The Cost of Migrating DBMS from a conventional environment to Virtual Machines

MOHAMED HELMY KHAFAGY
Fayoum University
EGYPT
mhk00@Fayoum.edu.eg

AMR AHMAD SALEM
University of Nottingham
UNITED KINGDOM
Amr.Salem.09@gmail.com

Abstract:- Virtualization becomes a very common solution in enterprise and small systems. Therefore, virtualization decreases financial costs and decreases system administration efforts. Also virtualization has been proven to be one of the most efficient approaches to implementing highly available systems.

One of the most common software systems deployed in virtualized environments is Database Systems. The key reasons behind these migrations are the great flexibility regarding database administration tasks and also the high availability that virtualized systems offer, which is a very critical factor for some database systems, so it is important to understand the cost of migration from a conventional environment to a virtualized one.

In this paper we present an experimental study of the overhead of DBMS migration from a conventional environment to a virtualized one, we use TPC-H Benchmark to calculate this overhead. We show that the average overhead can be around 7% for normal DBMS operations and 97% for fetching data directly from the external disks and we also present details of the different causes of this overhead. Our study shows that for normal DBMS operations the benefits of virtualization come at an acceptable cost

Key-Words: Database-Virtualization-Performance-Virtual Machine- Xen-Oracle

1. INTRODUCTION

Virtualization is simply hiding the physical resources (e.g. CPU, storage and memory) of a machine by implementing a special software layer on top of them [1]. This layer translates the machine's physical resources into virtual ones. We can then use these virtual resources to create multiple Virtual Machines (VMs), isolated from each other. Virtualization solutions are very appealing nowadays to implement many systems including database systems. Using virtualization, a single powerful enterprise server can host multiple VMs to do the job of multiple physical machines; this can be referred to as Server Consolidation. Server Consolidation solutions, obviously, decrease the initial costs and the running costs of relatively big systems. Moreover, Server Consolidation minimizes administration and maintenance efforts of multiple systems by merging them into one physical machine. Nowadays, most of the major players in the IT industry are participating in the virtualization scene [2]. Big name like VMware [3] and Citrix Systems [4] mainly work in virtualization related products. Also operating systems vendors like Redhat [5] and Ubuntu [6] have already integrated virtualization solutions in their operating systems.

Virtualization can be efficiently used to implement high-availability Database Management systems [7]. Also the nature of any VM allows for greater flexibility in manageability issues (e.g. Duplication or backup). Usually, any large enterprise or service provider will use a number of large databases that are always growing in size; therefore, having a DBMS for such business installed within a VM is very convenient due to the reasons discusses before. For instance, Server Consolidation will significantly decrease the number of machines used; and consequently, decrease financial costs, power consumption and maintenance efforts. Also reallocating resources among multiple VMs sharing the same physical server can be a very simple task, which means that it can be a dynamic process depending on each DBMS workload requirements. Moreover, due to the software nature of a VM, backup and recovery operations, which are essential for any database, can be much easier than conventional database backup and recovery, as any VM can be viewed as a single image file. Also for the same previous reason, a single VM, alongside with its DBMS, can be copied and duplicated in any other site in a very simple process.

Virtualization is becoming a very common trend in industry especially in Database Management Systems. Virtualization incurs an abstraction layer on top of physical hardware, which makes it necessary for the guest VMs to communicate with this layer first in order to access the physical hardware. This layer means an extra overhead and performance degradation. We can even assume it can be a serious issue if multiple VMs are trying to access the same resource simultaneously.

Our Goal in this Paper is to know the exact cost of such migration before implementation and what are the main reasons causing this overhead.

To know what exactly the cost and its reason we compare the performance of a DBMS with a certain workload in a conventional environment to the performance of the same DBMS with the same workload running within a VM; to find out how much do we lose by running a DBMS on a VM using TPC-H [8] benchmark as our database workload on Oracle DBMS [9] under the operating system Red Hat Enterprise Linux Appling on Xen virtualization hypervisor [10]. We show that the average overhead can be around 7% for normal DBMS operations and 97% for fetching data directly from the external disks and we report details on the nature and causes of this overhead. We view this as an encouraging result, since it means that for normal DBMS operations the benefits of virtualization come at an acceptable cost

The rest of this paper is organized as follows. In Section 2 we present an overview of related work. Section 3 describes our Test Environment, Section 4 reports our experimental results and Section 5 concludes.

2. RELATED WORK

Diwaker Gupta et al. [11] introduce a Xen hypervisor as an x86 open source virtualization solution. It was primarily designed to host up to one hundred virtual machines simultaneously. He used six different benchmarks to prove that the performance of a Xen virtual system is very close to the performance of a conventional system. They also compared Xen to other virtualization solutions such as User Mode Linux and VMWare Workstation, and proved that Xen's virtualization overhead is much less than others'.

After Xen was introduced in 2003, it started to become a very common choice among researchers to carry out their virtualization experiments. In [1], it is proven that virtualization increases availability to a great extent. by carry out live migration experiments for some Xen based systems, such as web servers and online game servers, and show that the downtimes for the virtualized systems are very low.

Also in the area of resources management, Padala et al. [12] developed an automatic resources allocation tool. This tool is based on the classical control theory; it monitors the performance of some Xen virtual machines, hosted on one physical system, and then allocates the resources accordingly.

Some researchers were only interested in virtual machines monitoring, such as in[13] by introduce XenMon, a performance monitoring tool designed for Xen-based systems. XenMon mainly focuses on applications with intensive I/O.

Also Padala et al. [14] evaluate Xen-based systems and OpenVZ-based systems [15]. The experiments show that when quadrupling the workload of a virtualized system, the response time of a Xen-based system can increase by 400% while the response time of an OpenVZ can increase by only 100%. We could only find very few papers handling the problem of database management systems performance in virtual machines, In the work of [16], the authors define the problem of virtualization design. They define virtualization design as the problem of statically allocation of resources to multiple virtual machines running on the same physical system, while each virtual machine runs a database workload. They consider the problem of virtualization design as an extension to the conventional database physical design problem. In order to solve this problem, they suggested a cost modeling approach to mathematically represent the problem.

3. TEST ENVIRONMENT

We use Two Dell Optiplex 760 machines both have Intel Core 2 Duo E8500 processor with two cores, 6 megabytes cache memory, 3.16 gigahertz clock speed and 64 bit instruction set and 8 gigabytes of physical memory. We refer to one of these machines as SysA and to the other as SysB. It is operated by Red Hat Enterprise Linux 5.4 (RHEL), and it has Oracle 10g Release 2 installed. SysB has two domains, Dom0 and DomU; Dom0 is the privileged VM which is used to control and configure the Xen hypervisor, and DomU is used to run the Oracle DBMS and the database workload. Both VMs are operated by RHEL 5.4.

TPC-H benchmark version 2.10 is used to generate the workload, with scale factor 1 (i.e., 1GB). We use the 22 queries of the benchmark. This benchmark implementation is optimized for Oracle 10g Oracle was identically configured for the Base and Xen systems. The Oracle client and server are both run on the same machine [17], the client adds a negligible overhead to the machine, consuming well below 1% of the CPU and very little memory.

Our experiments can be categorized into two main categories, Warm Experiments and Cold Experiments. In Warm Experiments we run each query at least once before conducting our actual test, to make sure that the data we fetch from our database is already cached in the system memory. On the contrary, in Cold Experiments, we clear the system memory from any cached data before conducting any tests, to make sure that Oracle fetches required data directly from disk.

4. EXPERIMENTS RESULTS

4.1 Warm Experiments

In this section we discuss the results for all of our Warm Experiments. As mentioned before, we make sure that our queries are already cached in memory before conducting the actual experiments to eliminate the I/O waiting factor. This approach gives us the chance to investigate the CPU and memory performance without having the high I/O overhead.

We compare the average runtime for each of the TPC-H 22 queries in both systems, SysA and DomU. **Error! Reference source not found.** presents the measured average runtimes. Each runtime is the average runtime of five consecutive runs. It also shows the Overhead in seconds and in percentage.

The following equations are used to measure overhead:

$$Overhead\ (in\ seconds) = DomU\ Runtime - SysA\ Runtime$$

$$Overhead\ (in\ percentage) = \frac{Overhead\ (in\ seconds)}{SysA\ Runtime} \times 100$$

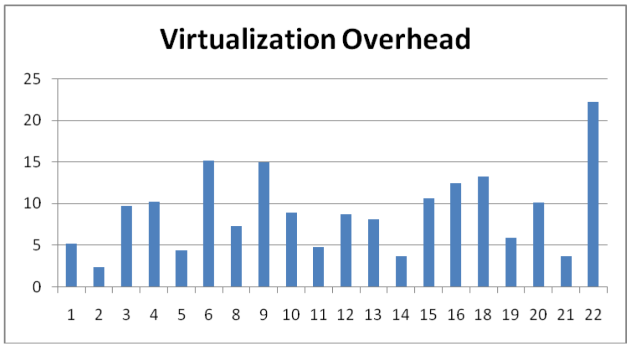

Figure 1 Virtualization Overhead

Figure 1 show that the average Overhead about 7% this can be an answer for our first research question. In the following sections, we will have more experiments to find out what are the main reasons behind this overhead

4.1.1 User and System Times

User time is the time at which any process spends in the user mood within the CPU, and system time (sometimes called kernel time) is the time at which any process spends in the system mode executing system privileged tasks within the CPU.

Our first approach to understand the reasons behind performance degradation in Virtual Machines is measuring the time spent by CPU either in user mode or kernel mode while serving each query. We assume the following for Warm Experiments:

$$Query\ Run\ Time = CPU\ User\ Time + CPU\ System\ Time$$

The previous assumption is true for most of the cases. However, since we eliminated I/O waiting time, CPU rarely spends time serving interrupts (IRQ time). We discarded runs with IRQ times and recorded our measurements according to the previous assumption. We use mpstat [18] tool to collect CPU measurements [19]. We run the tool for a certain interval, and make sure our query is run within this interval.

Figure 2 shows the Overhead in user and system time for the 8 selected queries. We can see that both the user time and the system time of almost all queries cost overhead in DomU compared to SysA. However, the slowdown in user time is small compared to the slowdown in system time. This is expected since virtualization adds overhead to system level operations and does not affect user level operations. So we focus next on where does the slowdown in system time come from? For these queries, system time is attributable to either system calls or page fault handling. We look into these two components next. In the interest of space, we only present results for the 8 queries whose overhead more than 10%.

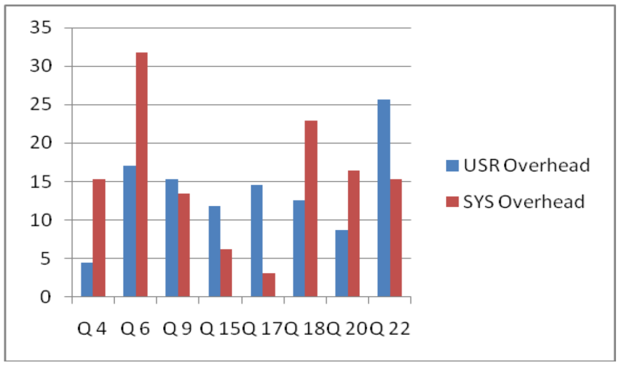

Figure 2. Overhead: User vs. system time

4.1.2 System Calls

In a Xen virtualized system, some system calls will go directly from the OS to the CPU to be executed and some will have to go through the Xen hypervisor layer first. However, this is only true for para-virtualization; in full virtualization, all the system calls must go through the hypervisor layer first. This is one of the advantages of para-virtualization and also one of the reasons for para-virtualization better.

In this section we have a deep insight for Oracle server process system calls for the 8 selected queries in both systems. We also use Strace [18] tool to collect detailed information about the server process and its system calls. Strace gives the required information only after the process under investigation terminates.

Table presents summarized information for system calls. We use the following equation to define the System Calls Time Overhead:

$$\text{System Calls Time Overhead} = \frac{\text{DomU SysCalls Time} - \text{SysA SysCalls Time}}{\text{SysA System Calls Time}} \times 100$$

In Table 2 we show that system calls time is always a small fraction of the total system time as expected, we can notice quite relative large overheads for all of the queries. However, queries with less system calls count suffer from larger relative overhead, which means there is a dominant factor affecting system calls time in DomU

4.1.3 Page Fault Handling

There are two kinds of page faults; Major Page Faults and Minor Page Faults. A Major Page Fault is the exception generated by the hardware when an application tries to access a memory page that is not loaded into physical memory. And a Minor Page Fault is what generated when the application tries to access a memory page that resides in the physical memory, but not assigned to the application [20] Since our Warm Experiments previously discussed procedures ensure that all of our queries' data resides in the physical memory, we only consider Minor Page Faults in this section. Page Fault Handling can be a very expensive factor for applications performances. We carry out this experiment to investigate how the Oracle DBMS handles page faults with a TPC-H workload.

Error! Reference source not found.3 presents the count of minor page faults generated per second by Oracle's server process when running each of the TPC-H queries. We include the overhead presented in **Error! Reference source not found.**1 trying to relate virtualization overhead to minor page faults rate.

Table 2: Summarized System Calls Information

	SysA			DomU			Overhead
	SysCalls Count	**SysCalls Time**	**System Time**	**SysCalls Count**	**SysCalls Time**	**System Time**	
Q4	4151	47.341	258	11893	195.35	340	312.64
Q6	2602	37.915	187	7559	190.71	210	403.01
Q9	6735	49.37	379	17355	217.90	430	341.37
Q15	6981	55.776	358	18192	200.85	380	260.11
Q17	2938	44.425	194	8329	199.63	200	349.36

Q18	3669	45.016	236	9133	214.269	290	375.98
Q20	5337	45.691	275	13396	213.627	320	367.55
Q22	94	10.997	9	154	72.004	0	554.76

Table 0: Minor Page Faults Rate

	SysA	**DomU**	**Overhead**
Q4	53	52	10.26
Q6	0	0	15.15
Q9	1267.53	1244.34	15
Q15	45.93	61.76	10.68
Q17	10.8	10.53	12.5
Q18	1304.6	1303.86	13.26
Q20	974.33	975.07	10.17
Q22	11.47	12	22.22

4.2 Cold Experiments

In the previous section, Warm Experiments, we studied the CPU and memory performances. In this section, Cold Experiments, we focus on the disk performance. We study the TPC-H 22 queries runtimes under cold conditions, we study the disk I/O waiting time for each query and we finally study data reading rate in both systems.

4.2.1 Virtualization Overhead

In this section, we follow exactly the same procedures as in Warm Experiments Virtualization Overhead section. Figure 3 represents the results for our experiment and shows that the average Overhead about 97%. We can notice the large overhead for almost all of the queries, which indicates the high cost of virtualization in a cold system. Also In the interest of space, we only present results for the 9 queries whose Overhead more than 115%.

4.2.2 I/O Wait Time

As previously discussed in Warm Experiments, here we also use mpstat [18] tool to collect CPU usage information while running each of selected 9 queries. For SysA, we assume the following:

$$Query\ Run\ Time = CPU\ User\ Time + CPU\ System\ Time + I/O\ Wait\ Time$$

Where I/O wait time is time spent by the CPU waiting for the required data to be fetched from the physical disk. The previous assumption is proven to be quite accurate for SysA runs. However, in DomU, a new factor appears; which is steal time. Steal time is the time spent by the CPU of a VM waiting while the hypervisor is serving another CPU of another VM [18]. Accordingly, for DomU, our assumption becomes as follows:

$$Query\ Run\ Time = CPU\ User\ Time + CPU\ System\ Time + I/O\ Wait\ Time + CPU\ Steal\ Time$$

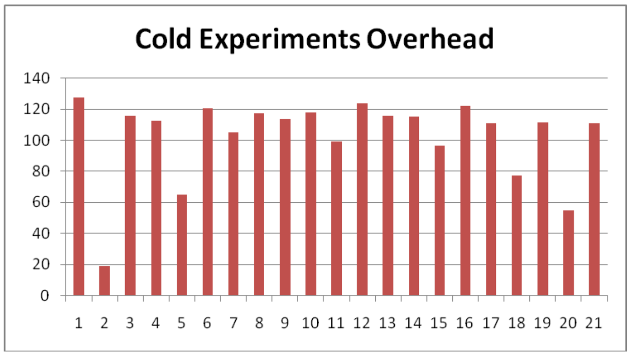

Figure 3: Cold Experiments Overhead

In this section we are concerned with studying the effect of the I/O waiting time on the overall performance. Table 0 shows the significant change in the CPU usage pattern between SysA and DomU. I/O time is dominant for all selected queries in SysA This means that I/O wait time is the major factor causing the poor performance of Oracle in a virtualized cold system.

Error! Reference source not found. represents detailed information about I/O wait times for the selected queries. All of the queries suffer from extremely high I/O time overhead.

Table 0: User, System and I/O Times Relative to Total

	SysA			**DomU**			**DomU**
	User	**System**	**IO**	**User**	**System**	**IO**	**Steal**
	(%)	(%)	(%)	(%)	(%)	(%)	(%)
Q 1	56	9.7	34.3	20.3	2.41	75.9	1.34
Q 3	12.5	9.88	77.6	1	0.41	98.1	0.48
Q 6	8.17	8.38	83.4	0.56	0.43	98.6	0.39
Q 8	11.6	9.96	78.3	0.94	0.36	98.3	0.36
Q 10	12.7	9.74	77.5	1.23	0.52	97.8	0.43
Q 12	11.5	10.74	77.6	0.95	0.63	97.9	0.47
Q 13	49.6	7.99	42.3	21.0	3.8	73.8	1.26
Q 14	9.71	11.64	78.6	0.55	0.42	98.7	0.29
Q 17	11.2	10.89	77.8	0.74	0.35	98.5	0.39

Table 6: I/O Wait Times

	SysA	DomU	Overhea d	Overhead
	(secs)	(secs)	(secs)	(%)
Q 1	3.65	18.38	14.73	403.56
Q 3	11.26	30.75	19.49	173.09
Q 6	8.9	23.22	14.32	160.9
Q 8	11.63	31.73	20.1	172.83
Q 10	11.48	31.52	20.04	174.56
Q 12	10.45	29.5	19.05	182.3
Q 13	1.41	5.3	3.89	275.89
Q 14	8.93	24.17	15.24	170.66
Q 17	8.87	24.94	16.07	181.17

5. CONCLUSIONS

Determining virtualization overhead for an Oracle DBMS can be straight forward. However, determining the reasons behind this overhead may not be that easy. The point of this paper is not to suggest using virtualization to implement Database. But also it means that virtualization does not cost much even for the cold case. By Using a TPC-H workload running on Oracle in a Xen virtual machine environment we show that the cost average is about 7% in cold case for system calls, page fault handling (normal DBMS operations). Whenever in the warm case the average cost about 97% for I/O operations, these numbers are for a simple system that is not highly optimized. Optimizations such as using Index and using raw disk in Dom0 for the DomU virtual disk can improve performance, so this overhead can be viewed as a worst case overhead that can likely be further improved.

Our hope is that these findings will encourage further research in the area of virtualization and self-managing database systems.

6. REFERENCES

[1] Christopher Clark,.. Live Migration of Virtual Machines .The 2nd Conference on Symposium on Networked Systems Design and Implementation., Vol. 2, 2005,pp. 273-286.

[2] Chen, Gary. Worldwide. Virtual Machine Software. Vendor Shares 2009..

[3] VMware. [Online] http://www.vmware.com/.

[4] Citrix Systems. [Online] http://www.citrix.com.

[5] Redhat Virtualization. [Online] http://www.redhat.com/virtualization.

[6] Ubuntu Virtualization. [Online] http://www.ubuntu.com/server/virtualisation.

[7] Olofson, Carl W. World Wide Relational Database Management Systems .Vendor Shares. International Data Corporation (IDC). 2007.

Transaction Processing Performance Council. http://www.tpc.org/.2010.

[8] Chip Dawes, Bob Bryla, Joseph C. Johnson, Matthew Weishan. OCA: Oracle 10g Administration I. sol.: Sybex. 2004.

[9] Xen. [Online] http://www.xen.org/.

[10] Diwaker Gupta, Ludmila Cherkasova, Rob Gardner, Amin Vahdat. Enforcing Performance Isolation Across Virtual Machines in Xen.. The ACM/IFIP/USENIX International Conference on Middleware. 2006. pp. 342-362.

[11] Pradeep Padala, Kang G. Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, Kenneth Salem. Adaptive Control of Virtualized Resources in Utility Computing Environments.. The 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems. 2007.

[12] Diwaker Gupta, Rob Gardner, Ludmila Cherkasova. XenMon: QoS Monitoring and Performance Profiling Tool. HP Laboratories Palo Alto. 2005.

[13] Pradeep Padala, Xiaoyun Zhu, Zhikui Wang, Sharad Singhal, Kang G. Shin. Performance Evaluation of Virtualization Technologies for Server Consolidation. HP Laboratories Palo Alto. 2007.

[14] OpenVZ Wiki. [Online] http://wiki.openvz.org/.

[15] Ahmed A. Soror, Ashraf Aboulnaga, Kenneth Salem. Database Virtualization: A New Frontier for Database Tuning and Physical Design.. The IEEE 23rd International Conference on Data Engineering Workshop. 2007. pp. 388-394.

[16] Lorentz, Diana. Oracle Database SQL Language Reference. s.l.: Oracle. 2008.

[17] Linux Man Pages. [Online] http://linuxmanpages.com/.

[18] Daniel P. Bovet, Marco Cesati. Understanding the Linux Kernel. s.l.: O'REILLY. 2003.

[19] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne. Operating System Concepts. 8th edition. 2010.