Mittuniversitetet
MID SWEDEN UNIVERSITY

Jails vs Docker

A performance comparison of different container technologies

Christian Ryding & Rickard Johansson

Computer Engineering, Final Project
Main field of study: Computer Engineering
Credits: 15 hp
Semester/Year: 2019/2020
Supervisor: Awais Ahmad
Examiner: Felix Dobslaw
Course code/registration number: DT133G
Degree programme: Degree of Bachelor of Science with a major in Computer Engineering

Jails vs Docker

- *A performance comparison of different container technologies*

Christian Ryding, Rickard Johansson
BSc. Thesis, Programvaruteknik
Institute of Computer and Systems Sciences
Mid Sweden University
Östersund, Sweden
chry1200@student.miun.se
rijo1001@student.miun.se

***Abstract* - Virtualization is used extensively by Enterprise IT architecture and cloud computing, it is used to provide customers a part of their hardware resources as a service. Container technology is the new generation of virtualization and provides performance benefits due to less overhead. Earlier research has compared different container technologies regarding their performance, including Docker which is the most popular container technology. Most of this research has been focusing on Linux based container technologies. Even though there is interest in knowing how other container technologies under different operating systems perform. In this study we explore the performance of Docker in contrast to the performance of a contending container technology named Jails. We present how well each container technology performs running one or multiple containers, in the areas of CPU, memory, read from disk, write to disk, network and startup time efficiency. The comparison was done using collected statistics from different benchmarking tools. Results from this study have shown that Docker is utilizing shared resources and has better stability compared to Jails. We also discuss what unexplored benefits Docker and Jails can have by implementing each other's unique features. Future work could consist of writing to disk or reading from disk performance tests under one common filesystem, e.g., ZFS file system.**

Index Terms* - *Docker, Jails, Container, Linux, FreeBSD

1. Introduction

Virtualization is commonly used in both Enterprise IT architecture and cloud computing, to provide a part of their hardware resources to a customer as a service [1]. The new generation of virtualization is referred to as containerization [2]. The increased interest in this container technology [3], along with its rapidly growing adoption rate [4], is due to the potential performance benefits, especially at large scale deployments in the cloud. Being able to run an application with less overhead as container-based virtualization can be an alternative to conventional virtualization, potentially reduces resource overhead and thus improves the utilization of data centers [3]. Google and Twitter are companies that currently utilize containers in some of the worlds largest server farms [5].

Linux is the most widespread UNIX-like operating system. FreeBSD which also is a UNIX-like operating system [6], is only used in some specific environments. Even though proved to be more stable and in some scenarios even more suitable than Linux [4]. A previous study that compared different container technologies, Jails and LXC. Indicated that there is importance to explore other operating systems and other container technologies. This can give an answer to

which containerization alternative is superior, or when specific containerization solutions should be used [4]. Another study which researched the effectiveness of different container technologies, Docker and Flockport, implies the topic is an interesting area for further study. Especially when looking at the effectiveness of multiple containers running with the same shared resources. Researching the utilization of shared resources is interesting because this is a common case within cloud services [7].

Most of the research covered in this area has been focusing on container technologies that run on Linux, for example Docker [4]. Even though FreeBSD are a very popular choice among cloud providers [8]. And FreeBSD:s container technology Jails, is considered a good candidate in container technology selection [4]. As per authors' best knowledge no research which compares Dockers performance up against Jails have been done. Hence the aim for this study is to compare and evaluate the performance of the two different container technologies Docker and Jails, while running one or multiple instances at the same architecture.

2. Problem Statement

Recently there is an increasing interest in container technology [3], and usage of this lightweight virtualization technique referred to as containerization is rapidly growing [4]. This is due to the potential performance benefits, especially at large scale deployments in the cloud. Being able to run an application with less overhead as container-based virtualization can be an alternative to conventional virtualization, to potentially reduce resource overhead and thus improve utilization of resources in data centers [3]. This is why the efficient utilization of resources is an important area of research. Good utilization of a host system's resources is a frequent scenario for cloud providers[7].

It is therefore important to research different container technologies and container technologies on other operating systems [4]. To be able to choose the right one based on its specific need, e.g. network performance or utilization of shared resources. Most of the research done in this area has mostly focused on container technologies like Docker and LXC which are Linux based [4]. That's why this paper is going to study the difference between Docker and FreeBSD:s proven container technology Jails which runs on a different operating system, is less researched and could be a potential contender in choice of container technology. Jails which is said to be more secure than Linux based containers [9], runs on the operating system FreeBSD, which is proven to be able to have a higher performance [10] and provide higher stability compared to Linux [4].

Hence the purpose of this paper is to study the difference between FreeBSD Jails and Docker on Linux, how well each of these perform running one or multiple instances. Jails and Docker differ somewhat in terms of feature set, which is why we also aim to explore what benefits could be had for each of these technologies by implementing each of their respective key features.

3. Research Questions

In this study we aim to measure Docker containers and Jails efficiency; with efficiency, we mean the individual measurable performance of the CPU, memory, write to disk, read from disk, network latency, startup times and stability. Stability will be measured in the form of the variance in results. This leads to the following research questions:

RQ.1. How efficient is Docker compared to Jails?
- When it comes to running one instance.
- When it comes to running multiple instances

RQ.2: What unexplored benefits Docker and Jails can have by implementing each other's unique features?

4. Limitations

This study is limited to investigate the performance of the two container technologies Docker and Jails, limitation consists of time restriction, which in this case is ten weeks dedicated to the study. We have also limited the study to test up to 32 simultaneously running containers. This is done to cover the most frequent use case, which is 30 containers per host [11], while not pushing our hardware too hard. The filesystem used in this study is also a limitation, we opted not to go for a common filesystem because FreeBSD:s ZFS filesystem is experimental under Linux and is advised against for production use [12].

5. Background

5.1 Virtualization

Virtualisation makes use of software a hypervisor to create an abstraction layer over the computer hardware that allows the hardware to be divided between multiple virtual computers as known as a virtual machine (VM) [1][13].

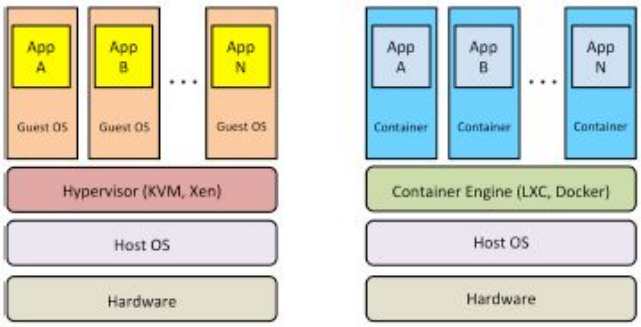

Figure 1. Hypervisor-based vs Container-based virtualization [14].

The use of virtualization enables efficient utilization of the underlying computer hardware, this is particularly important in the cloud computing industry where companies are using virtualization to be able to provide a part of their hardware resources to a customer or service[1]. Hypervisor based virtualization also has the benefits of being able to run multiple different operating systems with different kernels at the same time [2]. Virtualization enables cloud providers to serve users with the ability to adapt quickly by providing only the computer resources needed cost-effectively [1]. Virtualization does have an overhead on the performance that becomes a pervasive performance tax added on to the workload, which is the price of the flexibility gained [15].

5.2 Containerization

What has revolutionized many industries and has had a major impact on modern computing is container technology [3]. Container technology or containerization is another name for operating system level virtualization and became very popular with the release of Docker [16]. This is partly because this technology requires so little resources, is portable, has low energy consumption, is disk storage efficient, cost effective, performance efficient and can be started up fast [3]. Container technology thus enables rapid deployment, patching or scaling of different applications in a good way [17]. A container is a lightweight operating system that runs on the host operating system or under the host operating system in a VM. The container is limited to running on the same kernel as the host operating system in order to access its hardware. Even though the concept is not new, the technology is based on “chroot” (change root) which can isolate single processes from each other, without having to emulate some sort of hardware for each of them. Hence providing more computing power [2].

5.2.1 Chroot

Chroot Is a UNIX system operation that changes the root directory of the current running process and its children to a single subtree. A program in such an environment can typically not access or modify files outside of the designated directory tree but compartmentalization does not extend to processing or networking spaces [18].

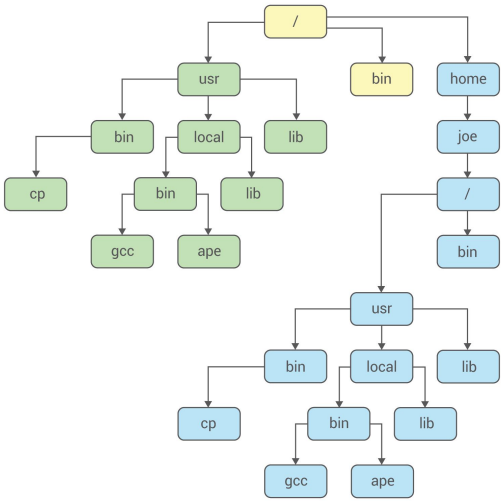

Figure 2. Chroot illustration [19].

5.2.2 Application and Machine container

There are two types of containers, application containers and machine containers [3]. Machine containers consist of container technologies like Jails and LXD [9], it runs an operating system small in size and can contain a complete filesystem [3]. A machine container can run multiple processes at the same time, and it's also possible to install different libraries, languages and databases [20]. Application containers run a single process, application or service, container technologies like Docker belong in this category [3]. Docker is optimized for the deployment of applications, as opposed to the mentioned machine containers, which is reflected in the design philosophy and interfaces [5], while container technologies like Jails and LXD are optimized to be machine containers [9]. Both of these types of containers run close to native performance, with negligible latencies [3].

5.3 Docker

Docker is platform independent container technology and can run under Windows Server 2016, Windows 10, MacOS but mainly Linux and multiple other cloud services [5]. Docker is one of the most popular container technologies [21], and is an extension of the container technology Linux Containers (LXC) that is native to the Linux kernel [2].

Docker is still growing in popularity, and according to Datadog's usage statistics Docker has seen large yearly growth such as in 2018 where deployments increased by 75% among Datadog users [22]. Much of Dockers growth is driven by Docker Hub, which is an Open Source platform [2]. Docker Hub is the official container image repository additionally makes it very easy for Docker users to extend or use existing images from other people or organizations such as Oracle, Microsoft, Canonical. Docker Hub is essentially a repository that solves the distribution problem for Docker users. Docker Hub can most certainly contribute to the success of Docker by solving the distribution problem, as of writing the Docker Hub registry is the largest container registry and is only growing with 8 Billion downloads in January of 2020 alone [23].

One of the major features of Docker is the usage of layered cashing. A Docker image consists of read-only layers each of which represents a set of Dockerfile instructions. The layers are stacked and each one is a delta of the changes from the previous layer [24]. When a change is made to a Dockerfile and rebuilds the image, only those layers which have changed are rebuilt. This is part of what makes images lightweight, when compared to other virtualization technologies [25].

5.4 Jails

FreeBSD is a popular operating system, in large part due to the network stack performance, and licensing [26]. FreeBSD is also being thought of as an operating system with higher isolation compared to Linux [8][27][28]. Jails is a platform dependent container technology native to the BSD UNIX operating system such as OpenBSD, FreeBSD and NetBSD [9]. Jails improve on the concept of chroot in several ways. Processes are only limited in the part of the files they can access. Jails expand on chroot by virtualizing access to the file system, the set of users and the networking subsystem. More fine-grained controls are available for tuning the access of a jailed environment. Jails can be considered as a type of operating system-level virtualization [29]. When a Jail is created it sets up a virtual instance of an operating system in a separate directory dedicated to that specific Jail. Jails who is based on chroot is considered more isolated than container based technologies on Linux based on chroot [9]. Iocage which is a tool to help manage and configure Jails using plugins [30]. These plugins make it easier to install different applications and configure Jails, although only twenty-one existing plugins at this time [31].

6. Related Work

An earlier study that compared the performance between the container technologies Docker and Flockport pointed out that there is interest in knowing how multiple running containers will utilize shared resources. Reasoning behind this is that microservices are using this kind of architecture and containers could be utilized to run every microservice in a separate container [7]. In another study that compared and analysed different implementations of DNS64 running on different operating systems (FreeBSD, OpenBSD & Linux). The study measured execution time, standard deviation (STDEV) and maximum time. It would also look at the CPU utilization and its STDEV. Measurements such as memory utilization and number of requests served per second by web application were also considered. Results indicated that the DNS64 implementation could be used for stability and performance under both Linux and FreeBSD, one implementation running on the operating system OpenBSD was considered unstable. The study showed that the different implementations performed overall better under Linux compared to the BDS:s, although there were situations where FreeBSD performed better than Linux, e.g. one performance (TOTD) result showed that FreeBSD could handle 1034 requests per second while Linux could handle 1010 requests per second [10].

Previous research comparing FreeBSD's container technology Jails and Linux container technology LXC looked at the efficiency of different workloads, using containers in clusters. The study shows that FreeBSD gives more stable results compared to Linux and is considered a good candidate in the choice of container technology. The study also means that it is important to explore other operating systems and other container technologies, for comparison and optimization of future design choices [4].

In earlier research the filesystem ZFS have shown to be slower compared to EXT4 with parallel I/O. With more threads accessing the filesystem on ZFS and larger file size, performance decreases much more than EXT4 [32]. ZFS was faster when performing writes of small file sizes and a low thread count, however, once the number of threads became higher or the size increased performance decreased [33]. Despite interest in comparing different container technologies and their performance, no academic publication to our knowledge has been published which compared Dockers performance versus Jails. Therefore this paper will be the first on that subject. This will help others

in choosing appropriate container technology with performance, optimization, resource utilization in mind.

7. Research Methodology

We have evaluated the performance of Docker and Jails by conducting different performance tests. Tests have covered CPU, memory, writing to disk, reading from disk, network and startup times performance. Tests for CPU and memory will be done with the help of Sysbench, for writing to disk and reading from disk performance, a program called IO-nugget which was created to help benchmark disk I/O. For benchmarking network performance, we utilized Iperf. At lastly, we made use of the program Date to measure the performance for Docker and Jails startup times.

7.1 Selection of tools

Tools chosen to be used in this study must fulfill the requirements of providing cross-platform compatibility between Linux and BSD, while also adequately being able to provide useful statistics for the areas of measurement outlined in the stated research questions and used in previous similar research. For our own construction part we used cross-platform tools like Python and Date. Bash scripts were also used in combination with benchmarking tools to collect statistics and handle the running of performance tests.

7.1.1 Sysbench

Sysbench, according to it's manual, allows for testing, Scheduling performance (CPU), I/O, memory allocation and transfer speeds, POSIX threads performance and database server performance [34]. Sysbench fills the requirement stated in 7.1.

7.1.2 Iperf

Iperf [35] is a cross-platform tool whose purpose is to measure the maximum achievable bandwidth on IP networks. For this study we choose to use Iperf2 in front of Iperf3 due to being able to run with multiple clients for the server. It also satisfies the basic requirements of section 7.1.

7.1.3 Python3

Python is an interpreted programming language that is highly supported on both BSD and Linux [36].

7.1.4 Date

Date is a program included in GNU coreutils, it is used to print or set system date and time [37].

7.2 Data collection

Jails and Docker will run on a host with equal hardware:
Processor: AMD FX-8320 Eight-Core Processor 3.50 GHz.
Graphics card: NVIDIA GeForce GTX 660 2048MB.
Ram: 32GB CL10 1600Mhz.
Hard drive: 128GB SSD.

Performance testing will be run on FreeBSD 12.1 and Ubuntu 19.10, which were the latest stable releases for each operating system at the time of testing [38][39]. Both operating systems have a base installation with the same desktop environment. GUI on both systems were disabled when tests were running, in order to free resources as much as possible. Data collection was performed with platform independent benchmarking tools, contained in both Jails and Docker containers. All of the benchmarks were only performed with the tool described in section [7.1.1] to [7.1.4].

To be able to answer our research questions, six experiments were performed. One CPU performance test, one performance test for memory, one write to disk performance test, one read from disk performance test, one performance test for network and one performance test for startup time. To improve reliability of the results each test was conducted 10 times for each container setup. This study contains 6 different container setups (1, 2, 4, 8, 16 and 32 containers). This is done for both Jails and Docker, so for one experiment 120 tests are completed. This is equal to a total of 720 tests performed.

7.2.1 CPU

Sysbench will stress and test the performance of the CPU for 10 seconds by verifying prime numbers up to 10 000. After each verification it will go on to the next verification [40]. Every instance of sysbench will use 8 threads. Performance for cpu will be measured in events per second, this is one of the most important values for comparison [40].

7.2.2 Memory

Sysbench will continuously write a set amount to an allocated memory buffer, until a set limit is reached [40]. In this case 1 KiB will be written repeatedly for 10 seconds, and 8 threads will be used for every sysbench instance.

7.2.3 Write to disk and read from disk

IO-nugget is a cross platform tool constructed for this study to measure disk I/O. It is written in Python 3.7 to be easily built and runnable on both FreeBSD and Linux. The script writes a temporary file while measuring write speed for each block or random data written. Reading is measured for each read block.

7.2.4 Network

Iperf was used to benchmark network performance under TCP. Container clients will connect to one container server, both running iperf. Container's network performance will be measured in its bandwidth. Default Iperf settings were used, which means a TCP window size of 64 KByte.

7.2.5 Startup times

Date was used for measuring the time it took to start Docker containers and Jails. Date was used to print out current time before starting the Docker containers and Jails. It was used to calculate the difference in timestamps for the started Jails or Docker containers. Startup time is the difference between the time logged before starting the Docker containers and Jails, and their slowest registered time.

7.3 Data analysis

In this study CPU performance was defined by how many events it can perform each second. How many Mebibyte that is written each second defined memory performance.
Write to disk as well as read from disk performance were defined by how many megabytes per second could be written or read from disk. Network performance was defined by how many gigabytes could be sent per second.
The difference between Date's unix timestamps of starting an existing container and stopping the container is how starting times are measured. Stability was defined by each test results STDEV. For all of the tests above, an analysis was done. In the analysis results were summarized with the median, which is the midpoint of a set of values [41]. This was done to be able to find different patterns in the results. The results have been examined to determine how it compares to the theory presented, and discussed in their similarities or differences, as well as possible explanations.

7.4 Validity threats

To increase the internal validity we aimed to make use of well tested programs like Sysbench, and Iperf which have been used in previous research and therefore have proven to give reliable results and as well as supporting multiple platforms. For our own created tests we did use cross platform tools and language like python and date. No major modification or interpretation differences exist for the benchmarking applications used. In this study however only one test rig was used, one or more test rigs could have been used in order to further increase validity. Another improvement container technology contributes to less carbon emission in the environment compared to other soet lutions lhus container technology contributes to less carbon emission in the environment compared to other soet lutions lent could also be to use more benchmarking tests of the same kind e.g multiple Network test, I/O test applications. Differences might come from the preferred file systems between FreeBSD (ZFS) and Linux (EXT4). Both file systems do have some inherent performance impact mainly on the I/O benchmarks in this study. External validity, there are no major variables that come into play that affect the result of this study outside of testing hardware and software that was used. The results of this study would be easy to reproduce given that reasonably similar hardware is used together with the tools and benchmarking scripts used in this study.

8. Results

In this section, we present the results of performance tests for CPU, memory, write to disk, read from disk, network, startup times, and stability. The experimental results are presented for one, two, four, eight, sixteen, and thirty-two running containers. These results will be used as the base for the discussion and answering the research question "How efficient is Docker compared to Jails?", when running one or multiple containers.

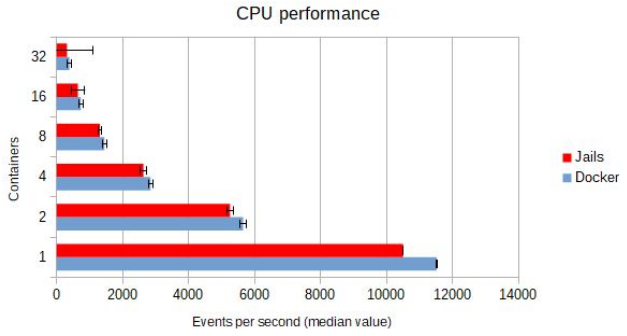

Figure 3. Evaluation of CPU performance. Sysbench performance (events/sec) of prime numbers (up to 10 000). Each data point is the median of ten test runs. Figure also shows the STDEV for each data point.

The graph above shows the median results of ten CPU measurements of events per second for a range of

containers, one to thirty two containers. By events per second in this case refers to previously defined performance measurement in section [7.2.1]. Results show that Docker performs better in every container setup regarding CPU performance. Graph above also shows that the STDEV for CPU performance is lower for Jails when running one or eight containers. From sixteen to thirty two containers it is a higher difference gap than previous.

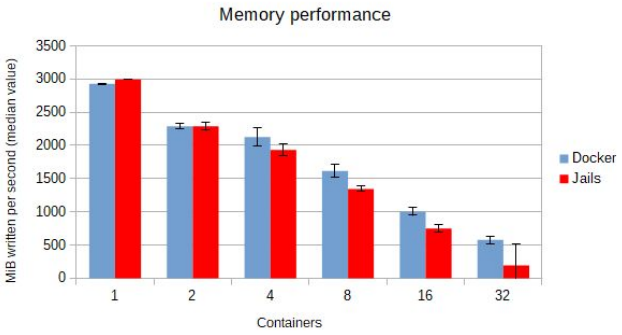

Figure 4. Evaluation of memory performance. Sysbench performance (write speed/sec). Each data point is the median of ten test runs. Figure also shows the STDEV for each data point.

The graph above shows the median results of ten memory measurements of writes speed per second for a range of containers, one to thirty-two containers. By write speed per second in this case refers to previously defined performance measurement in section [7.2.2]. Results show that Jails have a higher write speed when running one or two containers while Docker is having a higher write speed per second running four or more containers. Docker is showing a descending curve when running increasingly more containers, and has a very high STDEV when running thirty-two containers, even though it performs significantly inferior compared to Docker.

The following graph shows the results collected from the I/O performance tests, writing to memory. They all display the number of containers used from one to thirty two as well as the key performance number collected megabyte per second. For both read and write there are median and STDEV for megabyte per second to number of containers.

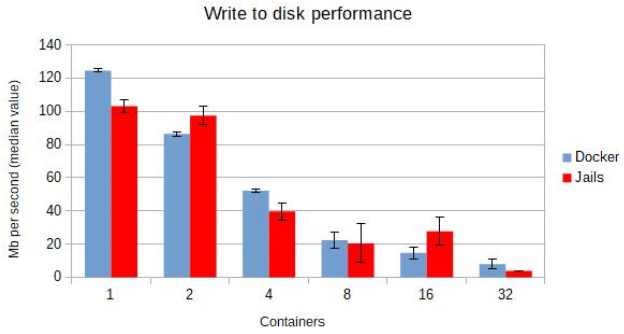

Figure 5. Evaluation of write to disk performance with help of IO-nugget (write speed/sec). Each data point is the median of ten test runs. Figure also shows the STDEV for each data point.

The graph above presents the median results collected for writing to disk in 10 runs. The results appear to be competitive, however at 1,4,8 and 32 containers Docker appears to be catching up in resource system utilization. The previously mentioned trend seems to continue where Docker indicates better performance at increasing container count. The STDEV is showing that the variance in results are significantly higher compared to Docker for the most part.

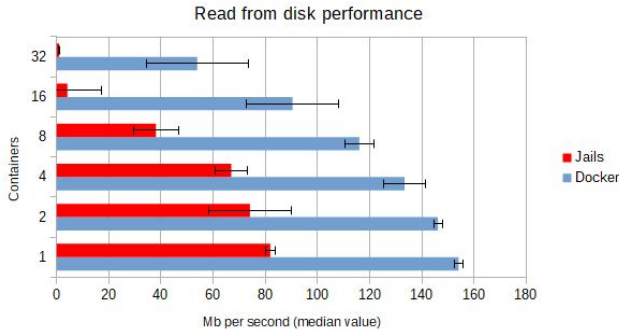

Figure 6. Evaluation of reading from disk performance with help of IO-nugget (read speed/sec). Each data point is the median of ten test runs. Figure also shows the STDEV for each data point.

The graph above shows the median of collected read statistics from the benchmarking tool IO-nugget described at [7.2.3]. The statistics show that Docker appears to exceed Jails performance at all container counts from one to thirty two containers. Thus the conclusion can be made that docker reads at a consistently higher rate. Notable in this test is that Jails is exceedingly far away in terms of performance and growing with container count. Graph shows higher variance for Docker at container count four, sixteen and thirty two, all other Jails have higher variance.

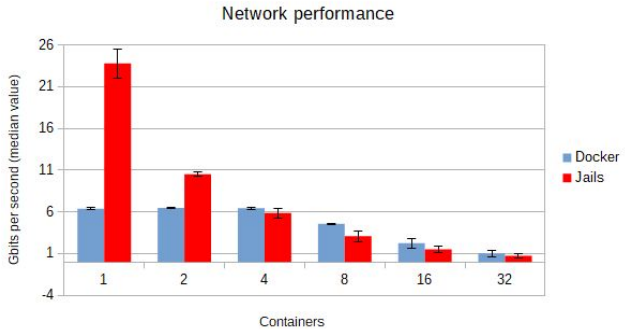

Figure 7. Evaluation of network performance with help of Iperf (throughput speed/sec). Each data point is the median of ten test runs. Figure also shows the STDEV for each data point.

The graph above shows the median results of ten network measurements, and how high throughput that was sent. Network throughput in this case refers to previously defined performance measurement in section [7.2.4]. Results show that Jails have higher throughput when running one or two containers. Docker is having a higher network throughput when running four or more containers. The STDEV for network performance is showing that Jails have higher variance in results when running eight or less containers. Docker shows a higher variance in results when running sixteen or more containers.

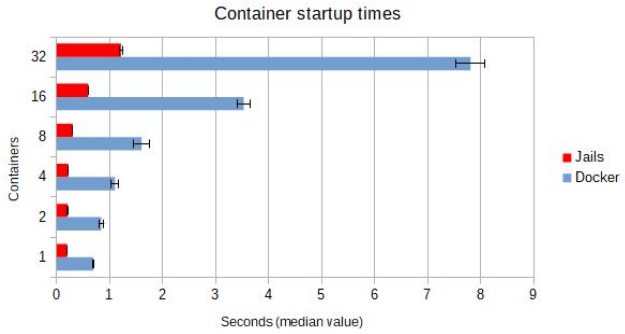

Figure 8. Evaluation of container startup time performance with help of Date script (startup time in seconds). Each data point is the median of ten test runs. Figure also shows the STDEV for each data point.

The graph above shows the median results of ten startup times measurements of how long it took to startup one to thirty two containers. By startup time in this case refers to previously defined performance measurement in section [7.2.5]. Results show that Jails are continuously faster compared to Docker. The STDEV for startup times is showing that Jails have less variance in results for every container setup, one to thirtytwo.

9. Discussion

The performed tests showed that Docker does have a higher overall CPU performance consistently across all tests while also showing less variance. The performance trend can be observed running one container but as well as when running multiple containers. This is consistent with earlier research [10], which showed that different implementations overall performed better under Linux compared to FreeBSD. In our tests Jails had a negligible higher memory performance running one container. Running increasingly more containers showed a trend of a higher descending curve, this indicates that Docker utilizes shared resources in a better way.

Regarding the I/O performance tests for Docker and Jails, Docker appears to be outperforming Jails significantly in read performance (Mb/s) across the measurements. And write performance initially was in favour of Docker but between two and sixteen remains competitive. Hower at 32 containers with many threads writing, Jails appears to fall off more in performance than Docker. Looking at the write performance benchmark is more interesting which seems to indicate a clear trend that Docker performs better overall but with a small edge over Jails except for with 2 and 16 containers at the 32 however the difference seems to increase. Expected Jails to be consistently faster in terms of writes but it appears to show a narrowing gap as the number of containers increases. ZFS has shown to struggle compared to EXT4 with parallel I/O when more threads are in use. ZFS on FreeBSD was actually faster than EXT4 was when performing writes at a lower thread count, however once the number of threads increased accessing the filesystem with container count performance decreased [33]. Could one explanation for this be due to the innate differences between the underlying filesystem EXT4 that was running on the Linux machine while the FreeBSD by default runs ZFS which is an advanced filesystem with many specialized features but is generally slower [32][33].

Jails network performance running one container shows a significantly higher performance compared to Docker in our tests, this confirms the theory that FreeBSD has a good network stack [26]. The results also show that Jails have a more descending performance curve compared to Docker when running multiple containers. This once again indicates that Docker is better at utilizing shared resources compared to Jails. Results for startup times showed that Jails were significantly faster than Docker, both when running one or

multiple containers. This was one of a few times Jails showed a better utilization of shared resources compared to Docker. An explanation to this could be Jails stable foundation, that Jails was introduced in 2000 with the release of FreeBSD 4.0 [38]. That is thirteen years before Docker was released [16]. Another possible explanation could be that Jails perhaps have less overhead, and therefore leads to shorter startup times.

Even though earlier research found Jails on FreeBSD to be more stable than the container technology LXC on Linux [4]. Results in this study proves the opposite, with the exception of startup times. A possible explanation to this could be Jails stable foundation, and that it was released 13 years before Docker [16][38]. Another possible explanation of the added startup time could be due to some of Docker additional features that adds some overhead to startup such as layered cashing. The most significant proof of Dockers stability is proven in our results when thirty-two containers are running. Two examples of this are for the variance in results for CPU and memory, where Jails are putting up very high variance in results compared to Docker. There is also high variance in results when running a small amount of containers. An example of this is in network performance, where Jails show a significant higher variance compared to Docker. A higher variance in results could be explained by its significantly higher performance, especially when running one container. In some cases Jails had a very low variance in results, for example read from disk when running thirty-two containers. A possible explanation to this could be because the performance was so low, so the variance in results could not be so high.

To answer our second research question, “What unexplored benefits Docker and Jails can have by implementing each other’s unique features?”, we will discuss three different features. These features are platform independence, machine container versus application container and container plugins. If we start with the feature platform independence, we can see that Jails is a platform dependent container technology native to the BSD UNIX operating system such as OpenBSD, FreeBSD and NetBSD [9]. Docker on the other hand is a platform independent container technology and can run under Windows Server 2016, Windows 10, MacOS but mainly Linux and multiple other cloud services [5]. Docker is one of the most popular container technologies [21], and availability on multiple platforms could be a possible explanation to this. Because even if it runs natively on Linux [2], the possibility to test, develop under other operating systems or platforms should be a big advantage. Other benefits Jails could receive by making it cross-platform like Docker, is to get portability. A container can be utilized in multiple different environments despite differences in native system dependencies and therefore should be more attractive for adopters such as programmers, business owners and enterprises. By attractive we mean an application or service created and containerized once, which then can be deployed on various operating systems and platforms at moments notice.

For the second feature machine container versus application container, we can see that Docker could get benefits by implementing a machine container approach. Because then Docker would be able to run multiple processes at the same time. It would also make it possible to install libraries and dependencies at any time [20]. This could possibly save time whenever this is needed compared to having to create a whole new container image. In the results we see that Docker generally shows better performance when running multiple containers sharing the same resources. If Jails would implement an application container approach, it could eventually improve its performance regarding shared resources utilization.

The third feature, container management through plugins with Docker Hub versus the closest equivalent for Jails which is Iocage. Docker Hub is the official container image repository additionally makes it very easy for Docker users to extend or use existing images from other people or organizations such as Oracle, Microsoft, Canonical. Docker Hub is essentially a repository that solves the distribution problem for Docker users. Docker Hub has most certainly contributed to the success of Docker. Today the Docker hub registry is the largest container registry and is only growing with 8 Billion downloads in January of 2020 [23]. In addition to this private Docker repositories can also be created for sharing images internally in a company or organization. This greatly contributes to the ease of use compared to Jails where you have to set up everything yourself mostly. Iocage does exist for Jails but it does not have as advanced features such as GUI and search and does not ship by default with Jails compared to Docker Hub. Iocage plugins allow for a simple and very fast method to get containers installed and configured. At its core, a plugin is a Jails image specifically running one program that is a pre configured Jails [31]. This is in principle similar to the idea of Docker Hub but with a very limited selection of pre configured containers and does not do layered caching.

Container technology has a number of ethical implications for sustainability and security. There is a sustainability aspect to container technology, given that less energy is used to essentially get the same amount of computational resources out of a computer system. Thus container technology contributes to less carbon emission in the environment compared to other solutions like e.g. previously mentioned VM sandboxing. Personal data aspec; computer systems often contain a lot of personal data, so from a business perspective and a user perspective that sensitive data is kept securely in order to maintain confidentiality. Containers have very strong security implications and can contribute to limiting access to a system in the eventuality of a security breach. This study contains nothing confidential or any non-disclosed flaw with containers that can be misused.

10. Conclusions

In this study it was proven that Docker showed better utilization of shared resources compared to Jails. For the most part Docker performed better than Jails when running multiple containers, especially when running thirty-two containers simultaneously. While running one or few containers Jails are competitive with Docker and in some cases perform better. Another major finding was that Docker showed a higher stability performance compared to Jails, which also contradict earlier research. Results showed that Jails had much higher spikes regarding variance in results, while Docker showed more stable variance in results, with fewer spikes.

For most situations Docker appears to be the superior choice in terms of performance, also given that Docker has much more industry support and momentum behind it as a platform makes for a compelling argument in favour of Docker. However if there is a specific use case where a machine container approach is preferred or where Jails on FreeBSD performs better, Jails should be used. Results of this study show that these use cases could be when network performance with fewer containers, or startup times efficiency are desired. Another situation could be that there is a specific feature or package that only exists on a FreeBSD system that is sought after then a Jails container could be an ideal situation to use Jails. An example of this could be the filesystem ZFS, which still is experimental under Linux and is advised against for production use.

Suggestion for further research could be to investigate how different real applications perform under various different operating systems and different container technologies. A future improvement to this work could be to use a common file system to run the write and read to disk performance tests on. This could possibly be done by comparing Jails on FreeBSD that runs on the filesystem ZFS and Docker on a Linux host installed with the filesystem ZFS. However at the time of writing the ZFS filesystem support on Linux is not as stable as other alternatives on Linux.

References

[1] ”What is Virtualization? | IBM”. [Online]. Available at: https://www.ibm.com/cloud/learn/virtualization-a-complete-guide. [Date accessed: 23-mar-2020].

[2] Á. Kovács, ”Comparison of different Linux containers”, i *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, 2017, s. 47–51, doi: 10.1109/TSP.2017.8075934.

[3] J. Watada, A. Roy, R. Kadikar, H. Pham, och B. Xu, ”Emerging Trends, Techniques and Open Issues of Containerization: A Review”, *IEEE Access*, vol. 7, s. 152443–152472, 2019, doi: 10.1109/ACCESS.2019.2945930.

[4] F.-X. Puig, J. J. Villalobos, I. Rodero, och M. Parashar, ”Exploring the Potential of FreeBSD Virtualization in Containerized Environments”, i *Proceedings of the10th International Conference on Utility and Cloud Computing - UCC '17*, Austin, Texas, USA, 2017, s. 191–192, doi: 10.1145/3147213.3149210.

[5] ”Docker frequently asked questions (FAQ)”, Docker Documentation, 20-mar-2020. [Online]. Available at: https://docs.docker.com/engine/faq/. [Date accessed: 23-mar-2020].

[6] C. Hoffman, ”What Is Unix, and Why Does It Matter?”, *How-To Geek*. https://www.howtogeek.com/182649/htg-explains-what-is-unix/ [Date accessed 7-apr-2020].

[7] Z. Kozhirbayev och R. O. Sinnott, ”A performance comparison of container-based technologies for the Cloud”, *Future Generation Computer Systems*, vol. 68, s. 175–182, mar. 2017, doi: 10.1016/j.future.2016.08.025.

[8] D. Stiawan, Mohd. Y. Idris, och A. H. Abdullah, ”Attack and Vulnerability Penetration Testing: FreeBSD”, TELKOMNIKA, vol. 11, nr 2, s. 399, juni 2013, doi: 10.12928/telkomnika.v11i2.942.

[9] ”Docker alternatives at a glance”, IONOS Digitalguide. [Online]. Available at: https://www.ionos.com/digitalguide/server/know-how/docker-alternatives-at-a-glance/. [Date accessed: 22-mar-2020].

[10] G. Lencse och S. Répás, ”Performance analysis and comparison of four DNS64 implementations under different free operating systems”, Telecommun Syst, vol. 63, nr 4, s. 557–577, dec. 2016, doi: 10.1007/s11235-016-0142-x.

[11] E. Carter, ”Sysdig 2019 Container Usage Report: New Kubernetes and security insights”, *Sysdig*, okt. 29, 2019. https://sysdig.com/blog/sysdig-2019-container-usage-report/ [Date accessed: 8-apr-2020].

[12] J. Paul, ”Using ZFS Filesystem on Ubuntu 19.10”, https://itsfoss.com/. https://itsfoss.com/zfs-ubuntu/ [Date accessed: 21-may-2020].

[13] Morabito, Roberto. ”Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation”. *IEEE Access* 5 (2017): 8835–50. https://doi.org/10.1109/ACCESS.2017.2704444.

[14] R. Morabito och E. Research, ”Power Consumption of Virtualization Technologies: an Empirical Investigation”, s. 6.

[15] Felter Wes, Ferreiara Alexandre, Rajamony Ram, Rubio Juan ”An updated performance comparison of virtual machines and Linux containers”, *IEEE Access*, vol. 7, s. 978-1-4799-1957-4 2019, doi: 10.1109/ACCESS.2020-03-25

[16] V. Balasubramanian Sekar, V. Patil, M. Giusti, A. Bhide, och A. Gupta, ”AWS EC2 vs. Joyent’s Triton: A Comparison of Docker Container-hosting Platforms”, i *Proceedings of the 8th Workshop on Scientific Cloud Computing - ScienceCloud ’17*, Washington, DC, USA, 2017, s. 33–36, doi: 10.1145/3086567.3086572.

[17] ”What are Containers? – Benefits and Use Cases | NetApp”. [Online]. Available at: https://www.netapp.com/us/info/what-are-containers.aspx. [Date accessed: 23-mar-2020].

[18] Kamp och Watson - Jails Confining the omnipotent root, 2000 Phoul-henning Kamp, Rober N. M. Watson IN proc. 2nd Intl. SANE Conference, s2

[19] What is: chroot – system call and utility in Linux Available at: https://rtfm.co.ua/en/what-is-chroot-the-system-call-and-utility-in-linux/[Date accessed: 29-mar-2020]

[20] ”Application Containers vs. System Containers: Understanding the Difference”, *Sumo Logic*. https://www.sumologic.com/blog/application-containers-vs-system-containers-understanding-difference/ [Date accessed: 18-may-2020].

[21] Morabito, Roberto. ”Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation”. *IEEE Access* 5 (2017): 8835–50. https://doi.org/10.1109/ACCESS.2017.2704444.

[22] Datadog - Docker adoption Available at: https://www.datadoghq.com/docker-adoption/ [Date accessed: 29-mar-2020]

[23] ”Docker Blog - Introducing the docker in dex”, Docker. https://www.docker.com/blog/introducing-the-docker-index/ [Date accessed: 18-may-2020].

[24] ”Best practices for writing Dockerfiles”. https://docs.docker.com/develop/develop-images/dockerfile_best-practices/ [Date accessed: 27-may-2020].

[25] ”Docker overview”. https://docs.docker.com/develop/develop-images/dockerfile_best-practices/ [Date accessed: 27-may-2020].

[26] S. Weisz och M. Carabas, ”FreeBSD Virtualization-Improving block I/O compatibility in bhyve”.

[27] ”Linux vs FreeBSD | Learn the Key Differences of Linux vs FreeBSD”, EDUCBA, 16-jan-2020. [Online]. Tillgänglig vid: https://www.educba.com/linux-vs-freebsd/. [Date accessed: 23-mar-2020].

[28] ”What is FreeBSD? Why Should You Choose It Over Linux?”, 07-juli-2018. [Online]. Available at: https://www.fossmint.com/what-is-freebsd-why-should-you-choose-it-over-linux/. [Date accessed: 22-mar-2020].

[29] FreeBSD Handbook 1995 - 2020 Ch14 Available at: https://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ [Date accessed: 28-mar-2020]

[30] ”iocage”. https://www.freebsd.org/cgi/man.cgi?query=iocage [Date accessed: 23-may-2020].

[31] Plugins — iocage 1.2 documentation”. https://iocage.readthedocs.io/en/latest/plugins.html [Date accessed: 23-may-2020].

[32] Mohd Bazli Ab Karim ; Jing-Yuan Luke ; Ming-Tat Wong ; Pek-Yin Sian ; Ong Hong ”Ext4, XFS, BtrFS and ZFS Linux file systems on RADOS Block Devices (RBD): I/O performance, flexibility and ease of use comparisons”. *ISBN*: 978-1-5090-2603-6.

[33] ”Benchmarking ZFS On FreeBSD vs. EXT4 & Btrfs On Linux”, Phoronix. https://www.phoronix.com/scan.php?page=article&item=zfs_ext4_btrfs&num=5 [Date accessed: 18-may-2020].

[34] Imysql.com. 2020. [online] Available at: https://imysql.com/wp-content/uploads/2014/10/sysbench-manual.pdf [Date accessed: 9-apr-2020].

[35] ”iPerf - The TCP, UDP and SCTP network bandwidth measurement tool”. https://iperf.fr/ [Date accessed: 9-apr-2020].

[36] ”Welcome to Python.org”, *Python.org*. https://www.python.org/ [Date accessed: 29-may-2020].

[37] ”Top (GNU Coreutils)”. https://www.gnu.org/software/coreutils/manual/html_node/index.html#SEC_Contents [Date accessed: 10-may-2020]..

[38] ”Release Information”. https://www.freebsd.org/releases/ [Date accessed: 14-apr-2020].

[39] ”Releases - Ubuntu Wiki”. https://wiki.ubuntu.com/Releases [Date accessed: 14-apr-2020].

[40] ”Sysbench - Gentoo Wiki”. https://wiki.gentoo.org/wiki/Sysbench#Using_the_CPU_workload [Date accessed: 13-apr-2020].

[41] ”Your Guide to Qualitative and Quantitative Data Analysis Methods”, *Atlan | Humans of Data*, sep. 05, 2018. https://humansofdata.atlan.com/2018/09/qualitative-quantitative-data-analysis-methods/ [Date accessed: 21-may-2020].

Appendix 1: Time Plan

Week 1: Proposal
Week 2: Intro, Problem Statement, Research Questions
Week 3: Background, Related Work
Week 4: Research Methodology, Limitations
Week 5: Testing & Results
Week 6: Discussions
Week 7: Conclusion & Future Work
Week 8: Abstract
Week 9: Prepare for seminar
Week 10: Examination and Publication

Appendix 2: Contributions

Christian researched, configured and wrote scripts for Jails on FreeBSD. Christian also conducted all performance tests, compiled the results and created related graphs.

Rickard researched, configured and wrote scripts for Docker on Linux. Rickard also wrote the benchmarking program IO-nugget.

Appendix 3: Source Code

The scripts to run the experiments from this paper are available at:
https://github.com/christianryding/bachelor-thesis
https://github.com/Maokei/bsdvlinux/