Optimizing the Management of a Database in a Virtual Environment

Timoteo Lange, Paolo Cemim, Miguel Xavier, Cesar De Rose
Pontifical Catholic University of Rio Grande do Sul - Porto Alegre, Brazil
timoteo.lange@acad.pucrs.br

***Abstract*—Recent studies have demonstrated advantages in using Data Base Management System (DBMS) in virtual environments, like the consolidation of several DBMS isolated by virtual machines on a single physical machine to reduce maintenance costs and energy consumption. Furthermore, live migration can improve database availability, allowing transparent maintenance operations on host machines. However, there are issues that still need to be addressed, like overall performance degradation of the DBMS when running in virtual environments and connections instabilities during a live migration. In this context, new virtualization techniques are emerging, like the virtual database, which is considered a less intrusive alternative for the traditional database virtualization over virtual machines. This paper analyzes aspects of this new virtualization approach, like performance and connection stability during a database migration process and its isolation capabilities. Our evaluation shows very promising results compared to the traditional approach over virtual machines, including a more efficient and stable live migration, maintaining the required isolation characteristics for a virtualized DBMS.**

***Keywords*-Database Virtualization, Virtual Database, Live Migration, Performance Evaluation**

I. Introduction

Virtualization of computational resources has been widely used in Information Technology (IT) in scenarios which varies, including server virtualization, networks, applications and databases consolidation. Server virtualization is one of the oldest virtualization technique that implements a logic layer over the physical layer, in such a way that all physical devices can be accessed by the virtualized servers transparently [1]. Such technique has brought benefit for large data centers including isolation, security and server consolidation, introducing better manageability through news capabilities that permits now to guarantee high availability and load balancing in high throughput systems [2]. In this way, it is possible to notice a trend in attempt to explore those virtualization benefits and its techniques in database systems [2] [3] [4] [5].

Recent studies have demonstrated the advantages of using Data Base Management System (DBMS) in virtual environments, such works have proposed solutions in order to improve manageability as it allows system administrators to consolidate several of DBMS isolated by virtual machines on a single physical machine, reducing the maintenance cost and energy consumption [2]. Furthermore, the high availability of databases can also be addressed by using the live migration technique provided by the virtualization technologies.

Although there are advantages in using DBMS on virtual environments, there are problems that still need to be addressed, such as the problem of loss of client connections and performance degradation when a live migration technique is performed [4] [5]. Furthermore, virtualization by it's very nature introduce a layer which normally results in performance overheads in a whole virtualized system, in such a way the performance of DBMS could also be affected

In this work we conduct experiments in order to compare the performance overheads between the DBMS migration process over a virtual machine (DBVM), and the migration process of a virtual database (VDB) instance. The database instances consolidation is also evaluated through the possibility of reserving used resources per instance, identified as isolation. These analyzes are useful for identifying novel forms management of a database in a virtual environment.

The rest of this paper is organized as follow: Section 2 provides a overview of database approach use; Section 3 describes the migration and isolation process; Section 4 discusses benchmarks, monitoring tools, metrics and the obtained results, the Section 5 shows the related works; The conclusion and future work are show in Section 6.

II. Database Approaches

DBMS is a system class with very specific characteristics that must be considered before a database server can be virtualized [5]. This section makes an overview about databases in virtualized environments identified in this work as DBVM (Database on Virtual Machine) and VDB (Virtual Database).

A database system is basically composed of two components: a set of programs responsible for managing the access to data and the data itself [5]. The former provides interfaces for data creation and manipulation, and other functionalities like security and integrity control. This set of programs is known as DBMS. As mentioned before, the second part of a database is the data itself, that, in general, are arranged in one or more files according to a physical structure proprietary to the DBMS [2].

One characteristic of the current DBMSs is the independency of its logical data structures, such as tables, views and physical storage indexes (file structure). As the physical and logical structures are separated, the physical storage can be managed without interfering in the access to the logical structures. For example, it is possible to rename a physical file in the database without to need to rename its tables that are accessed from a database instance.

The definition of instance is very important for the correct understanding of this work. Instance is the structure used by the programs that compose the DBMS to access the records stored in the data files. The parameter values of an instance can be defined at the moment of its creation. These parameters will allocate the configured amount of memory and operating system resources and start the processes responsible for database manipulation [3].

A. Managing Virtual Database

The virtualization of computational resources is a technique applied in several areas of the Computer Science, such as fault tolerance, high performance and databases [3] [4] [6]. The resource virtualization usually adds a layer between the hardware and applications. Traditionally, this intermediary layer is provided by a Virtual Machine Monitor (VMM), also known as hypervisor, which allows the execution of multiple VMs. Each VM has its own Operational System (OS) and runs isolated from the rest of the VMs.

In this context, the utilization of databases in virtualized environments is usually achieved by executing a DBMS on top of a VM [3]. This technique (in this paper, identified as DBVM) imposes a computational overhead because of the software layers used to provide abstraction of a complete virtual machine. An alternative to reduce the onus related to this approach may be the utilization of virtualization at database level. Figure 1 presents the differences between the virtualization at VM level (Figure 1(a)) and at database level (Figure 1(b)). Instead of virtualizing an entire machine (with its own host operational system), the virtualization at database level

978-1-4799-3755-4/13/$31.00 ©2013 IEEE

virtualizes only an instance of the database (in this paper, identified as VDB).

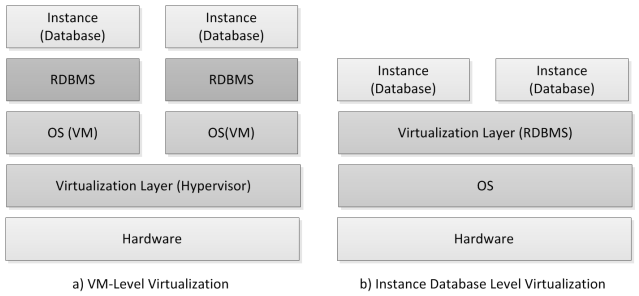


Figure 1. System stack for VM level Virtualization and DB level Virtualization.

The VDB technique is similar to the virtualization based in containers, available in some modern operational systems (i.e. Linux Containers [7], FreeBSD Jails [8] and Solaris Zones [9]). The virtualization based in containers allows the creation of multiple virtual environments that work isolated at user level and share the kernel of the operational system. This type of virtualization usually provides performance similar to native as it does not replicate the entire stack of the operational system [6]. In an analogous manner, virtualization at database level enables the creation of multiple instances of the database, that are isolated from the rest, without the need of a virtual machine that replicates the entire operational system and the DBMS itself.

III. Virtual Database

A. Database Live Migration

Migration of virtual machines is one of the most important functionalities to perform consolidation of data centers. However, with the evolution of VM migration techniques, it is now possible to migrate a virtual machine without shutting down. This process is known as Live Migration [10]. This is an attractive feature for datacenters as it is possible to consolidate servers without breaking SLAs (i.e. high availability clauses) as the migration does not cause downtimes.

Similarly is the case of database migration, except that the virtualization technology has made it un-needed to perform physical data migration. The steps of database migration in virtualized environment differ in case of virtual machine running database (DBVM) from the case of virtual database (VDB). DBVM migration can be done in the same way as the migration of a VM running any application. These migration steps in this case can be found in Section III-A1, and the steps of VDB migration can be found in Section III-A2.

In the scope of databases, it is possible to virtualize it in two different ways, as presented in Section II-A. In this section we will present the different ways to perform virtual database migration, which are: migration of virtual machines running databases and virtual databases.

1) Migration of Databases on VM: VM migration, a technique to migrate tenants with minimal service interruption and no downtime, is critical to allow lightweight elastic scaling [1]. Another definition of VM migration shows that migrating consists basically of transferring its memory image from a source server to a destination server [11] [10]. It is required mainly when organizations or individuals change their computer systems or upgrade to new systems, or when systems merge. The process of migration demands mapping data from the old system to the new one, this mapping relates old data formats to the new system formats and requirements.

The process of migrating a VM running a database can be done in the same way as the migration of a VM running any application. The steps needed to perform the migration are: (i) copying all memory pages from the source host to the destination host. If during the copy a memory page is changed, it is necessary to perform a new copy of the entire page until a threshold where the hypervisor consider that the majority of the page has been copied (Figure 2(a) and 2(b)). (ii) When the copy crosses the threshold mentioned before, the hypervisor suspends the source host and finishes the memory copy to the destination host. This suspension and copy of the pages left in the source host is considered a downtime and may last from milliseconds to several seconds depending on the type of application being executed (Figure 2(c)). (iii) After fully stopping the source host and concluding the copy of memory pages, the destination host starts the migrated VM and from now on it will answer the requisitions made to it (Figure 2(d)).

It is important to note that in this work we do not consider VMs that are in different storage areas. Another important point is that all hosts must be in the same network areas.

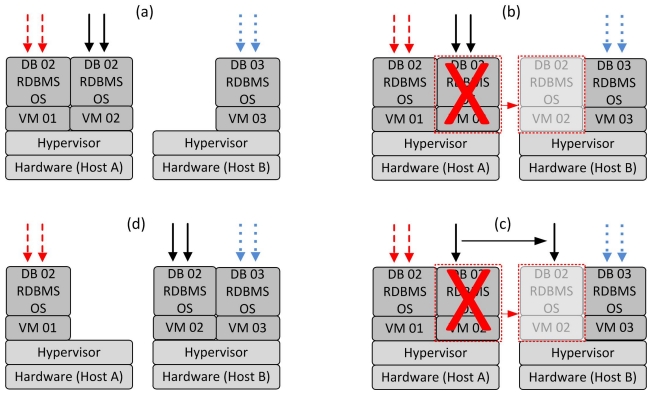


Figure 2. DBVM Live Migration.

2) Migration of Virtual Database: As described in Section II-A, in this new approach, only the database instance is virtualized, which could extend the flexibility through this architecture. Some of the disadvantages of using a DBVM could be minimized as performance loss, keeping the benefits that virtualization could bring and including the consolidation of database systems and easiness of database instances migration between hosts [12].

Differently from using DBMS in a virtual machine, a VDB could be migrated from one ambient to another without the necessity of service interruption and minimal performance loss. The databases could be stored in transparent locations and be relocated to other servers without having to modify the applications.

The migration of a Virtual Database system can be done by the following steps: (i) initialization of a new database instance in the new server (Figure 3(a)), (ii) migration of all the existing connections, transactions and states of each transaction (Figure 3(b) and 3(c)) and (iii) only after the migration of every connection and transaction the original service is stopped (Figure 3(d)). Thus, in this type of migration it is only required to migrate the database instance (provided by the DBMS) and not the entire database (data file) that will remain in the storage.

B. Virtual Database Isolation

One important characteristic of virtualization is the possibility of having several services running on the same machine without interfering. VDB allows the virtualization of databases without the need of a VM layer, decreasing the virtualization overhead. However, the usage of Virtual Database without resources control may result in interference between instances sharing the same hardware.

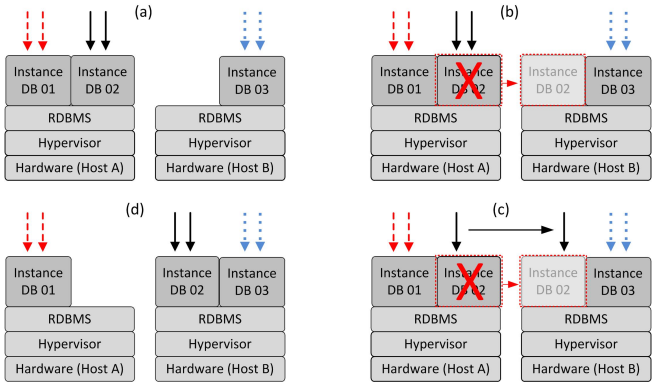


Figure 3. VDB Live Migration.

On a machine running several virtualized databases instances, a CPU-intensive instance may cause dearth in resources and, therefore, significantly degrade the performance of other instances. Thus, isolation solution for CPU consumption interference between virtualized instances Virtual Database. This intervention allows the DBA to limit CPU utilization of databases instances, avoiding one instance to consume CPU allocated for other instances.

Isolation is a necessary resource to ensure performance and security between instances of the virtual database. It can manage multiple facets of session behavior on a per workload basis, such as idle time limits, runaway queries, degree-of-parallelism, etc.

The possibility of running instances of Database Isolated form should allow manages CPU usage by controlling the database load in a very precise way. It does not bind processes to CPUs. Instead, must works much like the OS scheduler, using an VDB run queue and a quantum of time to share the resources between the processes. In a Virtual Database environment, this manages each database instance independently. Thus, each instance can be configured with its own resource limits.

By default, the DBMS the workload in a way that all CPUs are utilized. However, in virtualization is possible to use the Isolation to limit CPU utilization in a database instance.

IV. Evaluations of Database Live Migration and Virtual Database Isolation

A. The Experiment Testbed

The evaluation environment comprises two DELL PowerEdge 810 servers with 2 Intel®Xeon®6500 series for each and a 64Gb RAM memory, interconnected to DELL EqualLogic PS400 storage with iSCSI dedicated protocol. The network is divided into distinct sub-networks, a public one and a private one for direct communication between storage and host, both using a GigaBit Ethernet connection. The OS in use is OEL 5.6 (Oracle Enterprise Linux 5 update 6) with kernel 2.6.18-238.el5. The DBMS in use is Oracle®11g r2. The hypervisor in use is VMware ESXi 5.

In the DBVM evaluation environment, the OS has been set up over VM, and the DBMS has been installed directly in the OS. The database instance has been created applying 100% of the VM available resources. Within this environment, the VM resources are managed by the hypervisor.

Similarly, the VDB evaluation environment worked upon a hardware-setup OS, and the DBMS upon this OS. The instance has been created employing 100% of the available resources, with support to virtualization activated in the DBMS.

The isolation test environment uses the resources addressed above, however, to test a database instance isolation, only a server with multiple database instances running simultaneously was used.

B. Benchmarks

To evaluate the performance overhead of the database, we use On-Line Transaction Processing (OLTP). OLTP systems are characterized by supporting multiple concurrent users executing transactions (e.g. select, update and delete operations). This kind of systems allows identifying typical operations of environments that need high availability of simultaneous access.

In order to analyze the performance of OLTP systems, we chose the TPC-C benchmark defined by Transaction Performance Council. TPC-C provides information regarding how many transactions are done in a interval time. There are 5 basic transactions that represent the behavior of an OLTP system.

The database benchmark tool used in the experiments was Hammerora [13], which is a free load generator and benchmark designed to perform stress tests in databases. It implements the TPC-C benchmark [14]. The number of virtual clients to be used and the quantity of transactions per virtual client are configurable parameters in the Hammerora tool.

C. Workloads

The tests were made running the TPC-C benchmark. The testing environment was setup to ensure that only the DBMS would be executed in each test case, besides ensuring that the OS would be always in the same initial status.

For the migration test, in every single case only one instance of the database or VM has been in use for each test case. We have chosen to run our tests using only one instance because running several concomitant, concurrent instances is useful for isolation or scalability testing, evaluated in Section IV-E5.

During migration, the volume of transactions has been determined by the execution of 10 data warehouse with 10, 50, 100 and 200 simultaneous connections. In every test the architecture resources were fully employed. In all tests the capabilities of the architecture were used in its entirety, i.e., considering that the tests have been executed with one instance, this instance employed 100% of the available resources, for example, processing and memory.

To evaluate the isolation we conducted two experiments. The first experiment evaluated the effects by decreasing the number of instance CPUs during the benchmark execution. Initially, the execution starts with 16 CPU cores and during the execution the number of CPU cores is decreasing to 8 (see Figure 8). The goal of this experiment was to evaluate the time needed to change the instance limit configuration during the benchmark execution and the impact on the database and application caused by this changing.

The second experiment of isolation verified if the Isolation really limit the CPU load. For this we performed benchmark executions, using four instance with 4 CPU cores and during the execution, the limit was changed from one of the instances to 16 CPU cores. The goal of this experiment was to compare the performance between the executions: before and after change CPU.

The test result of live migration is presented in the Section IV-E1 and isolation tests are shown in Section IV-E5.

D. Monitoring Tools and Metrics

Our evaluation of the Database Live Migration and Database Isolation involves monitoring of system resources during the execution. The purpose of this monitoring is to know the resource utilization pattern for these activities.

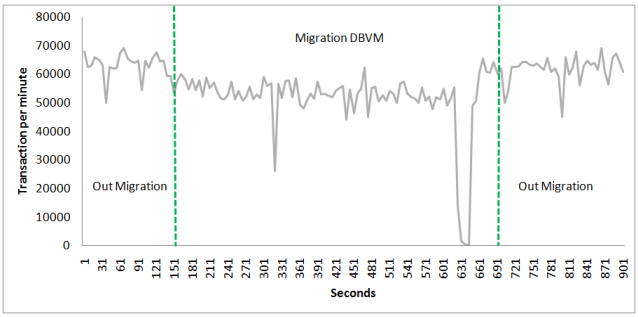
Although Unix tools are not database aware, they are easy to use and result in little overhead, and because this they were chosen. Two softwares were used to monitor the resources: *mpstat* and *dstat*. The first was used to monitor CPU usage and the second to monitor network traffic and memory usage. The commands *time* and *date* were used to monitor time and duration.

-In order to monitor the performance, in this case identified as TPM (Transactions Per Minute), we have used the Hammerora tool to extract information regarding database statistics.

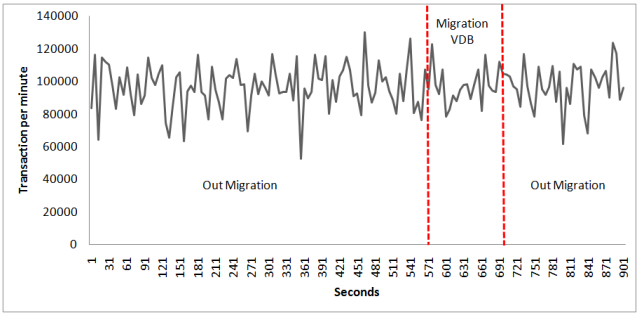
The following metrics were used: process completion time, CPU utilization, Memory utilization, Network traffic generated and Transaction per second.

E. Results

1) Performance Results During Live Migration: Our first evaluation is based on the TPC-C benchmark execution during the migration process. The VDB performance is 15% greater than the DBVM performance. Figure 4 represents a 10 warehouses charge with 200 concurrent connections.



(a) Analyse of Performance DBVM.



(b) Analyse of Performance VDB.

Figure 4. Analyse of performance during live migration

Another possible analysis is the time spent in the live migration process. The VDB spent an average 126 seconds (Figure 4(b)) whilst the DBVM spent an average 540 seconds (Figure 4(a)). This behavior is expected because database applications perform constant modifications in the server's memory pages. This implies the need to perform the copy of the modified memory pages several times during the process of VM migration. On the other hand, the VDB migrates only the session processes after the transaction execution is over.

It is also perceivable that during the VDB migration process the performance pattern kept stable, with no performance loss. On the other hand, the DBVM transactions stopped responding for a short period of 24 seconds, on average, something that has been observed in other works [11]. However, it didn't bring any downtime regarding the DBMS sessions.

This initial analysis allows us to say that the VDB executions kept the performance up during the live migration, which is not the case in the DBVM migration, shown in Figure 4. The resources analysis that follows is a further step towards our goal of evaluating the migration process.

2) CPU Utilization During Live Migration: We chose the *mpstat* tool in order to monitor the CPU utilization during the execution of the tests. The *mpstat* is computer command-line software used in Unix type operating systems to report processor related statistics. It is commonly used in computer monitoring in order to diagnose problems or to build statistics about computer CPU usage.

In accordance with other papers [6], the DBVM's CPU usage is as high as two percentual digits superior in comparison to that of the VDB. This behavior may be observed in the Figure 5.

It can be observed in Figure 5(a) that the VDB environment has an inferior CPU usage in comparison to the DBVM environment with the same volume of transactions - Figure 5(b). The DBVM environment, besides obtaining more processing, still keeps on the average of its usage. It is also possible to notice that the DBVM target host gets an increase in processing right from the start of the migration. This behavior cannot be observed in the VDB migration.

3) Network Utilization During Live Migration: The network is the resource that exhibits the biggest difference in usage, which can be observed in the Figure 6. It is noticeable that the VDB network, Figure 6(a), undergoes a small increase in usage during the whole migration period, showing no change in terms of consumption. An explanation for this behaviour might be, considering that the VDB migrates sessions only after the transaction execution is over, rendering unnecessary to copy data during the execution.

The DBVM environment (Figure 6(b)) undergoes a considerable increase of more than 100% in usage along the whole migration process. The data transference before and after the migration process is similar to the network usage rate of the VDB model, however, it is observed that the DBVM has an overload during this period due to the VM copy which has its own OS and DBMS installed, that are copied from a host to another. The volume of data transferred from a host to another is related to the size of the VM, i.e., the more resources are allocated to the VM, the longer the period is and the higher the data rate are during the migration.

4) Memory Utilization During Live Migration: The DBVM memory usage, in comparison to that of the VDB, is nine times more and it can be observed in Figure 7. This is due to the fact that the VM uses the whole memory the hypervisor made available for it.

Considering that the VM uses 100% of the available resources and that the guest host has 64GB of RAM, the VM occupies the 64Gb of available memory (Figure 7(b)) whilst the VDB (Figure 7(a)) has its memory used mainly by the RDBMS. We think that this is one of the main reasons for the high flow of data transfer and time spending, once smaller VMs demand less time and transfer rate.

Our experiments allow us to observe that the VDB migration is performed in a shorter period of time, with lower data flow and memory consumption in comparison to the same execution using the DBVM. We have also observed that a VDB does not lose performance during this period, and that the size of the database instance does not affect the observed parameters for the VDB. This evaluation complements the previous works which an evaluation of the DBVM migration was made.

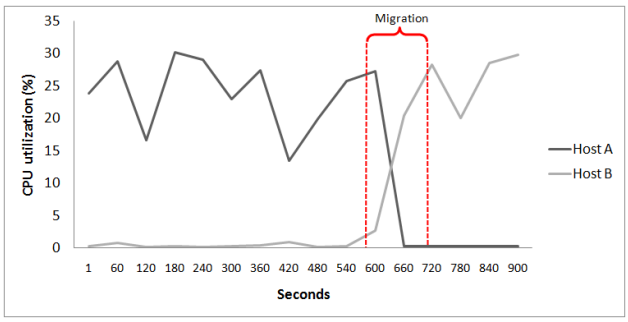
5) Results for Virtual Database Isolation: As mentioned earlier, this experiment aimed the performance effects by decreasing the number of CPUs during the benchmark execution and the impact on the database and application caused by this changing.

Figure 8 shows the behavior during the process of reconfiguration of resource Database for a given database instance, changing from 16 to 8 CPUs. In these graphics we can observe that the CPU usage during the execution are stable and high with 16 CPUs. When changed to 8, the CPU usage decrease generating a curve in the graph.

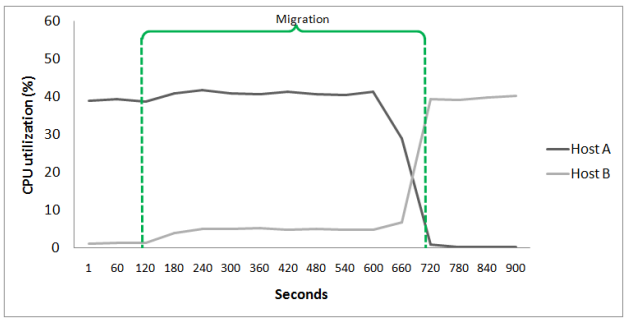
As can be seen in the Figure 8, when using 16 CPU, the CPU usage remains until 50% for most of the time. When changed to 8, the CPU usage decrease as expected for 25%.

The second experiment aimed to validate the isolation really limit the CPU load. To do the experiment, we compared the execution of it the benchmark four database using 4 CPU cores and during the benchmark execution change the CPU usage for 16 cores for a database.

Figure 9 shows the results of simultaneous execution for four database instances. Initially, all instances were configured with four CPU cores. We can consider that all the instances executed the same number of transactions (Figure 9(a)). During the execution, only Instance 04 had its CPUs quantity altered from 4 to 16. We can see that after the alteration, the transactions volume of instance 04

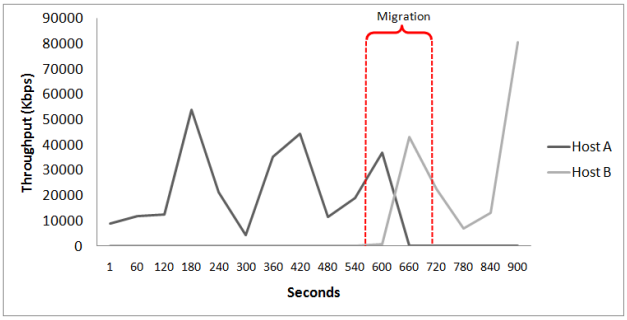


(a) CPU Utilization of VDB.

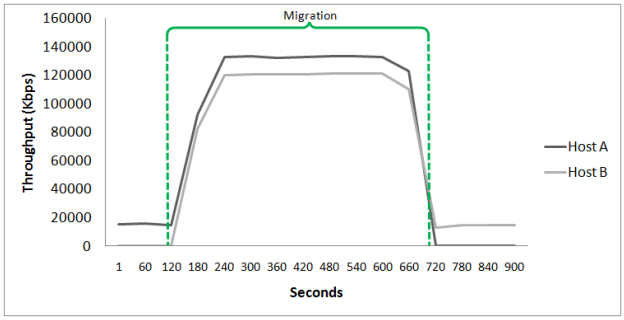


(b) CPU Utilization of DBVM.

Figure 5. Utilization of CPU during live migration

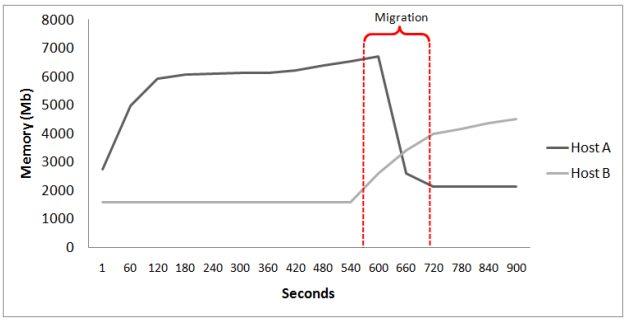


(a) Throughput network of VDB.

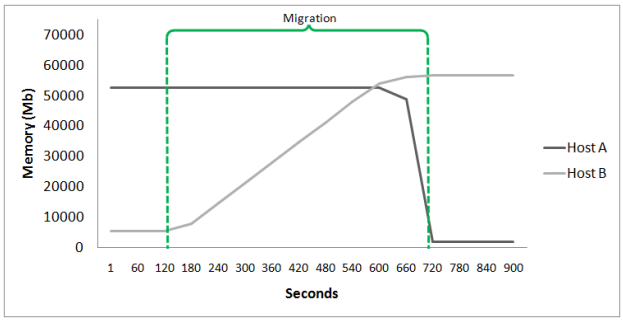


(b) Throughput network of DBVM.

Figure 6. Utilization of network during live migration



(a) Memory Utilization of VDB.



(b) Memory Utilization of DBVM.

Figure 7. Utilization of memory during live migration

increased from 43532 (Figure 9(a)) to 88382 (Figure 9(b)). With the resources increase, in this case CPUs, for instance 04 there was a higher efficiency without interfering in the other database instances, which continued with the same behavior (Figure 9(b)). Thus, we can suggest that through the isolation of one instance execution does not affect the others.

V. Related Works

The technology of virtualization adds an intermediate layer of software between the applications and the hardware. This layer is called hypervisor or virtual machine monitor (VMM) which maps the virtual resources visible to applications (in the context of this work, the DBMS) mediating the physical resources available to the host.

The virtualization can solve many critical problems regarding the database, such as usability, management, scalability, availability and implementation. Database applications may benefit from this type of infrastructure through some characteristics such as migration and consolidation that may be seen at [11] [15]. Then, depending on the demand of the database application, the environment can be adjusted dynamically providing more or less resources to the application.

Many papers analyze the virtualization layer overhead such as in [3] [4] that show a loss of performance between 7% and 11% if compared to application in a native system operation. Some papers like [2] [6] evaluate the virtualization layer overhead with database applications. Papers that evaluate a virtualized database environment security and isolation are also presented [12] [15]. In some high performance applications the gain of performance is superior to the overhead [16], what justifies its use in this type of environment.

Other papers also present an evaluation of the migration process of a virtual machine [10] [15] [17] [18]. The techniques presented

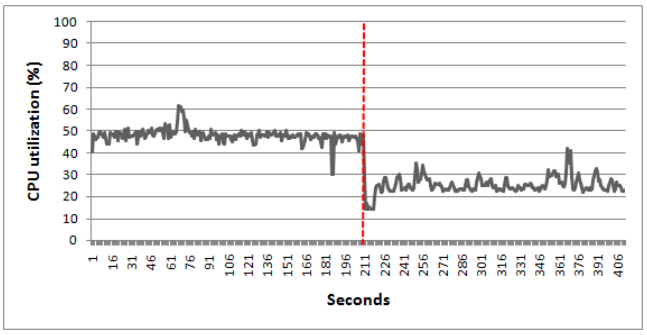
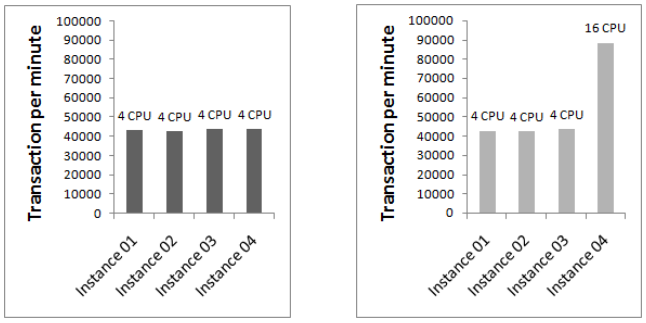


Figure 8. Results with CPU Decrease.



(a) Before Increase. (b) After Increase.

Figure 9. Results of Performance of TPM with CPU Increase.

by these papers allow us to do the migration of a VM with a minimum loss of performance. There are also papers that measure the cost of migration [11] and that present an evaluation of the performance of a DBMS on a VM during the live migration process [3].

Our paper complements other works because this one not only evaluates a virtual machine database migration costs, but it brings a new database virtualization proposal in this VDB context and evaluates the Isolation capacity that instances could offer when sharing physical resources. Also is evaluated the live migration capacities of a database instance without losing performance (TPM) and lower migration time and network data flow.

VI. Conclusion and Future Works

This paper presented an evaluation of the live migration process and performance isolation of virtualized databases, comparing databases over virtual machines (DBVM), the traditional approach, to a new less intrusive approach called virtual databases (VDB).

From the obtained results it was possible to conclude that VDB has 15% more transactions per minute than a DBVM during the live migration process, preserving its isolation capabilities. Furthermore, we can observe that CPU and network usage during the migration process of a VM with DBMS were greater than VDB due to the fact that the destination host is more active during the migration process, what will have an negative impact in the energy consumption. It was also possible to verify that the DBVM needs a longer time interval to perform the migration (including some downtimes) when compared to a VDB. These results indicate an overall greater complexity of the migration mechanism in the DBMS approach. These experiments successfully validated the isolation of VDB, since it was able to limit the CPU usage for a given database instance. We also conducted an experiment to monitor the utilization of resources when this limits were increased during execution.

As a future work, we are planning to develop a strategy to efficiently manage virtual databases consolidation taking host overload and migration costs into consideration.

References

[1] M. Ahmadi and D. Maleki, "Performance evaluation of server virtualization in data center applications," in *Telecommunications (IST), 2010 5th International Symposium on*, dec. 2010, pp. 638 –644.

[2] A. Aboulnaga, C. Amza, and K. Salem, "Virtualization and databases: state of the art and research challenges," in *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, ser. EDBT '08. New York, NY, USA: ACM, 2008, pp. 746–747.

[3] U. Minhas, J. Yadav, A. Aboulnaga, and K. Salem, "Database systems on virtual machines: How much do you lose?" in *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, april 2008, pp. 35–41.

[4] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, "Automatic virtual machine configuration for database workloads," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 953–966.

[5] Y. Wada, Y. Watanabe, K. Syoubu, J. Sawamoto, and T. Katoh, "Virtual database technology for distributed database," *Advanced Information Networking and Applications Workshops, International Conference on*, vol. 0, pp. 214–219, 2010.

[6] T. Lange, P. Cemim, F. Rossi, M. Xavier, R. Belle, T. Ferreto, and C. De Rose, "Performance evaluation of virtualization technologies for databases in hpc environments," in *Computer Systems (WSCAD-SSC), 2012 13th Symposium on*, 2012, pp. 88–94.

[7] "LXC Linux Containers," 2013, [Online; accessed 19-January-2013]. [Online]. Available: http://lxc.sourceforge.net

[8] "FreeBSD Jails," 2013, [Online; accessed 19-January-2013]. [Online]. Available: http://www.freebsd.org

[9] "Solaris Containers," 2013, [Online; accessed 19-January-2013]. [Online]. Available: http://www.oracle.com/technetwork/server-storage/solaris/containers-169727.html

[10] C. C. Keir, C. Clark, K. Fraser, S. H, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI*, 2005, pp. 273–286.

[11] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *CloudCom*, 2009, pp. 254–265.

[12] F. Almari, P. Zavarsky, R. Ruhl, D. Lindskog, and A. Aljaedi, "Performance analysis of oracle database in virtual environments." in *AINA Workshops*, L. Barolli, T. Enokido, F. Xhafa, and M. Takizawa, Eds. IEEE, 2012, pp. 1238–1245.

[13] "Hammerora," 2013, [Online; accessed 19-January-2013]. [Online]. Available: http://hammerora.sourceforge.net

[14] "Tpc consortion," 2013, [Online; accessed 19-January-2013]. [Online]. Available: http://www.tpc.org/tpcc/detail.asp

[15] P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu, and H. Hacigumus, "Intelligent management of virtualized resources for database systems in cloud environment," in *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ser. ICDE '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 87–98. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2011.5767928

[16] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A case for high performance computing with virtual machines," in *Proceedings of the 20th annual international conference on Supercomputing*, ser. ICS '06. New York, NY, USA: ACM, 2006, pp. 125–134.

[17] S. Das, S. Nishimura, D. Agrawal, and A. E. Abbadi, "Live database migration for elasticity in a multitenant database for cloud platforms," UCSB CS, Tech. Rep. 2010-09, 06/2010 2010.

[18] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *Proceedings of the 20th international symposium on High performance distributed computing*, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 171–182. [Online]. Available: http://doi.acm.org/10.1145/1996130.1996154