Cloud Computing
An Evaluation of Rules of Thumb for Tuning RDBMSs

Tarcizio Alexandre Bini, Marcos Sfair Sunye and Adriano Lange
Federal University of Paraná, Curitiba, PR, Brazil

Keywords: Virtualization, Cloud Computing, Legacy Database, Tuning.

Abstract: Cloud computing environments are attractive for IT service provision as they allow for greater flexibility and rationalization of IT infrastructure. In an attempt to benefit from these environments, IT professionals are incorporating legacy Relational Database Management Systems (RDBMSs) in them. However, the design of these legacy systems do not account to the changes in resource availability, present in cloud environments. This work evaluates the use of rules of thumb in RDBMS configuration. Through an evaluation method that simulates concurrent I/O workloads, we analyzed the RDBMS performance under various settings. The results show that well-known configuration rules are inefficient in these environments and that new definitions are necessary to harvest the benefits of cloud computing environments.

1 INTRODUCTION

Over the past few years, cloud computing environment has become attractive for providing IT services (Buyya et al., 2009). The massive use of *virtualization* (Smith and Nair, 2005) in such environment has created not only a flexible and simple way to manage computing resources, but has also reduced IT costs considerably. As a consequence, it became desirable to incorporate legacy systems in cloud environments.

Most legacy RDBMS systems use a client-server architecture, in which a client program has access to a RDBMS server through a local network. A simple strategy to integrate such systems in a cloud environment is to host the client program and the RDBMS server in different Virtual Machines (VM).

Despite virtualization's high administrative and economic benefits for legacy systems maintenance, dynamic resource provisioning on cloud computing environments boosts the performance setup and adjustment problem of RDBMS: performance maximization and resource usage minimization. In other words, RDBMS configuration parameters do not consider resources' availability over time, which may vary due to concurrent workloads led by other VMs on the same hardware.

The efficient adjustment of RDBMS's performance in cloud computing environments must account for existing workloads from other VMs as they compete for the same physical resources, among them the disk units to which the access represents the greatest bottleneck of RDBMs systems that deal with large amounts of data (Hsu et al., 2001). Taking disk units as mechanical devices, I/O workloads can be characterized into two dimensions: (1) *reading* and *writing* operations, (2) *random* and *sequential* access (Delimitrou et al., 2012).

This paper puts into discussion the application of RDBMS tuning techniques, and it questions the employment of rules of thumb on virtualized environments. The paper assesses through analytical methodology different tuning rules against concurrent I/O workloads characterized by the dimensions above. The results reveal the following about RDBMS on virtualized environments: (1) rules of thumb can be inefficient or even harm performance dramatically; (2) performance improvements can be obtained using not recommended configuration parameters; (3) situations in which any attempt to optimize performance by tuning is useless.

The paper is structured as follow: Section 2 discusses related work. Section 3 describes the methodology used to simulate the virtualized environment, the I/O workloads, and the database workloads. Section 4 presents and discusses experimental results. Finally, Section 5 concludes and presents future work.

Bini T., Sunye M. and Lange A..
Cloud Computing - An Evaluation of Rules of Thumb for Tuning RDBMSs.
DOI: 10.5220/0004882601870192
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 187-192
ISBN: 978-989-758-027-7

2 RELATED WORK

Tuning of RDBMS systems is challenging due to the high combinatorial number of configuration parameters. Therefore, some researchers propose approaches to aid or even automate this configuration process (Debnath et al., 2008; Storm et al., 2006; Tran et al., 2008). For instance, the Ituned tool uses sampling methods at runtime to tune the configuration parameters of RDBMS (Duan et al., 2009). It selects parameter values which produce the higher improvements on performance and the least possible overhead on the workload.

In virtualized environments, there is a concern regarding the rationalization of physical resources among all the VMs running on the same host. Soror *et al.* (Soror et al., 2008) proposes a configuration assistant to determine the amount of memory and processing time to each VM. This assistant uses a greedy algorithm that on each stage increases or reduces the VM's resources and calculates the workloads' costs using the RDBMS cost model. In this process, RDBMS tuning configuration parameters are defined according the quantity of resources assigned to the corresponding VM. Other researchers suggest approaches for resource allocation among VMs and the RDBMS in them (Soror et al., 2007; Cecchet et al., 2011; Xiong, 2012).

Regarding VM's resource allocation, Rao *et al.* (Rao et al., 2009) presents a dynamic approach which employs learning techniques to adjust the resources of each VM, on-the-fly.

While developing this work, none of the research regarding RDBM's tuning in virtualized environments, considers the influence of concurrent I/O workloads characterized by the dimensions *read/write* and *sequential/random*. Such influence may largely affect the quality of service.

3 METHODOLOGY

We employ a base methodology to examine the RDBM's performance against distinct tuning rules. In this context, a RDBM running on a VM competes for physical resources with other processes running on other VMs, which perform properly characterized disk access.

Environment: We conducted experiments on a machine with the following specification: 3.10 GHz Intel Core I5 (3350P), 6 MB of L2 cache, 8 GB of RAM, and two 512 GB SATA disks. The operating system is a GNU/Linux, kernel 2.6.38X86-64, with qemu-kvm version 0.14, and PostgreSQL (Pos, 2013) database system version 9.2.0 running on VMs with the same version of GNU/Linux kernel. All the VMs have the same resources: one virtual CPU, 512 MB of RAM, and 50 GB disk.

Database Configuration Parameters: A typical RDBMS has several configuration parameters where proper values have a high impact on its performance. The evaluation of all possible values of the configurations parameters would require an exponential number of experiments. Therefore, we consider the work of Debnath *et al.* (Debnath et al., 2008) to mitigate this problem. Their work establishes a rank of the parameters that exert greater influence on tuning optimization, from which we take the three higher ranked parameters for the PostgreSQL RDBMS, detailed on Table 1. The minimum and maximum values follow the recommendations on the PostgreSQL documentation (Pos, 2013) and tuning references (Smith, 2010).

The experiments use the following values and percentages for the database configuration parameters:

- **shared_buffers:** 2,5% (*Min*), 5% (*Default*), 40% (*RuleofThumb*), 70% (*Max*) of VM total RAM;
- **effective_cache_size:** 10% (*Min*), 25% (*Default*), 60% (*RuleofThumb*), 90% (*Max*) of VM total RAM;
- **work_mem:** 300 kb (*Min*), 1 MB (*Default/RuleofThumb*), 3MB (*Max*). As the default *work_mem* value is in the range of the rules of thumb, this parameter uses only three values.

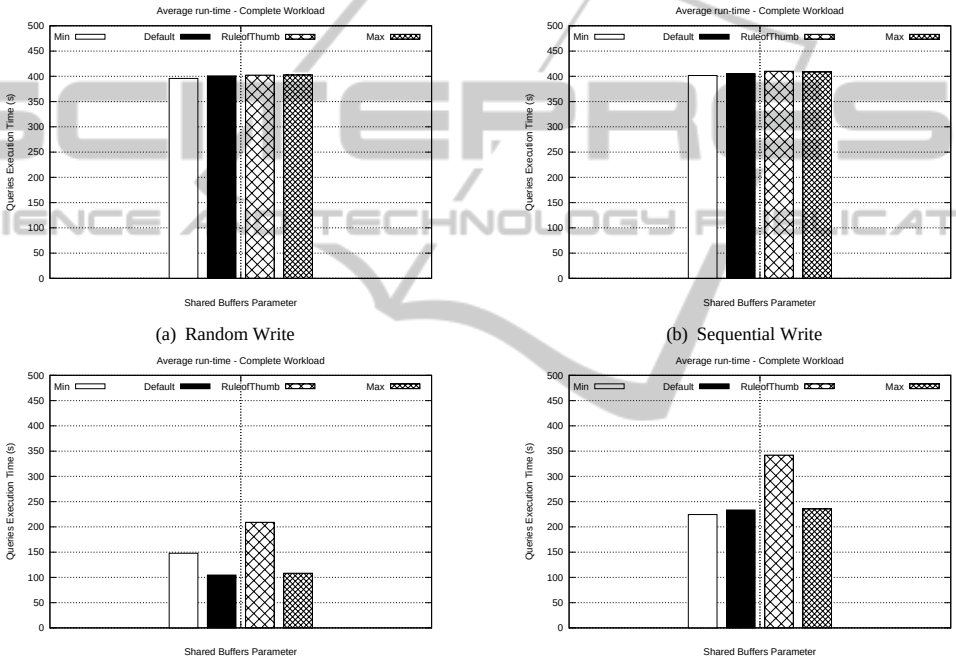
Database Workload: We used the OSDL DBT3 (dbt, 2013), an implementation of the TPC-H benchmark (Council, 2013), tuned for PostgreSQL, with a scale factor of 10 (10 GB). The size of the database with the indexes is 17 GB. The TPC-H benchmark provides a set of 22 SQL queries which we adjusted because we are interested in I/O workloads, resulting in a set of 18 SQL queries[1]. For testing, we randomly submitted each query five times to the RDBMS with each of the three parameters and their respective values, from which we computed the average of the results. We used a warm database cache for the test.

Concurrent I/O Workloads: For our experiments, we modified the bonnie++ benchmark (bon, 2013) version 2.3 to generate four types of concurrent I/O workloads: *random-read*, *random-write*, *sequential-read*, and *sequential-write*. For each workload, the modified bonnie++ repeatedly requested a synchronous 4KB-sized read/write operation from/to a file stored on the VM's virtual disk. The file size was bigger than the VM's cache memory.

[1]https://github.com/tarciziobini/QueriesPaperICEIS

Table 1: PostgreSQL configuration parameters used in our experiments.

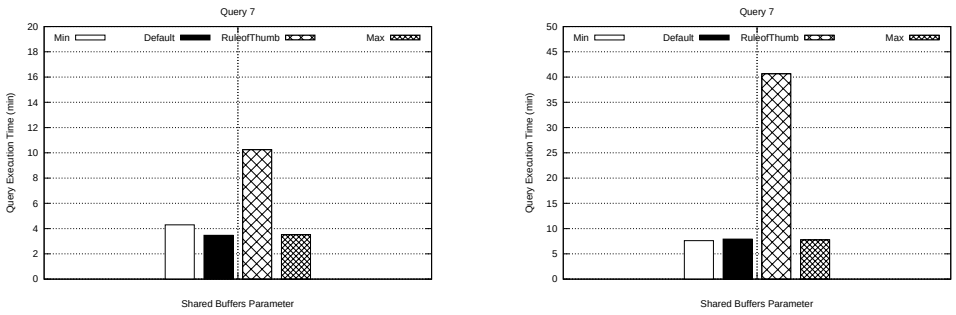
Parameter	Description	Default Value	Min. Value	Max. Value
work_mem	Amount of memory to be used by each sorting and hashing operator. Consider *max_connections* parameter.	1 MB	Total RAM / max_connections / 16	Total RAM / max_connections / 4
shared_buffers	Controls the size of the block in memory, for storing data to be written or already read in the database.	24 MB	25 % of total RAM memory	50 % of total RAM.
effective_cache_size	The amount of RAM memory used to effective cache of the database.	128 MB	50 % of total RAM memory.	75 % of total RAM.



(a) Random Write (b) Sequential Write

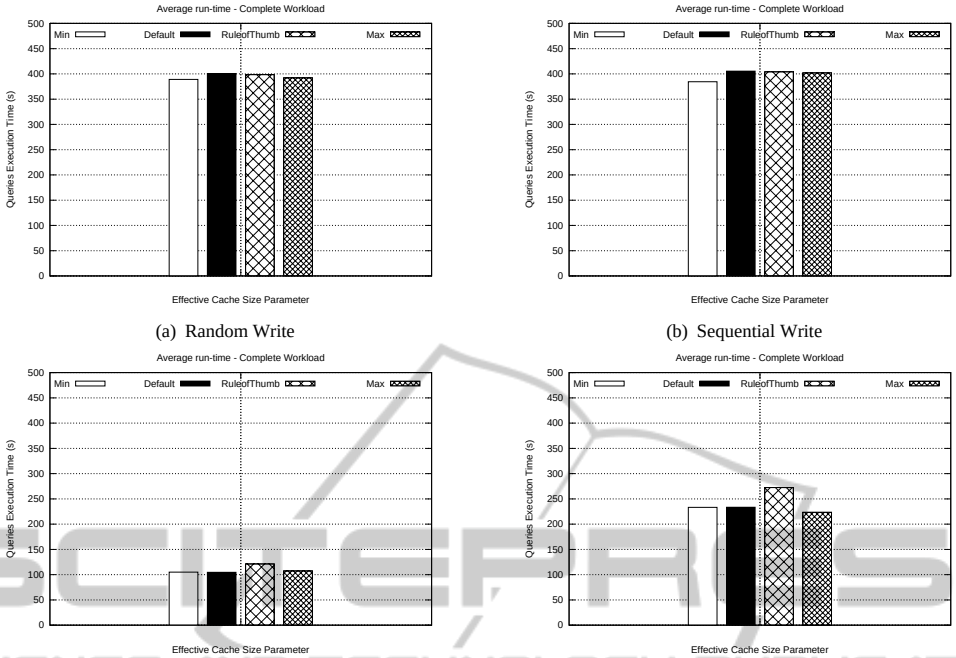
(c) Random Read (d) Sequential Read

Figure 1: SQL workload execution average time considering I/O workload and tuning in *shared_buffers* parameter.



(a) Random Read (b) Sequential Read

Figure 2: Query 7 Execution average time considering I/O workload and tuning in *shared_buffers* parameter.



(a) Random Write

(b) Sequential Write

(c) Random Read

(d) Sequential Read

Figure 3: SQL workload execution average time considering I/O workload and tuning in *effective_cache_size* parameter.

4 EXPERIMENTS

This section presents and discuss the results from the experiments.

4.1 Shared_Buffers

Figure 1 shows the average execution time in seconds from the workload of the 18 SQL queries, running on a VM. Each graphic indicates a particular concurrent I/O workload from another VM which competes for physical resources. Figures 1(c) and 1(d) show that the rules *RuleofThumb* for *Random Read* and *Sequential Read* have underperformed the RDBMS's *Default* rule and *Min* rule, respectively, by more than 100 seconds. On the other hand, there are no significant performance gains by tuning the configuration parameters neither for *Random Write* nor *Sequential Write* workloads as shown on figures 1(a) and 1(b).

Figure 2 shows that by following the rules of thumb the system's performance might degrade drastically. In the case of the modified Query 7 from the TPC-H benchmark, the *RuleofThumb* for *Random Read* and *Sequential Read* have had underperformed the best result by a factor of 3 and 5 times, respectively.

4.2 Effective_Cache_Size

The tuning of the *effective cache size* parameter has reached best performance results with the rules *Max* e *Min* for the *Read* and *Write* workloads, respectively as shown on Figures 3(d), 3(a), and 3(b).

Figure 4 shows the execution time of the Query 3 where the Min rule has outperformed both the *Default* rule and *RuleOfThumb* rule for the *Sequential Write* operation.

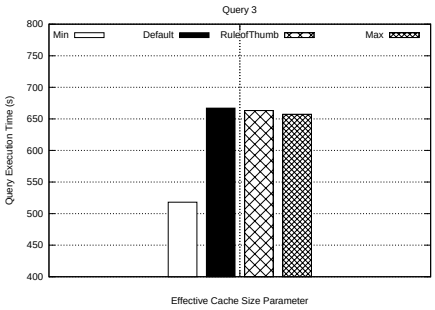
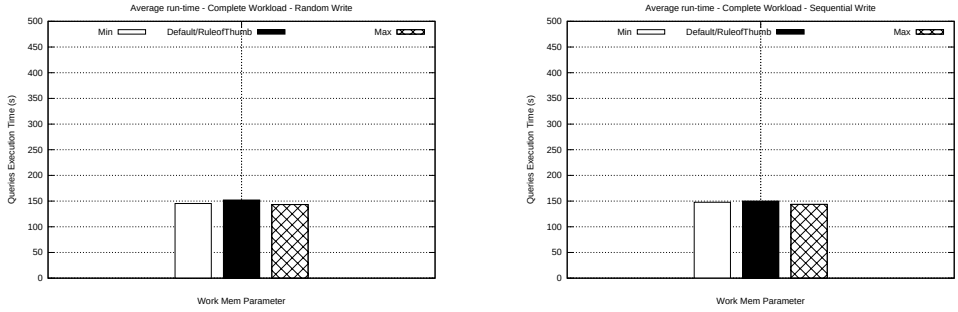


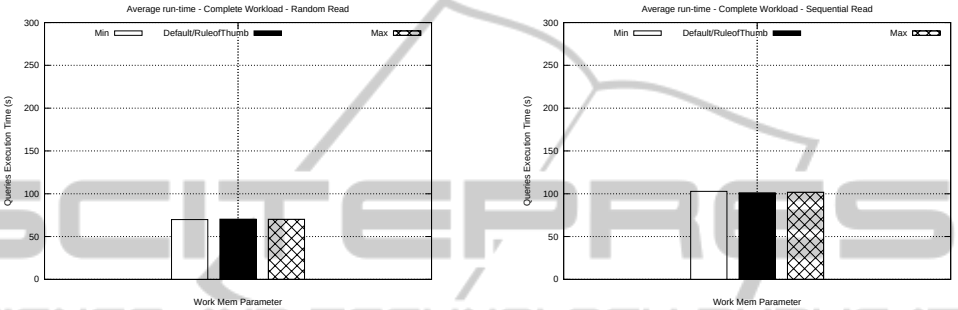
Figure 4: Query 3 - Execution average time considering *sequential write* and tuning in *effective_cache_size* parameter.

4.3 Work_Mem

The *work_mem* parameter, which limits the available amount of memory for ordination operations, affects

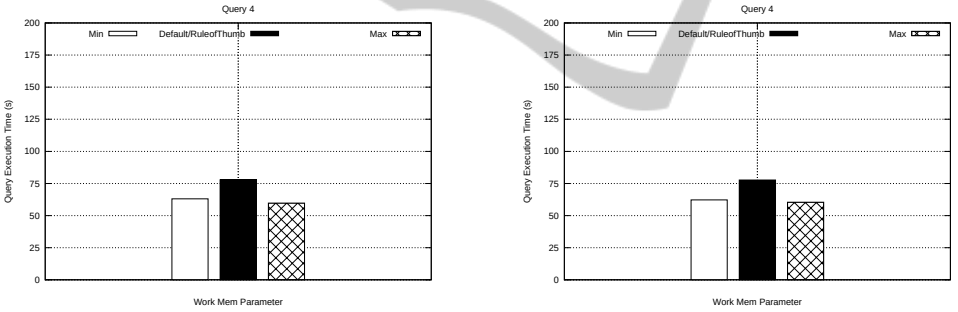


(a) Random Write (b) Sequential Write



(c) Random Read (d) Sequential Read

Figure 5: SQL workload execution average time considering I/O workload and tuning in *work_mem* parameter.



(a) Random Write (b) Sequential Write

Figure 6: Query 4 - Execution average time considering I/O workload and tuning in *work_mem* parameter.

only 1/3 of the queries in the workload. The average execution time of these queries shows that the *work_mem* parameter has low sensibility regarding the tuning rules as shown on Figure 5.

Performance improvements could be achieved by tuning each query or group of queries considering the concurrent I/O workload. Figure 6 shows an improvement for Query's 4 execution time on the workload for *Write* operations.

5 CONCLUSIONS

Virtualization provides exciting capabilities for IT services offerings. However, the simple partitioning of resources on virtualized environments does not guarantee good performance of legacy RDBMS. In this paper, we demonstrated this assertion through experiments simulating different concurrent I/O workloads on a virtualized environment. The experiments show that any efforts on tuning a given configuration parameter will not provide the expected benefits, due to the concurrent I/O workloads competing for the same physical resources. Finally, we demonstrated that using previously not advised parameters values might lead to systems performance improvements.

The results also highlight a need of new tuning rules of RDBMS in virtualized environments. Such rules must be aware of the peculiar characteristics of concurrent I/O workloads in these environments. Later, these rules may be applied to each query, according to their needs.

Similar to this work, a future research might investigate the potential benefits from tuning of configuration parameter values for *Online Transactional Processing* (OLTP) workloads. Furthermore, another relevant work, in complement to Soror's work (Soror et al., 2008), could be a tool to perform analysis on both types of concurrent disk access as well on the executing SQL queries. This way, tuning rules could be suggested on-the-fly, which would allow RDBMS to answer requests more accurately against the constant resources variation and workloads common in cloud computing environments.

REFERENCES

(2013). Bonnie++ benchmark. Available at URL: http://www.coker.com.au/bonnie++/.

(2013). Dbt3-tollkit database test 3. Available at URL: http://sourceforge.net/projects/osdldbt/files/dbt3.

(2013). Postgresql: The world's most advanced open source database. http://www.postgresql.org.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616.

Cecchet, E., Singh, R., Sharma, U., and Shenoy, P. (2011). Dolly: virtualization-driven database provisioning for the cloud. *SIGPLAN Not.*, 46(7):51–62.

Council, T. P. P. (2013). Tpc benchmark h - version 2.16.0. Technical report, Transaction Processing Performance Council.

Debnath, B. K., Lilja, D. J., and Mokbel, M. F. (2008). Exploiting the impact of database system configuration parameters: A design of experiments approach. volume 31, pages 3–10.

Delimitrou, C., Sankar, S., Khessib, B., Vaid, K., and Kozyrakis, C. (2012). Time and cost-efficient modeling and generation of large-scale tpcc/tpce/tpch workloads. In *Proceedings of the Third TPC Technology conference on Topics in Performance Evaluation, Measurement and Characterization*, TPCTC'11, pages 146–162, Berlin, Heidelberg. Springer-Verlag.

Duan, S., Thummala, V., and Babu, S. (2009). Tuning database configuration parameters with ituned. *Proc. VLDB Endow.*, 2(1):1246–1257.

Hsu, W. W., Smith, A. J., and Young, H. C. (2001). I/o reference behavior of production database workloads and the tpc benchmarks an analysis at the logical level. *ACM Trans. Database Syst.*, 26(1):96–143.

Rao, J., Bu, X., Xu, C.-Z., Wang, L., and Yin, G. (2009). Vconf: a reinforcement learning approach to virtual machines auto-configuration. In *Proceedings of the 6th international conference on Autonomic computing*, ICAC '09, pages 137–146, New York, NY, USA. ACM.

Smith, G. (2010). *PostgreSQL 9.0 High Performance*, chapter Server Configuration Tuning, pages 125–149. Packt Publishing, Limited.

Smith, J. E. and Nair, R. (2005). The architecture of virtual machines. *Computer*, 38(5):32–38.

Soror, A. A., Aboulnaga, A., and Salem, K. (2007). Database virtualization: A new frontier for database tuning and physical design. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, pages 388–394, Washington, DC, USA. IEEE Computer Society.

Soror, A. A., Minhas, U. F., Aboulnaga, A., Salem, K., Kokosielis, P., and Kamath, S. (2008). Automatic virtual machine configuration for database workloads. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 953–966, New York, NY, USA. ACM.

Storm, A. J., Garcia-Arellano, C., Lightstone, S. S., Diao, Y., and Surendra, M. (2006). Adaptive self-tuning memory in db2. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 1081–1092. VLDB Endowment.

Tran, D. N., Huynh, P. C., Tay, Y. C., and Tung, A. K. H. (2008). A new approach to dynamic self-tuning of database buffers. *Trans. Storage*, 4(1):3:1–3:25.

Xiong, P. (2012). Dynamic management of resources and workloads for rdbms in cloud: a control-theoretic approach. In *Proceedings of the on SIGMOD/PODS 2012 PhD Symposium*, PhD '12, pages 63–68, New York, NY, USA. ACM.