The Impact of Hypervisor Layer on Database Applications

Daniel Yokoyama, Victor Dias, Henrique Kloh, Matheus Bandini, Fabio Porto, Bruno Schulze, Antonio Mury
Distributed Scientific Computing
National Laboratory for Scientific Computing
Petropolis, Rio de Janeiro 25651–075
E-mail: yokoyama, victord, henrique, mbandini, fporto, schulze, aroberto@lncc.br

Abstract—This paper presents the impact analysis of the hypervisor layer over database applications. The conclusions were reached by performing experiments with hybrid environments, composed by bare-metal servers and two types of hypervisors. The main objectives were to compare the behaviour of database applications and to determine the maximum service capacity of a virtualized database server on different environments. With the results, it was possible to assess the scalability of a database environment, according to the type of hypervisor and the number of simultaneous connections, and decide how and when to expand or reduce the virtualized infrastructure.

I. INTRODUCTION

Cloud computing has been largely adopted commercially as a platform for running applications that require dynamic resource allocation. Particularly, in the case of web applications facing huge fluctuation in the number of running transactions can benefit from the elasticity provided by cloud environments to extend, and conversely shrink, available resources keeping service guarantees. More importantly, from the application point of view, such fluctuation is reflected as a short turn variation cost, avoiding over-investing in hardware and software as a result of seasonal variations.

A basic principle underlying the cloud environment, and elasticity therein, is virtualization. A virtual machine (VM) simulates the resources of a concrete one, by providing applications with a container specially tuned for them. A VM is instantiated into a particular kind of operating system, called hypervisor, responsible for managing the various virtual machines allocated to a concrete (i.e. real or bare-metal) machine. A VM is configured by defining its resources, such as: number of cores, memory, disk space and network devices, in addition to the software supporting the application.

Thus, running an application in a cloud setting involves enabling the VMs configured for the given application. In particular, under stressing conditions, a monitor may enable new instances of a VM, providing elasticity to the affected application. It turns out that, according to the type of application (i.e. CPU intensive, data intensive, scientific), adopting a virtualization approach may incur overheads that may hinder its benefits. Typically, requests for disk and network access involve some indirection that can potentially influence the throughput.

In this paper, we investigate the impact of the virtualization strategy on database intensive applications. Supporting an increasing transaction rate is a complex task involving tailored database tuning, but also involves the measurement and the redistribution of the workload, through database elasticity [1]. In this work, we are interested in evaluating the effect of virtualization on database running traditional OLTP (OnLine Transaction Processing) transactions.

We selected two existing benchmarks, Pg-Bench and DBT-5 (DataBase Test 5), that implement TPC-B and TPC-E, respectively, both of them working with PostgreSQL as DBMS (Database Management System). The first one stresses the database system with the intensive submission of traditional OLTP transactions, requiring the system to sustain increasing I/O requests. The second one explores the increasing number of user connections with intense transaction frequency. In both cases, we evaluate the ability of the system to respond to such workload under a VM scenario. With these tests we aim to evaluate the behaviour of a hybrid environment under intensive transactions and to identify the maximum load service supported by a virtual database server.

We ran experiments in order to compare databases of different sizes, running under two different hypervisors, and taking as a baseline a scenario having one VM instance allocated to a bare-metal machine, i.e. without hypervisor.

Our initial results highlighted the relevance of I/O on the hypervisor’s behaviour, the relationship and impact of hyper-threading technology with the hypervisors and how the right memory and swap memory setting could improve the performance. We were also able to observe a very satisfactory result of the container virtualization technology, through the use of BareVZ, due to the absence of a virtualization layer, resulting in a closer integration with the bare-metal server.

The paper is structured as follows. Section 2 gives an introduction on the foundational concepts used in this paper. Next, section 3 presents the two benchmarks used for evaluation and comparison. Section 4 introduces the experimental set-up and section 5 presents the experiment results and discussions. Finally, section 6 concludes the paper.
II. RELATED WORK

A. Database Virtualization and Benchmark

Database virtualization is an attractive solution to apply on-demand scalability for large data-oriented services, within the context of Cloud Computing, and for the provision of academic and research test environments. However, the virtualization of I/O intensive applications, such as a database environment, may imply in performance effects, depending on the hypervisor choice and the virtualization technology [2].

As an effort to evaluate the performance of database applications, the Transactional Performance Council (TPC) [14] established a series of benchmark algorithms, that use sets of operations (transactions), in order to assess the behaviour of database systems. Each TPC flavour addresses the simulation of a real database intensive usage situation, with distinct sets of operations, to assess the allocation of computing resources under different use cases.

B. Virtualization and Performance

Concerning the loss of performance due to the hypervisor, [3] discuss the performance benchmark analysis between three types of hypervisors. Analysing their results, they noticed that the hypervisors caused a loss up to 33% in peak performance, when compared to the native bare-metal. They also highlighted a high variance degree in one of them, which may impact on the performance and, depending on the applications, causing errors and delays. In their analysis, they showed that one hypervisor presents the best performance result (for Linpack and Fast Fourier Transform benchmarks) while the other has the best results for Bandwidth and Latency tests (for PingPong benchmark).

The above results are supported by the extensive work in [4] that performed a deeper analysis with results pointing to the diverse demands of Cloud environments facing scientific and commercial applications. Clouds demand a review of the metrics used to evaluate the performance of the environments. The existing contention problems (processing, memory and I/O) affect the overall performance, particularly, communication issues. The results of a series of tests indicate other aspects, in addition to communication, such as: the variability of the environment, the cores' context exchange, the type of processor architecture, the hypervisor type and the type of application being executed. A similar conclusion about the influence of communication, contention and type of application is observed in [5], [6] and [7]. In [8], there is a detailed analysis of the performance loss, resulting from the contention effects in a virtualized environment and from the influence of the message type traffic.

The influence of the system architecture and resource contention is well presented in [9], which analyses the data obtained for 2 years and lists the following factors as variability causes: symmetric multiprocessing (SMP) resource contention, communication among and within the nodes, kernel process scheduling, cross-application contention and system activities. These factors become critical if there are multiple virtual machines and, consequently, multiple virtual “environments” competing for the same resources.

The research presented so far shows the need for a deeper analysis of the effects of the hypervisor being used, its relation to the application being executed in the virtual environment and the overall contention effects. This work presents references regarding the load of the virtual environment and its ability to pin threads to the cores, as well as the influence of the variability of the environment in the stability and reliability of the results. They all pose questions about the influence of the hypervisor's management capability, its relationship to the host system, its ability to manage this relation and how the application characteristics may contribute to the performance results.

The work in [10] presents a study of the influence of the hypervisor in a real-time operating system running in a virtual environment. It presents how the setting up of the Linux kernel may reduce losses caused by the hypervisor and its management capability. Another aspect they addressed is the relationship among the application, the hypervisor and the host's hardware characteristics. In addition to Skinner and Krammer’s work, there is the analysis of the effects of multi-core architectures. The work described in [11] states the effects of data temporal locality and spatial locality and the performance degradation according to the stride size used in the read and write memory operations.

[12] also addresses the effects of multi-core architectures and how operating systems (OS) may interfere in the performance. They highlight how the execution time variability influences the OS scheduler, when it tries to balance the executing threads among the cores, mostly when the cores are on the same chip. It points out to cache contention and thread migration across cores. Along with cache contention, they list memory controller contention, memory bus contention and prefetching contention as causes of performance degradation. They also present how the contents degree varies from benchmark to benchmark.

The work [13] adds three other elements at the architectural level, influencing performance regarding multi-core processors: shared cache capacity, shared off-chip bandwidth and DRAM page conflicts, stating the non-deterministic interaction among the cores on a chip or nodes.

C. Findings

The related work cited above are a piece of the research done on database virtualization and benchmark, performance, virtualization effects and multi-core architecture, but they highlight some aspects of Clouds:

1) Clouds may provide resources on demand with cost-effectiveness, albeit with a loss of performance;
2) Some applications are better suited for Cloud environments than others;
3) The variability in the environmental contention impacts the results significantly, and
4) The relationship among the hypervisor type, the application, the hardware architecture and the obtained
performance.

Based on the points highlighted above and the experiments conducted, we show the impact of hypervisors in a database intensive test. We discuss possible strategies to manage the infrastructure avoiding bottlenecks and identifying the optimal scenario to expand or to contract the virtual infrastructure.

III. EXPERIMENTAL SETUP

Two experiments were conducted to analyse the influence of hypervisors over a virtualized database environment, consisting of distributed connections accessing a database server, stored locally:

1) The first experiment aims to assess the effect of the virtualization layer and the type of hypervisor over the database server performance under stress situations.

2) The second experiment simulates (exclusively) virtualized data access environment, analysing the behaviour of a database server, against the gradual increase of the number of connected clients, checking the need of expanding the infrastructure by adding more virtual servers, as soon as the performance degrades.

The second experiment does not intend to assess and compare the performance of a virtual database server with a native bare-metal. The intention is to set a baseline for the number of clients that the server is able to attend, identifying the optimal situation to expand or to contract the virtual infrastructure.

In order to achieve the objectives, we:

- Planned and executed experiments to analyse the behaviour of a hybrid environment, composed by bare-metal and virtual resources, simulating a database intensive access situation;
- Ran experiments regarding two fully virtualized database intensive access environments, based on two different hypervisors, comparing and analysing the results; and
- Based on the experiments’ results, we identified the maximum workload supported by a virtual database server and estimated the best moment to scale the database infrastructure.

A. Benchmarks

These experiments use two different database benchmark packages:

1) The Pg-Bench benchmark [15] is an implementation based on the TPC-B benchmark. The TPC-B benchmark is a database benchmark approved by the Transaction Processing Performance Council (TPC) [14]. The Pg-Bench allows database stress tests, measuring the system transaction capability (transactions per second).

The benchmark was conducted in four different scales (1, 10, 100, 1000), i.e., four databases sizes, where scale 1 corresponds to 100,000 tuples, for each scale, there is a simulation of concurrent access of 1, 2, 4, 8, 16 and 32 users.

2) The second database benchmark package is the DBT-5 (DataBase Test 5), which is an open source implementation of the TPC-E benchmark [14]. The DBT-5 is an on-line transaction processing system (OLTP), executing multiple transaction types, with a balanced mixture of disk input/output and processor usage [17].

B. Infrastructure

The first experiment used 5 servers, each one with two Quad-core Xeon 5520 (2.26GHz), 12 GB of DDR3 RAM (1333), hyper-threading and virtualization instructions and Gigabyte Ethernet connection.

The architecture of the bare-metal servers used in these tests enabled the use of hyper-threading technology (HT). To verify the effects of the HT on a virtual environment, tests were conducted enabling and disabling the HT in the bare-metal servers and in the virtual resources.

Concerning the virtual environment, two scenarios were specified, one consisting of a hypervisor with eight cores, corresponding to the actual number of cores of the bare-metal server, and another with 16 cores corresponding to the total bare-metal and virtual cores created using HT.

The second experiment used 5 servers, each one with two hexa-core Xeon 5650 (2.67GHz), 24 GB of DDR3 RAM (1333), hyper-threading and virtualization instructions and Gigabyte Ethernet connection.

For the comparison, Two hypervisors with different characteristics were used during the tests. Hypervisor1 (VT1) is communication and I/O intensive, while hypervisor2 (VT2) is CPU intensive [3], [16]. These hypervisors were chosen due to their virtualization technology (full virtualization), incorporating libvirt libraries and a standard configuration for the host operating system.

The names of the hypervisors used for the experiments in this paper have been omitted, because the main purpose was to assess the impact of the virtualization layer over a specific application, and not to compare the performance between them.

In the first experiment, we also evaluated the BareVZ virtualization approach, which is characterized by the non-existence of a hypervisor. In this case, virtualization is achieved by the creation of isolated containers, each one corresponding to a virtual environment, running over the bare-metal server.

In the next section we present the results and analysis of the two experiments:

1) The first one evaluates the behaviour of a hybrid environment, composed of a bare-metal and a virtual resource, simulating a database intensive access situation.
and regarding two fully virtualized data-base intensive access environments, based on two different hypervisors, comparing and analysing the results; and

2) Based on the experiments' results, identify the maximum load service supported by a virtual database server and estimate the timing of the insertion of a load balance database server.

IV. Benchmark Results

A. Evaluating the behaviour of a hybrid environment under intensive transactions

The experiments described in this subsection investigate the impact of hypervisors in database applications running in a virtual environment.

1) Scale 1000 - Saturation Region: Figure 1 shows the obtained results under scale 1000, that corresponds to the saturation region of the bare-metal and virtual environments. It is possible to notice that the BareVZ has the best performance among the virtualized environment. The BareVZ performance range from 15% better performance for one user, to practically equal with four concurrent users, up to a loss of performance of 20% for 32 concurrent users if compared to the bare-metal with the HT disabled. If compared to the bare-metal with HT enabled, it presents a gain of performance ranging from 53% with one user, to practically no gain with 16 concurrent users, up to a loss of performance of 7% for 32 concurrent users.

Hypervisor2 has the best performance among the hypervisors, ranging from a 14% (average) better performance with one user to practically no gain (5% in the best case) with two concurrent users and to a loss of performance of 25% with 32 concurrent users, if compared to the bare-metal with the HT enable.

It is important to note that the TPS for scale 1000 shows inferiors results, if compared to the other scales. In this scale, the maximum transaction rate achieves 400 TPS, unlike the other scales that reach up to 55,000 TPS. Note that the slight performance increase observed in 1 is due to the processing capability, in which new transactions are served by available cores.

2) Hypervisor1 and Hypervisor2 comparison: Figures 2, 3 and 4 show the results of scales 1, 10 and 100. An initial analysis of the graphics indicates that scales 1, 10, and 100 have two regions that stand out against the others: the region related to 4 concurrent users and the region of 16 and 32 concurrent users.

In all three scales, for up two concurrent users, Hypervisor2 presented a better performance than Hypervisor1, even matching the performance of the bare-metal.

Figure 5 shows the number of transactions per user for scale 100, considering the Bare-metal, Hypervisor1 and Hypervisor2 with HT enable. We can see that, as the number of concurrent users arises, the bare-metal shows the number of TPS/user between 5000 TPS/user (worst case) up to 7000 TPS/user (best case) until 4 concurrent user. With 8 concurrent users up to 16, it falls to 3000 TPS/user and falls again to 700 TPS/user with 32 concurrent users. Hypervisor2 has 7000 TPS/user (better than Bare-metal) with one user, 5000 TPS/user for
2 concurrent users, falling from 4800 TPS/user until 3000 TPS/user and falling with 4 users and from 800 TPS/user until 400 TPS/user between 8 to 32 concurrent users. Hypervisor1 has the 2000 TPS/user (average) between 1 to 4 users and falls from 1500 TPS/user with 8 concurrent users, until 600 TPS/user with 32 concurrent users.

However, analyzing the region with four concurrent users (Figure 6 shows a sample in Scale 10), it presents a very interesting information regarding the relation and influence between the hosting OS and the hypervisor layer. Although Hypervisor2 presents better performance than Hypervisor1 (2.3 times - average), its variability is much larger, with an average deviation\(^1\) of 3000 TPS if compared to a 450 TPS average deviation for Hypervisor1, and from 8 concurrent users Hypervisor1 exceeds hypervisor2 (1.7 times - average).

To understand the reason for this variability, it is interesting to observe that Hypervisor1 pins its processes to a particular core, as default configuration, and runs them with the same privileges as the OS processes. Hypervisor2, as a default configuration, does not pin its processes, running them as a normal user process by the hosting OS. Thus, during the execution of the process, these differences explain this variability when associated to: the number of cores in each processor (4) and the allocation of the process and pre-emption rules used by

\(^1\)The average deviation measures how much certain values deviate from the average value.

The region is characterized by the change of performance position between Hypervisor1 and Hypervisor2. Hypervisor1 presents 8 concurrent users, the best performance, attributed to a better I/O capability. From this point Hypervisor1 is capable of scaling up to 15000 TPS (best case with 8 cores and HT disable in the host server), while Hypervisor2 enters in its saturation region falling to 5000 TPS (Figures 2, 3 and 4).

3) Hyper-thread performance effects: The region encompassing 16 to 32 users also presents important results. These results are noteworthy, although not linked to the effects of virtualization, the main subject of this work. The analyses of the bare-metal results in this region indicates the effect of enabling or disabling the HT. For the specific case of these tests, within 16 concurrent users for the bare-metal machine, with the HT disabled, resulting in 36% less performance, if compared to the bare-metal machine with the HT enable. With 32 users this situation is reversed, and the bare-metal machine performance with the HT disabled is about 3 times better than the bare-metal machine performance with the HT enable. The reason for this change in the results is due to:

- For 16 concurrent users, since this is a stress test, the HT enabled contributes positively to the management of the processes. Since the number of concurrent stress processes (users) is equal to the number of virtual cores, the hosting OS may schedule each process to a core, avoiding concurrency issues;
- For 32 concurrent users, the number of processes is twice the number of virtual cores, it results in two “stress” processes per virtual core, competing with each other, and causing a significant loss in performance, as shown in figure 7.

Figure 8 presents the same effect reported before, but within a virtualized environment. The virtual machine runs with 8 cores in a bare-metal machine with the HT disabled,
resulting in a better performance than the same virtual machine running in a bare-metal machine with the HT enabled, and still better when running with 16 virtual cores. The reason is the concurrency between the processes at the same core, since the hosting OS is unaware of the real and the virtual cores. When the competition between the processes is not significant, the losses suffered by the virtual environment is around 11%, and as soon as the competition between the processes arises, the maximum loss reaches 75% on a 100% scale.

4) BareVz - performance results: The BareVZ also had the best performance between the virtualized environments for scales 1, 10 and 100. Figure 9 shows its performance in comparison with Hypervisor1 and Hypervisor2. Its performance gain ranged from 17%, Hypervisor2 - scale 100/one user, to 1.7 times Hypervisor1 - scale 10/32 concurrent users.

Figure 10 shows the comparison of the BareVZ with the Bare-metal. BareVZ presented a gain of 23.5% for 1 to 4 users; the same performance for 8 users; a loss of 19% for 16 user scale 1 (major loss for 16 concurrent users) and a loss of 52% for 32 users scale 1 (major loss for 32 concurrent users). These values are based on the comparison between the tests of Bare-metal with HT off and BareVZ running in a HT off server.

5) On-line transaction test results: In the DBT-5 tests (figure 11) it is possible to notice that all 3 virtual environments have almost the same response, ranging the performance loss from 72% (1,2 and 4 concurrent users), to 90% for 8 users and to 68% for 16 users, when compared with the bare-metal environment. For 64 users, the bare-metal and the virtual environments begin to saturate. In this case, the virtual environment has a slight improvement, ranging from 10% (Hypervisor2) to 20% BareVZ. For 128 users all the environments saturate and, although the results for Hypervisor1 is better than for the bare-metal, it only finishes half of the tests.

B. Scale-up tests, identifying the maximum load service supported by a virtual database server

The final experiment set consists of two separate environments. Each environment comprises a copy of the TPC-E database, each one with a different hypervisor: Hypervisor1
and Hypervisor2. Each environment counts with three different network domains, and their clients submitting requests to the database. This was done in order to analyse the behaviour of the hosting hypervisors and database servers, against the gradual increase on the number of connecting clients. The metric used for the analysis of these experiments was the number of Trade-Results Transactions Per Second (TR-TPS). The experiment analyses the behaviour of a virtual database server under stress, receiving requests from three different network domains, simulated by DBT-5, hosted in three different virtual machines. The number of clients varied in 2, 4, 8, 16, 32, 64, 80, 96 and 100 on each of the three domains.

For these experiments, some configuration parameters were adjusted, in order to allow the maximization of the database server usage. It was necessary to modify a PostgreSQL parameter to simultaneous connections, supporting up to over 300 for the three domains (more than 100 each). Moreover, it was necessary to increase the OS kernel shared memory, by changing the maximum size of a shared memory segment (SHMMAX) and the maximum page allocation for shared memory (SHMALL).

In figure 12, considering the results related to TR-TPS for both hypervisors, the experiments presented a regular increase of the service capacity, between 48 clients (16 per domain) and 240 clients (80 per domain), representing the best usage rate of the database server. At this point, two comments can be done about the results: Hypervisor2 presented a more regular behaviour regarding its 3 domains, whereas Hypervisor1 resulted in a difference between domain 1, which access the data locally, and domains 2 and 3, accessing the data through the network; the worst domain of the experiments with Hypervisor1 (domain 1) presented nearly the same results of the 3 domains of Hypervisor2, which means that Hypervisor1 is more effective to manage distributed database access than Hypervisor2.

Above 240 clients, Hypervisor1 tended to limit the performance gain in accordance to the consumption of database server resources, while Hypervisor2’s service capacity decreased drastically due to its limitations for I/O operations demonstrated in [16].
Noteworthy also the fact that, for experiments involving 64 simultaneous connections or more per domain, it has been detected the use of swap memory by the DBMS, for both hypervisors, in order to meet the demand. For this reason, with the results of these experiments, it is possible to conclude that the maximized usage of a database server in a fully virtualized environment occurs when it attends between 192 and 240 simultaneous connections, depending on the system available memory and swap, for both hypervisors.

V. CONCLUSION

For the Database Experiments described in this paper, the use of virtualization should be done carefully, aiming to minimize the performance issues that may occur due to the hypervisor. For OLTP (OnLine Transaction Processing) database servers, it should consider the I/O operations for executed transactions. Moreover, it should also consider how the hypervisors and the operating systems manage the concurrent requests in a virtualized database server.

Regarding the database systems performance evaluation, a few issues, such as the chosen database management system (DBMS) and the hardware architecture, may impact directly on the system’s performance. These features, however, may be controlled by the system’s administrator, as they may be changed taking the necessary precautions, by reducing their criticality for the system. The uncontrollable issue that affects the performance of database systems is the number of clients that access a database server simultaneously. The system’s inability to behave in a scalable way may compromise its integrity, impacting on its performance, or causing services to be unavailable.

Concerning to HT, in case of stress conditions, it is useful as the number of concurrent users is lower than the number virtual cores. When this number exceeds the number of virtual cores it causes a loss of performance. For future work we are planning to disable the OS pre-emption capability and let only HT technology to deal with the core concurrency and compare with the results obtained with bare-metal with HT disable.

Another point is the analysis of 4 concurrent users between Hypervisor1 and Hypervisor2. We can see the variability of Hypervisor2 and how it falls in performance below Hypervisor1 for 8 concurrent users and up. This shows how the user could have significant changes in its virtualized environment, depending on the type of application, workload and the type of used hypervisor; all these must be taken into account.

By analysing these experiments and their results, it is possible to measure if a virtual database server is able to comply with quality criteria, as well as to define if and when a new virtual server is needed, in order to avoid resource idleness. It mainly allows to conclude if it is relevant to use virtualized resources to provide a database server infrastructure, aiming to enable load balance and to reduce costs. For this purpose, arises the possibility of using virtual resources, within the paradigm of cloud computing, as a strategy to make it possible to efficiently manage computing resources. This can be achieved throughout the activation or deactivation of virtual database servers, according to the number of requests, allowing to reduce the acquisition costs and to maximize the usage of physical servers.

ACKNOWLEDGMENT

This work has been supported by Brazilian Ministry of Science, Technology and Innovation (MCTI), the National Counsel of Technological and Scientific Development (CNPq - PCI/LNCC) and Research Foundation of the State of Rio de Janeiro (Faperj - E-26/112.221/2008).

REFERENCES