers consume around 2 to 3 times more energy than Nginx to successfully serve 1000 requests.

D. Database Container

In the previous Section, popular Docker-based web technologies have been compared with each other in terms of their performance and energy consumption. The second most used Docker technology after Web server is Databases. In this Section, two widely used Database Docker technologies (Mongo DB and Postgres) have been compared for their overhead and a stress test has been performed with a focus on energy consumption.

The energy consumption of two Database based Docker containers has been compared for the overhead and the stress test. The stress test is conducted using the ***POC Driver*** [38] for Mongo DB and ***pgbench*** [39] for the Postgres.

Figures 10 and 11 illustrate the starting and stress testing of these Docker containers. The X-axis denotes the time in seconds and the Y-axis denotes the instantaneous energy consumption of the host machine at that given time. The time when the containers were turned on and the stress test started and ended have been marked in the Figures.

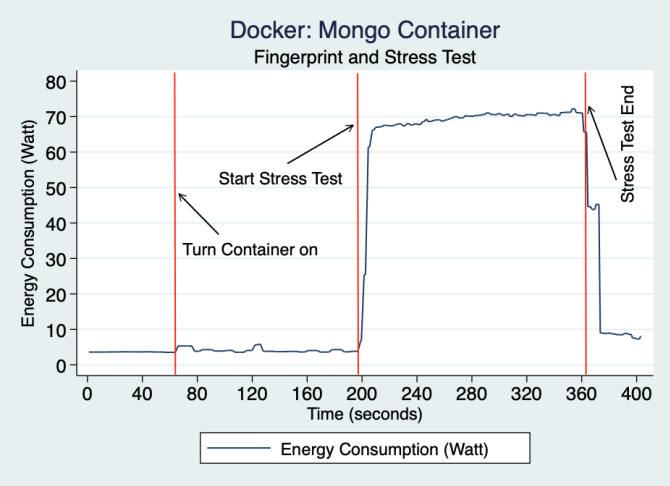

Fig. 10. Mongo enable and stress test

As seen from Figures 10 and 11, there is not much clear impact on energy for both the containers being turned on. The slight increase in energy consumption during that time could not be clearly associated with the start of the containers due to a wide range of internal processes by the host machine. Even though no energy overhead of starting Mongo DB and Postgres containers can be identified, these containers could be performing some other background processes.

A notable difference can be seen in the energy consumption during the stress test of the two containers. The peak energy

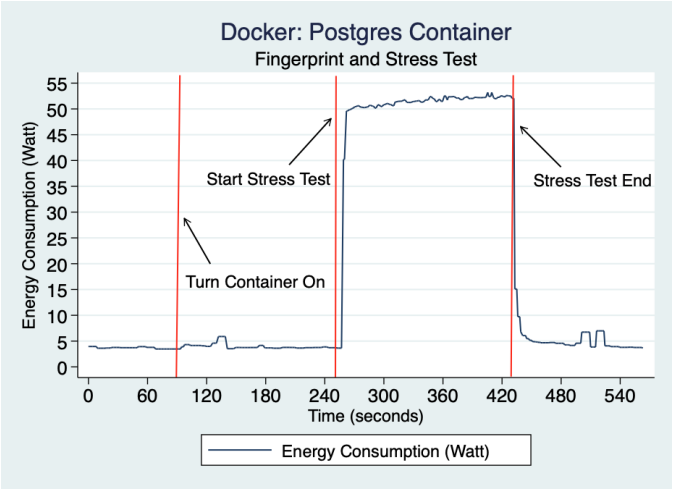

Fig. 11. Postgres DB enable and stress test

consumption of the Mongo DB is between 65 watts and 73 watts and for Postgres, it is between 50 watts and 53 watts. Mongo DB is consuming a lot more energy for the same amount of performance as compared to Postgres.

E. Fingerprint multiple containers

Starting a Docker container saw an increase in the energy consumption of the host computer (see Figure 5). In this Section, multiple different Docker containers were turned on and the energy consumption was measured to investigate the overheads of different Docker containers based on their functionality. The containers were turned off and a delay was added to ensure the containers do not affect each other's consumption data.

Figure 12 illustrates the energy consumption of different Docker containers being started one after the other. The type of Docker container is provided on the top of each corresponding spike in energy consumption. The X-axis denotes the time in seconds and the Y-axis represents the instantaneous energy consumption of the host machine at that particular time.

As seen from Figure 12, differences in the peak energy consumption can be clearly seen between a few groups of Docker containers. Nginx container and Python container have similar energy overhead starting at around 16 watts. The Hello World Docker container was found to be consuming the most amount of energy which was around 18 watts. PHP and Ubuntu containers consumed around the same amount of energy between 17 watts and 18 watts.

The Debian and Alpine containers consumed less energy to start as compared to all other containers. The Debian container used around 15 watts of energy and the Alpine container used around 10 watts (the lowest among the group). This fingerprinting of different Docker containers can have multiple security implications as it shows that the energy consumption of just starting the containers can be used to uniquely identify the type of container being started.

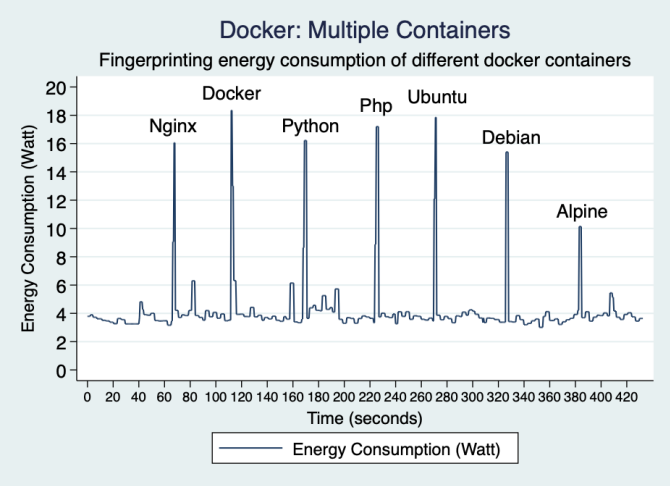

Fig. 12. Energy Consumption of different Docker container starting

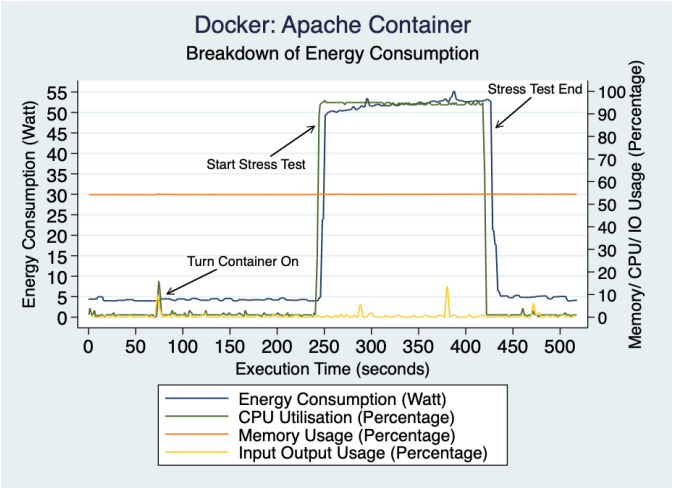

Fig. 13. Breakdown of Energy Consumption of Apache Docker container under stress

V. Breakdown of Energy Usage

To further investigate the differences between the energy consumption of different Docker technologies and the factors that might be affecting the energy consumption, a series of experiments were performed with a focus on getting the internal statistics of the computer and finding the root cause of the increase in energy consumption. The three main factors that are being considered are the CPU, Memory, and Input-Output calls (IO Utilisation). This will help co-relate the CPU, Memory, and IO utilization with the energy consumption of the host computer.

Figures 13 and 14 illustrate the energy consumption of web (Apache) and database-based (Mongo DB) Docker containers being started and stress tested. The graphs also include system information such as CPU, Memory, and IO Utilisation. The X-axis denotes the time in seconds. The left Y-axis denotes the instantaneous energy consumption in watts of the host computer at any given instant and the right Y-axis presents the following three attributes as a percentage - the CPU, Memory, and IO utilization. The point when the containers are started and the stress test is conducted is marked on the graphs.

Figure 13 illustrates the energy consumption of the Apache container as compared to the CPU, Memory, and IO utilization of the host computer. In contrast to Figure 5, a spike in energy consumption of the Apache container is not seen when the container is started. An increase in CPU and IO utilization can be seen at the same time. The memory utilization remains constant during the whole execution of the stress test and container starting up.

During the stress test, a corresponding increase in CPU utilization can be seen with less to no comparable increase in IO utilization. The energy consumption increases as the CPU utilization increases and it ranges between 50 to 55 watts. As there is no change in the memory or IO utilization, the major factor for the energy consumption of the computer can be attributed to its CPU utilization.

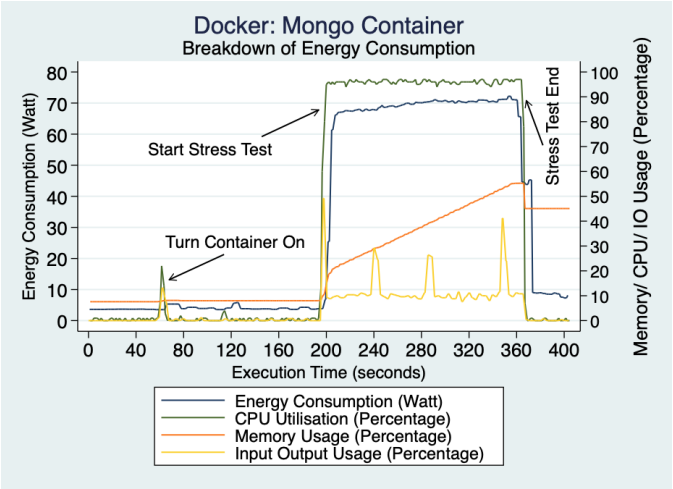

Fig. 14. Breakdown of Energy Consumption of Mongo DB Docker container under stress

Figure 14 illustrates the energy consumption of the Mongo DB container as compared to the CPU, Memory, and IO utilization of the host computer. Mongo DB is a database technology and its major functionality includes storing, reading, and updating data into memory and disk. The stress test being conducted tries to insert 100,000 records into the database which in turn gets stored on the disk. As expected, the CPU utilization is near 100% with all the processing of memory and reading and writing to the disk. As CPU utilization increases, a corresponding spike in energy consumption is also seen.

Database technologies are IO (input-output) intensive technology. As expected, the memory utilization gradually increases as we request the container to insert more records into the disk. These records are first stored in the memory and then a batch of these records is written to the disk altogether. Due to this, constant spikes in the IO calls to the disk are seen at regular intervals.

Based on the data from Figures 13 and 14, the energy

consumption of the host computer is majorly dependent on the CPU usage of the computation or workload. The memory and the IO utilization on their own marginally increase the energy consumption. A comparable increase in energy can be seen when all three attributes - CPU, Memory, and IO are being utilized. The maximum energy consumption range of the Apache container was around 50 - 55 watts and that of the Mongo DB container was around 68 - 73 watts. The CPU utilization is almost near 100% in both cases but in the latter case, we have a huge utilization of Memory and the IO calls to the disk. There are several factors that can affect the energy consumption of computers. The type of processor, the amount of RAM installed, the type of cooling system used, the type of Operating System, etc. can all have an impact on the amount of energy a computer consumes. The experiments presented in this paper are a step towards transparency on factors that affect the energy consumption of computers and Docker.

VI. Conclusion

Docker containers are becoming increasingly popular, as they offer a convenient way to package and deploy applications. However, there is a downside to this convenience: Docker containers can consume a lot of energy. Docker applications are being increasingly adopted for deploying all sorts of applications. Therefore, it is important to consider the implications of these applications and their parameters on energy consumption. Monitoring the energy consumption of Docker containers can help to save costs and reduce the carbon footprint of a computation. It can also help in identifying applications that are unnecessarily energy-intensive and necessary steps to improve their efficiency can be discussed.

The experimental results presented in this paper have a number of lessons, i) It is possible to measure the energy consumption of Docker containers. ii) It is possible to measure the level of workload by observing the energy consumption of containers. iii) Using this knowledge you can select from different container implementation technologies. iv) It is possible to analyze the underlying factors that impact the energy consumption of the machine.

This paper has demonstrated a strong need for approaches to optimize container workload executions with a focus on energy consumption and also performance. In future work, we hope to further investigate container-based workloads, including distributed workloads using Docker Swarm and Kubernetes. It is important to note that, all the experiments presented in this paper were conducted on a single machine, which makes extrapolating the results to other platforms difficult because different hardware setups can have varying power utilization. This will be further explored in the future by expanding the computing resource to include more diverse hardware. With this knowledge, we aim to develop approaches for optimizing energy-efficient decision-making in container orchestrations.

References

[1] I. Docker, "Docker," *línea].[Junio de 2017]. Disponible en: https://www. docker. com/what-docker*, 2020.

[2] J. Nickoloff and S. Kuenzli, *Docker in action*. Simon and Schuster, 2019.

[3] E. Carter, Nov 2022. [Online]. Available: https://sysdig.com/blog/2018-docker-usage-report/

[4] E. Casalicchio and V. Perciballi, "Measuring docker performance: What a mess!!!" in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, 2017, pp. 11–16.

[5] A. Lingayat, R. R. Badre, and A. K. Gupta, "Performance evaluation for deploying docker containers on baremetal and virtual machine," in *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2018, pp. 1019–1023.

[6] S. C. Peter, "Reduction of co2 to chemicals and fuels: a solution to global warming and energy crisis," *ACS Energy Letters*, vol. 3, no. 7, pp. 1557–1561, 2018.

[7] D. Gielen, R. Gorini, R. Leme, G. Prakash, N. Wagner, L. Janeiro, S. Collins, M. Kadir, E. Asmelash, R. Ferroukhi *et al.*, "World energy transitions outlook: 1.5° c pathway," 2021.

[8] L. Delannoy, P.-Y. Longaretti, D. J. Murphy, and E. Prados, "Peak oil and the low-carbon energy transition: A net-energy perspective," *Applied Energy*, vol. 304, p. 117843, 2021.

[9] M. Warade, J.-G. Schneider, and K. Lee, "Measuring the energy and performance of scientific workflows on low-power clusters," *Electronics*, vol. 11, no. 11, p. 1801, 2022.

[10] I. Miell and A. Sayers, *Docker in practice*. Simon and Schuster, 2019.

[11] M. U. Haque, L. H. Iwaya, and M. A. Babar, "Challenges in docker development: A large-scale study using stack overflow," in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2020, pp. 1–11.

[12] E. A. Santos, C. McLean, C. Solinas, and A. Hindle, "How does docker affect energy consumption? Evaluating workloads in and out of Docker containers," *Journal of Systems and Software*, vol. 146, pp. 14–25, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218301456

[13] M. V. Warade, J.-G. Schneider, and K. Lee, "Fepac: A framework for evaluating parallel algorithms on cluster architectures," in *2021 Australasian Computer Science Week Multiconference*, 2021, pp. 1–10.

[14] J. Shah and D. Dubaria, "Building modern clouds: using docker, kubernetes & google cloud platform," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0184–0189.

[15] N. Marathe, A. Gandhi, and J. M. Shah, "Docker swarm and kubernetes in cloud computing environment," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 2019, pp. 179–184.

[16] C. Anderson, "Docker [software engineering]," *Ieee Software*, vol. 32, no. 3, pp. 102–c3, 2015.

[17] S. Afreen, "What is a dockerfile: Everything you need to know," https://www.simplilearn.com/tutorials/docker-tutorial/what-is-dockerfile, 2022, accessed: 2022-12-10.

[18] S. S. Tadesse, F. Malandrino, and C.-F. Chiasserini, "Energy consumption measurements in docker," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, 2017, pp. 272–273.

[19] J. Luo, L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, and H. Luo, "Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot," *Future Generation Computer Systems*, vol. 97, pp. 50–60, 2019.

[20] D.-K. Kang, G.-B. Choi, S.-H. Kim, I.-S. Hwang, and C.-H. Youn, "Workload-aware resource management for energy efficient heterogeneous docker containers," in *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 2428–2431.

[21] Z. Li, H. Wei, C. Lian, and S. Qin, "Docker-based energy management system development and deployment methods," in *2020 4th International Conference on Power and Energy Engineering (ICPEE)*, 2020, pp. 1–5.

[22] M. Xu and R. Buyya, "Brownoutcon: A software system based on brownout and containers for energy-efficient cloud computing," *Journal of Systems and Software*, vol. 155, pp. 91–103, 2019.

[23] M. Sureshkumar and P. Rajesh, "Optimizing the docker container usage based on load scheduling," in *2017 2nd International Conference on Computing and Communications Technologies (ICCCT)*, 2017, pp. 165–168.

[24] M. Chae, X. THANGONGSAK, Y. GUANG, and H. Lee, "Energy efficient web load balancer using docker," in *Proceedings of the Ko-*

rea Information Processing Society Conference. Korea Information Processing Society, 2018, pp. 43–45.
[25] N. VasanthaKumari and R. Arulmurugan, "Reorganizing virtual machines as docker containers for efficient data centres," in *3rd EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing*. Springer, 2022, pp. 201–211.
[26] I. M. Murwantara and P. Yugopuspito, "Evaluating energy consumption in a different virtualization within a cloud system," in *2018 4th International Conference on Science and Technology (ICST)*, 2018, pp. 1–6.
[27] Ö. E. Demirkol and A. DEMİRKOL, "Energy efficiency with an application container," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 26, no. 2, pp. 1129–1139, 2018.
[28] S. Kreten, A. Guldner, and S. Naumann, "An analysis of the energy consumption behavior of scaled, containerized web apps," *Sustainability*, vol. 10, no. 8, p. 2710, 2018.
[29] M. Schwartz, "Internet of things with esp8266," in *Internet of Things with ESP8266*. Oxford: Packt Publishing Ltd, 2016, ch. 13, pp. 190–200.
[30] J. Cook, "Docker hub," in *Docker for Data Science*. Springer, 2017, pp. 103–118.
[31] A. Sergeev, E. Rezedinova, and A. Khakhina, "Stress testing of docker containers running on a windows operating system," in *Journal of Physics: Conference Series*, vol. 2339, no. 1. IOP Publishing, 2022, p. 012010.
[32] L. Benedicic, F. A. Cruz, A. Madonna, and K. Mariotti, "Sarus: Highly scalable docker containers for hpc systems," in *International Conference on High Performance Computing*. Springer, 2019, pp. 46–60.
[33] Datadog, "9 insights on real world container use," https://www.datadoghq.com/container-report/, 2022, accessed: 2022-12-10.
[34] G. Tene, "wrk2: a http benchmarking tool based mostly on wrk," 2018.
[35] W. Reese, "Nginx: the high-performance web server and reverse proxy," *Linux Journal*, vol. 2008, no. 173, p. 2, 2008.
[36] O. H. Jader, S. Zeebaree, and R. R. Zebari, "A state of art survey for web server performance measurement and load balancing mechanisms," *International Journal of Scientific & Technology Research*, vol. 8, no. 12, pp. 535–543, 2019.
[37] "Nginx vs apache: Head to head comparison," https://hackr.io/blog/nginx-vs-apache, accessed: 2022-12-10.
[38] johnlpage, "johnlpage/pocdriver: Workload driver for mongodb in java," Aug 2021. [Online]. Available: https://github.com/johnlpage/POCDriver
[39] "Postgresql manual," 2013. [Online]. Available: https://www.postgresql.org/docs/devel/pgbench.html