Exploring the Potential of FreeBSD Virtualization in Containerized Environments

Francesc-Xavier Puig, J. J. Villalobos, Ivan Rodero*, Manish Parashar
Rutgers Discovery Informatics Institute (RDI^2), Rutgers University, Piscataway, New Jersey
*irodero@rutgers.edu

ABSTRACT

Enterprise and Cloud environments are rapidly evolving with the use of lightweight virtualization mechanisms such as containers. Containerization allow users to deploy applications in any environment faster and more efficiently than using virtual machines. However, most of the work in this area focused on Linux-based containerization such as Docker and LXC and other mature solutions such as FreeBSD Jails have not been adopted by production-ready environments. In this work we explore the use of FreeBSD virtualization and provide a comparative study with respect to Linux containerization using Apache Spark. Preliminary results show that, while Linux containers provide better performance, FreeBSD solutions provide more stable and consistent results.

1 INTRODUCTION

Containerization, also called container-based virtualization, is being widely adopted in industry, academia, scientific communities and Cloud environments [3]. For example, Google took its container technology to the next level and adopted Ubuntu to run Docker containers, among others. In addition to solving long standing software development portability issues and increasing performance compared to virtual machines, containers are also able to live-migrate in multi-cloud deployments and there are many mature systems available for deployment automation, scaling and management of containerized applications such as Mesos, Kubernetes, Marathon, Amazon's EC2 Container Service and IBM Bluemix. Linux and FreeBSD, two Unix-like operative systems, are living in very different realities today. Linux, without any doubt, is the most widespread Unix-like operating system in all areas, whether business or academia environments, and has been supported by many companies. Nevertheless, FreeBSD [2] has not found its place in the market and it is only used in some specific environments despite proved to be more stable and even faster than Linux in some scenarios. Although FreeBSD adapted featuring of engine versions and support of mature, stable and secure virtualization mechanisms such as Jails, with the emergence of Clouds and Big Data, Linux has been positioned ahead and most of the production-ready containerized solutions are based on Linux. Furthermore, most of the existing related literature focuses on Linux-based containerization technologies such as Docker and LXC [1] as a viable mechanism to improve productivity, resource consolidation and workload scheduling.

This effort aims at characterizing the execution of production workloads on both Linux and FreeBSD operating systems. We specifically use Apache Spark workloads as a driving use case. Other factors taken into account for the exploration of the system design space are the use of operating system virtualization (through LXC containers in Linux and Jails in FreeBSD) and the programming language used on top of Apache Spark (i.e., Python and Scala). A better understanding of virtualization mechanisms is critical as most of the current and ongoing systems exploit virtualization to improve resource utilization; however, the potential overheads/tradeoffs need to be quantified and incorporated into the system models.

2 EVALUATION METHODOLOGY

In the proposed architecture, the physical nodes host the virtual nodes and are primarily responsible for managing the storage export services for these nodes. The physical nodes are the only ones with direct access to physical resources and make up GlusterFS or HDFS volumes. These volumes are exposed to the virtual nodes through its own physical host. Two different types of nodes are considered: (1) Master: nodes are interconnected via Apache Zookeeper to implement a highly available (HA) cluster – in the evaluation described below the HA cluster has 3 master nodes, and (2) Slaves/Workers: nodes are connected to the Master to form the Apache Spark cluster – the number of these nodes varies and it is important that it can scale up easily. The empirical experimental evaluation was conducted in an environment composed of physical servers with 12-core Xeon processors. The workloads used in this effort include Spark PI, PageRank and a set of benchmarks that are part of SparkPerf, which is facilitated by the Spark community. The different workloads are based on typical functions designed for Big Data applications. This work considered the following set of benchmark applications from SparkPerf (TestRunner) : scheduling-throughput, aggregate-by-key, aggregate-by-key-int, aggregate-by-key-naïve, sort-by-key, sort-by-key-int, count, and count-with-filter. The selected benchmark applications have been configured for optimal performance and have been evaluated using versions written in Scala and Pyton programming languages, which adds another dimension to the comparison.

3 PRELIMINARY RESULTS

The results obtained from the experimental evaluation in the environment described above show that the executions in Linux are slightly faster than in FreeBSD. They also demonstrate that container-based virtualization is a viable option as it does not greatly impact performance on these executions. Figure 1 provides

UCC'17, December 5–8, 2017, Austin, Texas, USA

ACM ISBN 978-1-4503-5149-2/17/12.
https://doi.org/10.1145/3147213.3149210

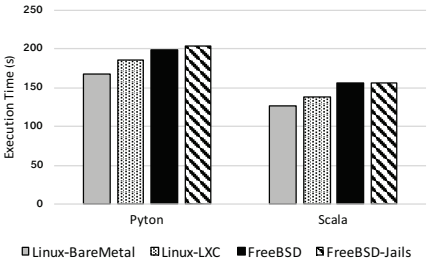

Figure 1: Execution of Aggregate By Key on Linux/LXC and FreeBSD/Jails using Python and Scala

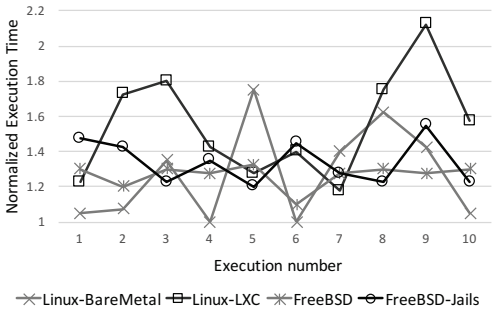

Figure 2: Normalized execution time (to minimum value) of ten executions of Spark PI with different configurations

the results for the aggregate-by-key benchmark, which is representative as the results for the other benchmarks follow a similar trend. The execution behavior is very similar for bare metal and virtualized environments for both Linux (LXC) and FreeBSD (Jails). As expected, the results show that Scala implementations run faster than Python implementations and Scala is more stable than Python, which is consistent for the different environments.

Figure 2 shows the results obtained with Spark Pi, which uses a Monte Carlo method. This method estimates the value of Pi by performing an aggressive parallelization. The goal of this evaluation is studying the variability of the results with moderate network and file system I/O utilization. The figure shows the results for a total of 10 executions per cluster and configuration. The executions were initiated via launcher (cluster mode) and a script was used to run a new instance every 10 seconds until the end of the executions. The results show that the outcomes using Linux are quite irregular (i.e., high variability in execution time) with both bare metal and LXC container-based configurations. Conversely, FreeBSD is more regular with average execution time values similar to Linux. This behavior is exacerbated with a larger amount of processor cores, i.e., the variability with Linux is higher with a larger core count.

Figure 3 shows the results obtained with PageRank, which is an algorithm to assign a numerical weight to the relevance of the elements of an indexed set. One use case of this algorithm is Google's search engine results page. The evaluation was conducted with two different input data sets, the first one of only 72MB size and the second one of 1.1GB size. The executions were initiated via

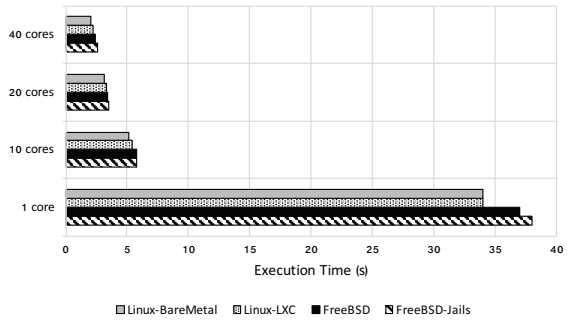

Figure 3: Execution time of PageRank with different number of compute cores and configurations

launcher (cluster mode). The results show that executions on Linux are faster than those on FreeBSD. The figure also shows that the difference between Linux and FreeBSD is consistent as the number of cores increases. This is also consistent with the results shown in Figure 1 (i.e., more I/O-bound workload).

4 CONCLUSION AND FUTURE WORK

In this paper, we explored the potential of FreeBSD virtualization (i.e., Jails) with respect to Linux-based solutions. We deployed a proof-of-concept architecture on both Linux and FreeBSD and compared the execution time of Spark workloads in these two environments. Preliminary results show that, while Linux containers provide better performance, FreeBSD solutions provide more stable results. It supports that FreeBSD virtualization can be a good candidate for relevant containerized systems (e.g., when execution time variability is not tolerable). This motivates us to explore FreeBSD usage modes in both Cloud-oriented infrastructures but also for next generation HPC deployments (e.g., high performance big data analytics). Our ongoing work includes a more comprehensive characterization of containerization technologies, exploring other tradeoffs such as those related to energy/power and resource utilization, and characterizing different big data processing frameworks (e.g., streaming solutions) with different operating system and containerization choices. This is especially interesting in order to understand appropriate design choices and policies that can optimize energy consumption and manage power budgets.

ACKNOWLEDGMENTS

This research is supported in part by NSF via grants numbers ACI 1464317, ACI 1339036 and ACI 1441376. This research was conducted as part of the Rutgers Discovery Informatics Institute (RDI^2).

REFERENCES

[1] Emiliano Casalicchio and Vanessa Perciballi. 2017. Measuring Docker Performance: What a Mess!!!. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion (ICPE '17 Companion)*. ACM, New York, NY, USA, 11–16. https://doi.org/10.1145/3053600.3053605

[2] G. McKusick, M.and Neville-Neil and R. Watson. 2014. *The Design and Implementation of the FreeBSD Operating System, Second Edition*. Vol. 42. Pearson Education.

[3] S. J. Vaughan-nichols. 2006. New Approach to Virtualization Is a Lightweight. *Computer* 39, 11 (Nov 2006), 12–14. https://doi.org/10.1109/MC.2006.393