Performance of Data Mining, Media, and Financial Applications under Private Cloud Conditions

Dalvan Griebler*†, Adriano Vogel†, Carlos A. F. Maron*†, Anderson M. Maliszewski*, Claudio Schepke‡ and Luiz Gustavo Fernandes†

*Laboratory of Advanced Research on Cloud Computing (LARCC),
Três de Maio Faculty (SETREM), 2405, Santa Rosa Av.– Três de Maio – RS – Brazil
†Pontifical Catholic University of Rio Grande do Sul (PUCRS), 6681, Ipiranga Av. – Porto Alegre – RS – Brazil
‡Laboratory of Advanced Studies (LEA), Federal University of Pampa (UNIPAMPA) – Alegrete – RS – Brazil
Email: {dalvan.griebler,adriano.vogel,carlos.maron}@acad.pucrs.br,
andersonmaliszewski@gmail.com, claudioschepke@unipampa.edu.br, luiz.fernandes@pucrs.br

***Abstract*—This paper contributes to a performance analysis of real-world workloads under private cloud conditions. We selected six benchmarks from PARSEC related to three mainstream application domains (financial, data mining, and media processing). Our goal was to evaluate these application domains in different cloud instances and deployment environments, concerning container or kernel-based instances and using dedicated or shared machine resources. Experiments have shown that performance varies according to the application characteristics, virtualization technology, and cloud environment. Results highlighted that financial, data mining, and media processing applications running in the LXC instances tend to outperform KVM when there is a dedicated machine resource environment. However, when two instances are sharing the same machine resources, these applications tend to achieve better performance in the KVM instances. Finally, financial applications achieved better performance in the cloud than media and data mining.**

***Index Terms*—Cloud Computing; Performance Benchmark; Infrastructure as a Service; Virtualization.**

I. Introduction

Cloud computing is a paradigm capable of providing Infrastructure as a Service (IaaS), where users/clients may perform on-demand computational resource provisioning (CPU, memory, storage), either for public or private cloud [1] deployment models. IaaS is the base service model for a cloud computing environment because it supports upper/higher-level service models such as Platform as a Service (PaaS) and Software as a Service (SaaS) [2]. In the lowest level of a cloud, there is the virtualization layer that allows for the necessary dynamism of the service models. Therefore, cloud users have elastic resource provisioning and only need to pay-per-use. [3].

Real-world applications are moving to cloud computing data-centers. Among them are high-performance and enterprise applications that can take advantage of the cloud characteristics. However, performance for the majority of the cloud platforms is difficult to predict, due to performance variations and load fluctuations caused by their multi-tenant environment and software stacks (*e.g.*, virtualization and drivers) [4], [5], [6]. Additionally, the literature lacks more in-depth and empirical investigations of how to better exploit and optimize the benefits of clouds.

Analyzing and studying applications' performance for different cloud deployments is fundamental to provide important insights for cloud providers and potential users. We chose to study the performance of a set of PARSEC benchmarks under private cloud conditions because they represent real-world multi-threading applications. Consequently, many other applications with similar characteristics may follow the performance trends of our experiments. In addition to cloud users, cloud providers can also benefit from our results, to identify the best way to deploy the cloud based on the chosen application. For instance, the provider could ask users for their applications' characteristics and then offer custom deployments to achieve optimized performance.

Previous works on performance in cloud environments have analyzed the benefits of moving enterprise applications to public cloud providers [7], and others have characterized cloud performance through benchmarking [8], [9]. Furthermore, studies have evaluated the feasibility of cloud infrastructures for running scientific applications [10]. For instance, they traditionally use the NAS benchmarks [11], [12]. These results are not representative of enterprise applications, although they are extremely important for the High-performance computing (HPC) area. We contribute different aspects and analysis to this research field,including an in-depth evaluation of applications and cloud environments.

Our goal is to evaluate the performance of PARSEC benchmarks that represent main stream and real-world enterprise applications. We focus on a private cloud scenario by deploying a CloudStack cloud as the infrastructure manager and enabling two different types of cloud instances (KVM and LXC). Also, we investigate the differences and benefits by running the applications in the cloud instances that may have dedicated or shared machine resources. Our main research contributions are summarized as follows:

- A performance analysis of PARSEC applications (from media, financial and data mining domains) running in private cloud instances. These empirical results are valuable for cloud users and providers that are interested in providing better quality service and/or reducing costs. The data and analysis can also help others researchers to improve cloud abstraction layers and minimize performance degradation when running/moving these applications to the cloud.

- A performance comparison of different types of cloud instances (container and hypervisor). This comparison contributes by identifying the best combination of application and virtualization technology in the cloud.
- An analysis of different cloud instances using dedicated and shared machine resources. Sharing the same machine resources among cloud instances is a recurring situation in the cloud. Thus, our analysis highlights the performance impacts for these applications and the efficiency of resource isolation in LXC and KVM.

We have organized our paper as follows. Section II provides the background for the paper. Methodology and cloud deployments of our experimental setup are presented in Section III. Section IV shows performance results. Section V highlights our findings in different scenarios and environments. Related works are presented in Section VI. In Section VII we provide the conclusion and future work.

II. Background

A. Private Cloud and Virtualization Technologies

We address performance aspects on private cloud with open sources solutions [1]. We used Apache CloudStack [13] to manage the cloud infrastructure. This cloud platform offers management of computational resources through an environment capable of offering access in the form of services.

The cloud instances were deployed with hypervisor-based virtualization (KVM) and OS-level virtualization (LXC). The main difference between KVM and containers is that LXC provides resources and the user processes run directly on the host operating system, while the KVM provides virtualized hardware where the guest OS is installed.

In KVM, hardware devices can be virtualized with QEMU device emulator. KVM uses a kernel module to intercept I/O requests from Linux and transfers to QEMU, which translates these requests into system calls. KVM also supports para-virtualization drivers, which often achieve better performance. The guests receive a virtual and isolated environment.

In LXC, user processes are executed directly on the Linux Kernel without a virtualization layer. Seeking performance improvements, LXC offers less isolation between containers. Containers share caches and other data structures at OS-level as well as activities inside a container that may impact on the performance of other containers through shared data structures. Finally, the resources are controlled by a mechanism called cgroup, which determines the resource limits of each container.

B. The PARSEC Application Set

In order to assess the performance and feasibility when running real-world applications on cloud environments, we chose the PARSEC benchmark. The Princeton Application Repository for Shared-Memory Computers (PARSEC[1]) is a benchmark suite composed of a variety of multithreaded applications [14]. This suite includes 13 different benchmarks (10 applications and 3 kernels) [15], which cover different application domains and parallel programming models. Each benchmark aims to simulate the behavior of real-world applications, such as computer vision, video encoding, financial analytic, physics animation, and image processing.

Although the PARSEC suite has been used to evaluate the performance of modern processors, we used it as a representative workload of real-world applications on cloud environments. The benchmark we chose included the Financial domain because of its relevance for enterprise systems, which demand computational capabilities and availability. We also considered the Data Mining domain because of the current high demand for this class of applications both in research and industry. Moreover, we included the Media Processing domain, because this area is increasing its usage and demanding specialized computational resources for image and video processing. For instance, in pattern recognition, rendering, and filtering. Our benchmarks are described below.

Blackscholes is a Financial Analysis application from the Intel RMS benchmark that represents the field of analytic Partial Differential Equation (PDE) solvers. This application is used in financial computations, calculating the amount of floating-point that a processor can perform. We used native input in our tests, which has 10,000,000 reference values [15].

Swaptions is another application from the Financial Analysis domain. It utilizes the Heath-Jarrow-Morton (HJM) framework to compute the evolving interest rate of risk management and equity liability for a predefined class of models. Swaptions utilizes the Monte Carlo (MC) method, which is used to calculate numerical probability based on taking a massive number of random samples several times.

Freqmine is used to simulate the data mining task for Frequent Item set Mining (FIMI), which is very common in applications that perform protein sequences, market data, and log analysis. Its main feature is to find a frequent pattern in a large database by mining a set of frequent patterns using fragmented growth. We used the native input that is composed by a collection of 250,000 HTML web documents. The parallel implementation was done with OpenMP, which has three parallel kernels. Freqmine uses a method called FP-growth that identifies frequently occurring patterns and stores the relevant database transactions in a compact data structure.

Streamcluster is an application used to simulate the data stream cluster (data mining domain). Stream represents continuous incoming data, such as multimedia data, financial transactions, and phone records. The operation of stream clustering organizes data under real-time conditions. Thus, the program spends most of the time evaluating the opening gain of a new center and ways to reduce costs. This operation uses a parallelism with static partitioning of data points and the original data is memory bound and becomes increasingly

[1]http://parsec.cs.princeton.edu

computationally intensive as the dimension increases. Streamcluster computes how to cut costs by opening a new center. It makes a comparison in order to analyze the cost of make a new center or reassigning work from the existing point. The native input used has 1,000,000 points, 200,000 block size points and 128 point dimensions, 10-20 centers, and has up to 5,000 intermediate centers allowed.

Vips belongs to the media processing domain and is based on VASARI Image Processing System (VIPS). It consists of an image processing system for larger images similar to the conventional image processing package. VIPS can evaluate the image in a parallel approach, meld together the operations, requiring no disk space for intermediate images, and no unnecessary disk I/O. VIPS perform image transformation (a common task on desktop computers) and construct a transparent multithreaded image processing pipeline on-the-fly. The image process transformation has 18 stages. The native input for VIPS uses an image with 18,000 x 18,000 pixels. VIPS uses memory-mapped I/O to load on-demand the parts of an input image. Subsequently, the operations are applied to the image region before the output region is written back to disk.

Raytrace is a rendering application that simulates video rendering and 3D scenes. The result is a photorealistic image using a shading model that employs global information to calculate the intensities of the shadows. Therefore, this technique is commonly used in real-time animations, such as movies and computer games. We used the native input that has a method which samples the object surfaces, computes the shadows and reflected light to render the image with HDTV resolution (1920 X 1080 pixels).

III. Experimental Procedure

The experimental methodology relies on passive measurements for evaluating the performance of workloads behaving on private clouds. We chose six PARSEC 3.0 benchmarks, described in the previous section, to represent real-world applications in the cloud. We deployed two private clouds with the same machine configurations, using two popular and growing open-source projects (KVM v.2.0.0 and LXC v.1.0.8). We designed experiments for dedicated and shared machine resource usage. In the experiment with dedicated resources, the performance was collected from a single and isolated instance. On the other hand for shared resources, which better represent a real-world cloud environment with multiple tenants sharing the physical resources, we launched 2 instances running on the same physical machine. The number of threads used were limited to the number of vCPUs available. For instance, in the shared environment, we chose to run PARSEC up to 4 threads in order to scale the performance. Consequently, each instance of the shared environment was sized with 4 vCPUs and 12 GB of RAM (half of the total host resource). This was done to test multi-threaded applications, since we do not focus on overcommitted resources.

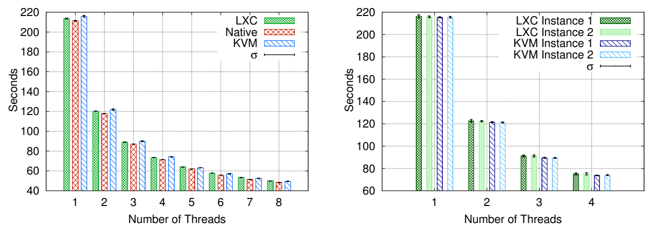
There are some architectural requirements for building an efficient cloud for production environments [16]. In our experiments, we focused on typical and popular cloud deployments in order to assess the performance of applications under private cloud conditions [1], [5]. We used identical machines to normalize the performance between the deployed clouds. Each machine had 24 GB of RAM, two Intel Xeon X5560 quad-core 2.80 GHz processors with the Hyperthreading intentionally disabled, and using disks SATA II. We tested the performance on created instances with the Ubuntu Server 14.04 (kernel 3.19.0) that was also used for the native environment. One host was configured as the cloud manager, using the CloudStack platform version 4.8 and another three hosts as computing nodes.

IV. Performance Evaluation

The results of the experiments with each application are presented below in the form of graphs. We plotted the execution time for each application using up to 8 threads in such a way that each thread has an available core/vCPU. In this approach, we consider cloud instances with a variant number of vCPUs and memory available. Also, the applications executed inside of the instances under a different number of threads. We considered the total execution time for all the applications, not only the time of the parallel region.

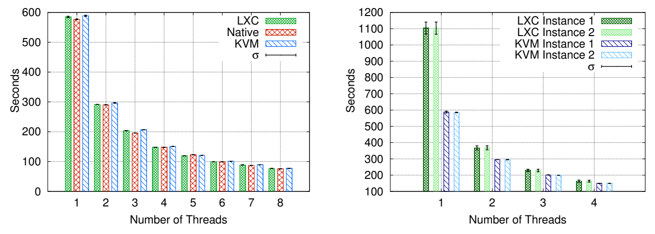
Blackscholes (Figure 1) can be characterized as a CPU intensive application that essentially performs floating-point operations. These operations are performed with 10,000,000 options for financial portfolios and are calculated using a portfolio with 1,000 options, which is pre-initialized in the application. The input portfolios are read from a file in the disk and fully allocated in memory before the processing begins. The I/O operations of this application did not influence the execution time. We observed that KVM instances in dedicated and shared machine resources experiments had the greatest performance degradation in Blackscholes. However, it was not as bad if compared with the native environment results, primarily because KVM is able to run natively the floating-point operations without interference of the KVM drivers. Therefore, the small application's working set assigned to each thread allows the cloud instances to exploit the machine's cache memory. These results also revealed the smaller overheads for the LXC instances with almost native performance due to its light virtualization layer.

Freqmine (Figure 2) processes a large input of transactions from a database. Each transaction is made up of a set of clicks on a web page with different sizes. Due the size of



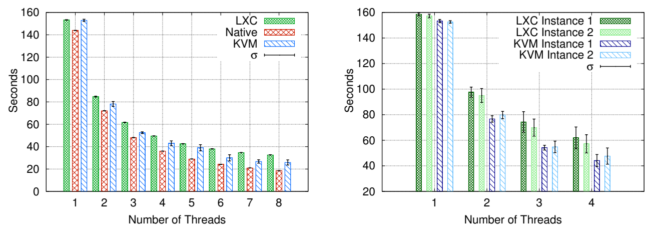
(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Fig. 1. Blackscholes Execution Times.



(a) Dedicated Machine Resources. (b) Shared Machine Resources.

Fig. 2. Freqmine Execution Times.



(a) Dedicated Machine Resources. (b) Shared Machine Resources.

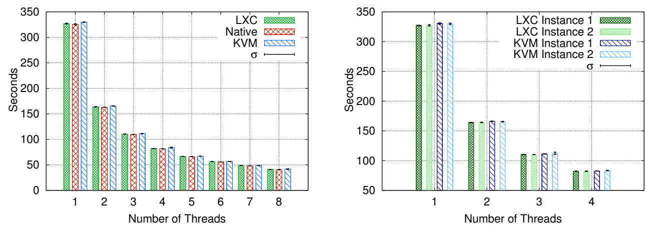
Fig. 3. Vips Execution Times.

the input, the allocation in the memory is gradual, according to the frequency of the processed values. The processing uses intensive memory resources, with constant accesses and loads. In fact, Freqmine reads more than it writes from memory [17]. As a consequence, Freqmine is characterized as a high data sharing because the worker threads process the same data. Figures 2(a) and 2(b) show the performance results of Freqmine execution. Overall, the execution time presented similar results for the dedicated machine resource environment in the LXC, KVM, and Native experiments.

Freqmine presented the same performance in the KVM cloud instances for the shared machine resource environment. In contrast, the results also revealed the greatest performance degradation for the shared machine resource environment in the LXC cloud instances. According to [17], this application has parallelism overheads. After further investigation, we concluded that this overhead combined with the high number of lock synchronizations indicates that the LXC cloud instances sharing the same machine resources have greater overheads in this application because the LXC instances are competing for memory. This issue was also reported by [18], where an observation was made about cgroups interference, which has poor memory isolation.

The performance of the VIPS application is shown in Figure 3. VIPS is a media processing application with data parallelism and coarse granularity. In addition, VIPS has a low data sharing and medium exchange. The results from executing VIPS from the dedicated machine resources showed performance degradation in cloud instances due to the large number of I/O operations that the application performed. These operations are not performed natively because of the additional layer of software that hypervisors apply over VMs. Therefore, it induces performance overheads for I/O intensive applications. In addition, VIPS loads the data from disk to memory and the application's threads communicate during processing phases, which are performing image processing operations. This behavior characterizes an intensive memory and cache usage. When machine resources are shared, we also noted that the application performed better in the KVM instances than in LXC because the latter has an additional overhead caused by poor resource isolation under intensive inter-thread communication and memory use. We discussed the same problem in the Freqmine application previously.

Swaptions (Figure 4) is a coarse grain data-parallel computation with low data sharing and exchanging, performing mainly floating-point operations, which is a similar characteristic of Blackscholes. It breaks the input into small chunks and stores it in the array of portfolios. Then, portions of the array are equally distributed among the worker threads. The results presented in Figure 4(a) and Figure 4(b) show that in both experiments (dedicated and shared machine resources) the execution times were similar among the cloud instance types and native scenario. However, we can observe that this application has a slightly better performance in the LXC instances than in the KVM instances. We can infer the same conclusion from Blackscholes, because they have the same performance and pattern behavior.

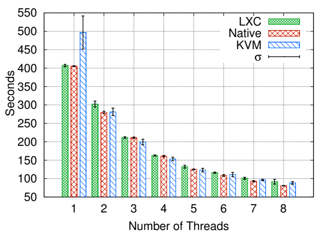


(a) Dedicated Machine Resources. (b) Shared Machine Resources.

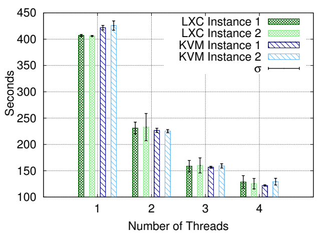
Fig. 4. Swaptions Execution Times.

Streamcluster (Figure 5) is a data mining application with medium granularity and low data sharing characteristics. In the dedicated machine resource environment, the execution times presented contrast between the cloud instance types with high standard deviations. This is a result of the environment and the application's characteristics. The execution of streamcluster with a single thread is different, because the traffic in the cache is only characterized by private read operations and without shared reads when using a single thread [17]. Therefore, in this situation there was low or nonexistent cache usage. Consequently, the aforementioned advantages available in the KVM instances were not exploited when running Streamcluster with a single thread.

Our Streamcluster experiments with shared resources (Figure 5(b)) also revealed performance gains in the cloud instances. With further investigation [17], we identified that it uses the cache memory efficiently for multithreading. In addition, both KVM and LXC can take advantage of Kernel

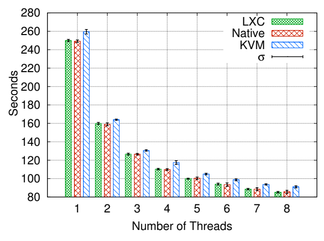


(a) Dedicated Machine Resources.

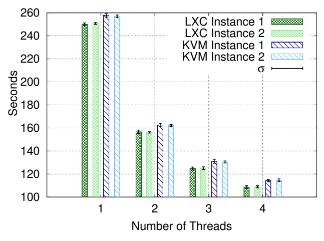


(b) Shared Machine Resources.

Fig. 5. Streamcluster Execution Times.



(a) Dedicated Machine Resources.



(b) Shared Machine Resources.

Fig. 6. Raytrace Execution Times.

Samepage Mergin (KSM)[2] that groups identical data allocated in different virtual machine instances running in the same host [19]. Therefore, we conclude that Streamcluster performed better in the cloud instances due to KSM on shared machine resources.

Another aspect related to the superior performance in the shared environment, is that our machines have Intel VT-x virtualization support [20]. This allows KVM instances to run instructions with native access to the CPU. This technology is specifically impacted in the Streamcluster, because the cache miss rate is minimal (the data size fits in cache) and the majority of operations are loads from cache for the native input that was used. In fact, running instructions with native access to CPU enables an effective cache usage. Also, this application's characteristics combined with the setup of our experiment has resulted in optimized performance.

Figure 6 presents the execution times of Raytrace. It is characterized by data-parallel computations, high data sharing, and low data exchanging. This application presented poor scalability in our experiments. In fact, a significant part of the computations are run sequentially. Raytrace's results show LXC performing well, with native performance in the dedicated machine resource and better than KVM in the shared environment. On the other hand, KVM performed the worst. Raytrace has a large working set that the KVM instances were unable to take advantage of memory and cache optimization. Moreover, there is no significant contrast between dedicated and shared environments. Thus, this application has proven to show suitable characteristics for being run in multi-tenant environments.

V. Final Remarks

We considered private cloud environments that are an attractive alternative for resource-intensive applications, such as PARSEC benchmarks. Therefore, we avoided overcommitting resources when running more than one cloud instance on a single machine, because the system administrator is given control of the private cloud to guarantee better performance. In contrast, related studies addressed performance concerning CPU overcommitment for public cloud environments.

[2]This feature was originally created for hosts that run virtual machines. Recently, it has been incorporated in the Linux Kernel. Thus, LXC is also able to take advantage of it.

Although PARSEC is a set of benchmarks, it can be used to represent real-world application workloads. For example, Raytrace can be found in 3D modelling applications and Blackscholes in financial modeling for several well known applications [21]. Thus, analyzing the findings regarding the performance of these applications running in private cloud infrastructures, can be extrapolated to the behavior of other applications with similar characteristics. Public providers may also benefit from our results, since we used the same virtualization technologies.

Our results show that the performance of an application in a cloud environment varies according to its behavior, environment, and the virtualization technologies used. In some applications, the cloud instances performed well, achieving almost native performance. On the other hand, some applications with specific characteristics presented overhead in the cloud instances. In the dedicated environment, LXC instances on average had slightly better performance than KVM instances. The Blackscholes, Swaptions, and Freqmine results reveal almost native performance, because of the negligible overhead in both cloud scenarios. For instance, comparing to the native performance in Blachscholes with 2 threads, we see a minor difference of 3.2% in KVM instances and 2.0% in LXC instances.

The results of Raytrace and VIPS running in KVM instances demonstrate the impact of data sharing in the cloud. In contrast, we see a performance overhead in KVM instances for Raytrace (up to 6.3% with 4 threads) due to the high data sharing between processing threads. Vips is characterized by low data sharing, thus KVM instances performed better than LXC instances (*eg.*, 3.1% with a single thread in the shared environment). KVM has challenges regarding data sharing between threads within instances [22], although Vips characteristics resulted in KVM instances performing better.

Finally, LXC is an option for performance sensitive workloads because containers often present better performance than full or para-virtualization technologies. Indeed, the performance of PARSEC applications in LXC instances (compared to KVM instances) was better for the dedicated environment in the most of the tests. In some scenarios, KVM outperformed LXC in a shared environment. Therefore, Hypervisor virtualization has proven to be more effective in resource isolation. Furthermore, from the resources management perspective, containers are not always the most suitable alternative because they lack functionalities and have flexibility and compatibility limitations.

VI. Related Work

Performance analysis of applications running in cloud environments has become relevant in the literature. We understand related works as those that share an application scenario or test environment that is similar to our approach. Nikounia and Mohammad [23] evaluated the performance of the PARSEC applications only under a KVM-based virtualized environment. The authors addressed so-called hypervisor noise, which is the performance degradation caused by the use of virtualization. In addition, they assessed neighbors' noise, which refers to interference among VMs running on the same host. They deployed two multi-core machines with the KVM hypervisor installed. The results revealed a significant overhead caused by the virtualization layer. This degradation was caused mainly by the overcommitment of resources, which reduces the processing power available for multithreaded applications. In our paper, we focused on different analysis, including container-based instances as well as instances sharing machine resources. We also selected a set of application domains to in-deep characterize the performance events.

In another related work, Wang et al. [24] address performance isolation at the application level. They also highlighted that multi-tenancy is divided in three parts: shared infrastructure, shared middle-ware, and shared application. The authors present an approach using the Kalman filter to dynamically estimate the application-level CPU consumption of multi-tenant Web applications to determine their performance isolation. Experimental results using TPC-W e-commerce suite on middleware TomCat and database MySQL, showed that the results correspond to the measurements with acceptable estimation errors. In contrast, our research performs an inclusive performance analysis in the enterprise application domains using different workloads. Additionally, we focused on application characteristics instead of the multi-tenant cloud user implications.

The performance of cloud environments has also been analyzed by Iosup et al. [25]. In fact, they have explored cloud feasibility for scientific computing workloads. Their performance evaluation is focused on Many-Task Computing (MTC) applications running on 4 IaaS public cloud providers (Amazon EC2, GoGrid (GG), ElasticHosts (EH), and Mosso). The evaluated cloud providers have poor performance with the applications tested. Despite the challenges regarding performance on cloud environments, cloud is still an alternative in some cases for scientific computing due to quick resource provisioning. Differently, our experiments covered private cloud conditions and multi-threading applications from the enterprise domain.

In Leitner and Cito [26], the performance of workloads running on public cloud providers is also evaluated, regarding EC2, GCE, Microsoft Azure, and IBM. These authors present a literature review concerning performance and its predictability on cloud environments. They validated their hypotheses with experiments. The extensive evaluation demonstrated a significant contrast of performance comparing the cloud providers. Also, the performance impact caused by the cloud multi-tenancy shown to be different among cloud providers. The most commonly used infrastructures such as EC2 and Azure caused higher variations and unpredictability. Again, our experiments covered a different cloud environment with a different application set.

Performance evaluation of Hypervisors deployed in a cloud data-center is presented in Huber et al. [27]. They tested XenServer and VMware ESX. The SPEC, PASSMARK, and Iperf benchmarks used stressed CPU, memory, disk and network performance. The main goal was to evaluate the Hypervisor's scalability and overcommitment. The results show that VMware has a better performance and resource isolation, while the CPU and memory results were similar in both scenarios. In contrast, we focused on real-world workloads and cloud environments (container and kernel-based instances).

Previous works have also studied relevant themes addressing performance through benchmarks in cloud data center infrastructures. The need for benchmarking is emphasized in Iosup, Prodan, and Epema [8], where important aspects such as methodology, system property, workload, and metrics are presented and described, which were taken into account in our experiments. Their approach proposed experiments using more complex workloads in order to be representative for cloud users. They also addressed aspects related to multi-tenancy on cloud environments. Furthermore, Folkerts et al. [9] also highlighted relevant aspects (*e.g.*, metrics, workloads, requirements) for benchmarking the cloud performance. In this paper, we aim to assess the feasibility of cloud environments for PARSEC applications using the aforementioned best practices.

Moreover, Nikounia and Mohammad [23] presented a performance evaluation of PARSEC benchmarks which addressed the problem of overcommitment. On the other hand, we evaluated performance isolation of cloud infrastructures. We also go even further by evaluating the combination of different applications for a multi-tenancy scenario, and comparing different virtualization deployments (KVM and LXC) as well as the impacts with respect to the bare-metal/native environment.

In contrast to Iosup et al. [25], Leitner and Cito [26], and works that have evaluated performance aspects on public providers, we are addressing the performance of private cloud environments using open-source virtualization technologies. Our motivation is that many organizations have their own clouds to achieve a more customized and dedicated environment. Huber et al. [27] and other studies have evaluated the performance aspects of virtualization technologies, but not necessarily considering the cloud platform. Also, we exploited representative workloads/applications for private cloud deployments using KVM and LXC virtualization, instead of Xenserver and VMware ESX. Finally, our cloud instances were deployed with CloudStack.

VII. Conclusion and Future Works

This paper has presented a performance analysis of representative application domains (data mining, financial analysis, and media processing). The cloud instances covered LXC and KVM options, as well as the use of dedicated and shared physical machine resources. Our experiments have shown that performance varied according to the application's characteristics, virtualization technology, and cloud environment. We concluded that applications running in the LXC instances tended to outperform the KVM instances in a dedicated machine resource environment. However, when there are two instances sharing the same machine resources, these applications tend to achieve better performance in KVM.

The financial domain proved to be more suitable for cloud environments in terms of application characteristics, with almost native performance. On the other hand, media processing applications had high variations in the cloud environments, which will be a challenge for improving performance. In data mining, although there is a high I/O operations, the cloud performance and resource isolation has been better than in media processing.

In the future, we plan to evaluate other application domains to discover different and additional behaviors, perform experiments with an overcommitted environment using the same benchmarks, and analyze the performance impact of the benchmarks with other virtualization technologies (e.g, HyperV, VMWare, Xen, Docker).

Acknowledgment

The authors would like to thank for the partial support from HiPerfCloud project, CAPES, FAPERGS, and institutional support of SETREM, PUCRS and UNIPAMPA.

References

[1] A. Vogel, D. Griebler, C. A. F. Maron, C. Schepke, and L. G. Fernandes, "Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack," in *24th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. Heraklion Crete, Greece: IEEE, Febuary 2016, pp. 672–679.

[2] P. Mell, T. Grance *et al.*, "Sp 800-145. the nist definition of cloud computing," Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, United States, Tech. Rep., 2011.

[3] R. Buyya, C. Vecchiola, and S. Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*, ser. ITPro collection. Elsevier Science, 2013.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[5] A. Vogel, D. Griebler, C. Schepke, and L. G. Fernandes, "An Intra-Cloud Networking Performance Evaluation on CloudStack Environment," in *25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. St. Petersburg, Russia: IEEE, March 2017, p. 5.

[6] C. Rista, D. Griebler, C. A. F. Maron, and L. G. Fernandes, "Improving the network performance of a container-based cloud environment for hadoop systems," in *2017 International Conference on High Performance Computing Simulation (HPCS)*, July 2017, pp. 619–626.

[7] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS," in *2010 IEEE 3rd International Conference on Cloud Computing*, July 2010, pp. 450–457.

[8] A. Iosup, R. Prodan, and D. Epema, "IaaS Cloud Benchmarking: Approaches, Challenges, and Experience," in *Cloud Computing for Data-Intensive Applications*, X. Li and J. Qiu, Eds. New York, NY: Springer New York, 2014, pp. 83–104.

[9] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun, "Benchmarking in the Cloud: What It Should, Can, and Cannot Be," in *Selected Topics in Performance Evaluation and Benchmarking: 4th TPC Technology Conference, TPCTC 2012, Istanbul, Turkey, August 27, 2012, Revised Selected Papers*, R. Nambiar and M. Poess, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 173–188.

[10] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*. IEEE, 2009, pp. 4–16.

[11] K. Kourai and R. Nakata, "Analysis of the Impact of CPU Virtualization on Parallel Applications in Xen," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 3, Aug 2015, pp. 132–139.

[12] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, and R. Biswas, "Performance Evaluation of Amazon EC2 for NASA HPC Applications," in *Proceedings of the 3rd Workshop on Scientific Cloud Computing*, ser. ScienceCloud '12. New York, NY, USA: ACM, 2012, pp. 41–50.

[13] N. Sabharwal, *Apache CloudStack Cloud Computing*. Packt Publishing, 2013.

[14] N. Barrow-Williams, C. Fensch, and S. Moore, "A Communication Characterization of Splash-2 and Parsec," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 86–97.

[15] PARSEC, "Princeton Application Repository for Shared-Memory Computers <http://parsec.cs.princeton.edu/overview.htm>," 2017, last access Oct, 2017.

[16] B. P. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach," *J. Grid Comput.*, vol. 9, no. 1, pp. 3–26, Mar. 2011.

[17] C. Bienia, "Benchmarking Modern Multiprocessors," Ph.D. dissertation, Princeton University, Princeton, NJ, USA, 2011.

[18] Z. Zhuang, C. Tran, J. Weng, H. Ramachandra, and B. Sridharan, "Taming Memory Related Performance Pitfalls in Linux Cgroups," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, Jan 2017, pp. 531–535.

[19] A. Arcangeli, I. Eidus, and C. Wright, "Increasing Memory Density by Using KSM," in *Proceedings of the Linux Symposium*, 2009, pp. 19–28.

[20] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig, "Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization." *Intel Technology Journal*, vol. 10, no. 3, 2006.

[21] A. Shinde and K. Takale, "Study of Black-Scholes Model and its Applications," *Procedia Engineering*, vol. 38, no. Supplement C, pp. 270–279, 2012, international Conference on Modelling Optimization and Computing.

[22] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, March 2015, pp. 171–172.

[23] S. H. Nikounia and S. Mohammadi, "Hypervisor and Neighbors' Noise: Performance Degradation in Virtualized Environments," *IEEE Transactions on Services Computing*, vol. pp, no. 99, pp. 1–1, 2015.

[24] W. Wang, X. Huang, X. Qin, W. Zhang, J. Wei, and H. Zhong, "Application-Level CPU Consumption Estimation: Towards Performance Isolation of Multi-tenancy Web Applications," in *2012 IEEE Fifth International Conference on Cloud Computing*, June 2012, pp. 439–446.

[25] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, June 2011.

[26] P. Leitner and J. Cito, "Patterns in the Chaos — A Study of Performance Variation and Predictability in Public IaaS Clouds," *ACM Trans. Internet Technol.*, vol. 16, no. 3, pp. 15:1–15:23, Apr. 2016.

[27] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments," in *CLOSER 2011 - Proceedings of the 1st International Conference on Cloud Computing and Services Science*, 01 2011, pp. 563–573.