A Live Migration Approach for Multi-Tenant RDBMS in the Cloud

Leonardo O. Moreira, Flávio R. C. Sousa, José G. R. Maia,
Victor A. E. Farias, Gustavo A. C. Santos, Javam C. Machado

Universidade Federal do Ceará (UFC), Brazil
{leoomoreira, sousa, gilvanm, gsantos, javam}@ufc.br, victorfarias@lia.ufc.br

Abstract. Cloud computing is a trend of technology aimed at providing on-demand services with payment based on usage. To improve the use of resources, providers adopt multi-tenant approaches, reducing the operation costs of services. Moreover, tenants have irregular workload patterns, impacting in the guarantees of quality of service, mainly due to interference between the tenants. This paper proposes an approach to improve quality of service for multi-tenant RDBMS. This approach uses migration techniques of tenants, system monitoring and benefits of cloud infraestructure to improve performance and reduce costs of the provider. We carried out experiments on performance and resource usage in order to evaluate our approach.

Categories and Subject Descriptors: H.2 [**Database Management**]: Miscellaneous; H.3 [**Information Storage and Retrieval**]: Miscellaneous; I.7 [**Document and Text Processing**]: Miscellaneous

Keywords: Cloud Computing, Database Live Migration, Multi-tenancy, Quality of Service, Service Level Agreement

1. INTRODUCTION AND MOTIVATION

Cloud computing is a paradigm of service-oriented computing and has revolutionized the way computing infrastructure is abstracted and used. Elasticity, pay-per-use pricing, and economies of scale are the major reasons for the successful and widespread adoption of cloud infrastructures. We consider a cloud computing enviroment as a set of Virtual Machines (VMs) which may be assigned to different Physical Machines (PMs). Since the majority of cloud applications are data-driven, Database Management Systems (DBMSs) are critical components in the cloud software stack [Elmore et al. 2011]. When users purchase computing time from a cloud provider, typically both sides, user and provider, establish the Quality of Service (QoS) via a Service Level Agreement (SLA). It is crucial that cloud providers offer SLAs based on performance for their customers [Schad et al. 2010]. For many systems, most of the time consumed by requests is related to the DBMS, thus it is important that quality is applied to the DBMS to control the consumed time [Sousa et al. 2012].

To improve the resource management, increase resource utilization and reduce costs, providers implement resource sharing among tenants [Barker et al. 2012]. Multi-tenant is a technique for consolidating various tenant applications on a single system, and is often used to eliminate the need for separate systems for each tenant. Sharing of resources at different levels of abstraction and distinct isolation levels result in various multi-tenant models; the shared VM, shared DBMS, shared database and shared table are well known [Lang et al. 2012]. Such sharing improves providers' profits. Multi-tenant DBMSs have been used to host multiple tenants (databases) into a single system, allowing efficient resource sharing at different levels of abstraction and isolation [Elmore et al. 2011].

Database migration is a technique to move one or more tenants to another environment when changes in the performance of this tenant are detected [Barker et al. 2012]. This technique can be used to decrease the load on the host environment, and also to consolidate multiple tenants in an environment that they may share resources with reduced interference. Thus, in a way, it becomes possible to establish a scenario allocation for a set of tenants in a set of resources, where the main objective is to reduce interference between such tenants. In the cloud, the migration brings up some costs, for example, SLA costs and related costs to human intervention. However, to reduce these costs, the database migration technique must make fair decisions that involve the following variables: "when" determines the instant of time that the tenant must be migrated, "which": specifies what tenants should be targets of migration, "where": determines the tenant destiny and "how" that determines the manner, process or procedure used to migrate the tenant [Barker et al. 2012].

According to [Chaudhuri 2012], an interesting challenge is to develop techniques to guarantee the performance of multi-tenant DBMS. Dealing with unpredicted load patterns and elasticity is a challenging but critical task to ensure that tenant's SLAs are met [Elmore et al. 2011]. Moreover, before develop new techniques, it is necessary to understand how the workload of a tenant influences other tenant's behavior and even the isolation provided by a DBMS to prevent interference between tenants. This problem is addressed in some works [Elmore et al. 2011] [Ahmad and Bowman 2011] [Xiong et al. 2011] [Lang et al. 2012] [Hatem A. Mahmoud and El-Abbadi 2012] [Schiller et al. 2013] [Moon et al. 2013]. However, the studies do not address aspects of performance, they are resources-oriented because they focus on consolidating as many tenants in same hardware or VM, and/or do not check the isolation of each tenant.

This paper presents an approach to improve the QoS for multi-tenant RDBMS (*Relational Database Management System*) in the cloud. Our approach aims to reduce SLA violations through the migrating tenants when they need a resource with greater capacity. Furthermore, a performance experiment was made to show the effectiveness of our approach, in which techniques for migration, monitoring and QoS are used to achieve these goals. The major contributions of this paper are as follows:

—We present an approach to improve the QoS for multi-tenant RDBMS in the cloud by the migration techniques. While other approaches to improve QoS of resources have been proposed, our approach is to tackle aspects of performance, checking the interference between each tenant in the same RDBMS by monitoring techniques and SLA violations.
—We present a database migration strategy for cloud-based multi-tenant environments. Our strategy migrates the tenant quickly according to the current workload.
—We present a full prototype implementation and experimental evaluation of our end-to-end system. We demonstrate that this strategy is effective in managing databases in scenarios from data-driven cloud applications.

Organization: This paper is organized as follows: Section 2 explains theoretical aspects and implementation of our approach. Evaluation results are presented in Section 3. Section 4 surveys related work and, finally, Section 5 presents the conclusions.

2. OUR APPROACH

In this section we describe our approach to improve the performance of multi-tenant database systems. We use monitoring techniques to check the status of tenants according to the workload to ensure the SLA. To develop our approach, we extended our QoSDBC (*Quality of Service for Database in the Cloud*) architecture for managing databases in the cloud [Sousa et al. 2012].

We use shared DBMS model, because it is used more than shared table and shared database models [Elmore et al. 2011], and shared VM has lower interference among tenants. Thus, according to the model used, which has a database corresponding to a tenant on the system, in our approach, each VM can run multiple RDBMSs. Despite using more resources, this approach reduces interference among tenants as it works with fewer tenants on the same RDBMS. We use the definitions of QoS from our previous work [Sousa et al. 2012]. We use the SLA metric of response time: The value of maximum response time expected for processing the input workload. In our approach, each service has a SLA associated with it (e.g. Wiki_{SLA}= 0.5s).

To achieve our goals, some components had to be changed, while others had to be implemented from scratch. An overview of the architecture of the QoSDBC is shown in Figure 1.

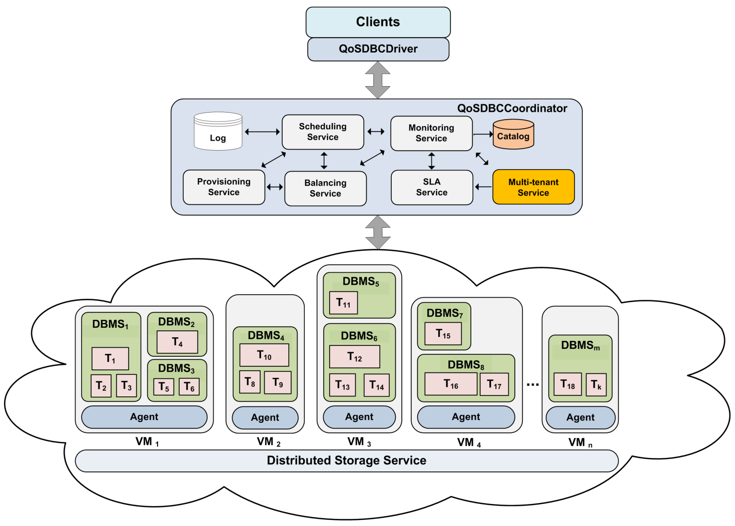

Fig. 1. QoSDBC architecture.

The *QoSDBCDriver* component is used by Java applications that want to use our platform of data management in the cloud. This component implements all the features of a common JDBC (*Java Database Connectivity*) driver. In this sense, any application that uses the JDBC standard to interact with the database, can migrate to QoSDBC. The *QoSDBCCoordinator* is the component responsible for receiving requests for connections through *QoSDBCDriver*. Moreover, it is the component that acts as an orchestrator of all the actions that are performed by QoSDBC. The component that has the responsibility of maintaining information about the cloud environment for QoSDBC is the *Agent*. It provides information from the environment through the *Catalog* component. The *Agent* knows the location of the *Catalog* and uses a JDBC connection to send the information collected in the VM. The pieces of information being monitored for a VM are: free CPU percentage, free space of main memory and secondary memory size available. Also the pieces of information that are tracked for each DBMS are: number of active connections for each database and their sizes they occupy in secondary memory. To find out what queries are running on tenants, the *QoSDBCCoordinator* intercepts the query, executes it, extracts both its local and global response times. The local response time is the time it takes for the query to run in the tenant, while the global response time is the time it takes for a request to come to the QoSDBC and its response returns to the tenant. All this information is stored in the *Catalog* component. The *Catalog* component keeps some information about the state of VMs, the RDBMS and tenants. The *Catalog* is a distributed database that is available in the cloud. Thus, characteristics of fault tolerance, availability and redundancy are achieved by the same cloud infrastructure where QoSDBC is operating. This database stores information about what tenants that are active in each VM, number of connections in each RDBMS, the size of each tenant, state of memory, disk and CPU for each VM. The *Log* is the component responsible for maintaining information about the queries that are being sent and executed in tenants. However, only update informations are kept in this database. The purpose of this database is to keep this information to assist the migration process and restore of the tenant.

Our migration strategy is tightly coupled to the components of QoSDBC. For each application, we associate an SLA described by our quality model. When you start QoSDBC, the agents begin monitoring the tenants of RDBMSs and VM to know the state of the environment. When *QoSDBCCoordinator* identifies an SLA violation of a tenant, it explores the monitored data and starts, in parallel, the backup of the tenant, provisioning a new VM to receive the tenant. After that, the collection of queries that came after the backup starts are executed in the old copy of tenant and stored in QoSDBC sytem. Until the completion of the backup process and provisioning of a VM, the tenant is restored in the new VM. When the restore process is completed, to ensure consistency the update queries that were executed in the old copy are propagated to the new VM. At this time, there is a lockout period that the tenant is inactive. When the tenant is fully migrated, the *QoSDBCCoordinator* warns *QoSDBCDriver* for direct connection to the new address of the tenant. After that, the old tenant is deleted.

3. EVALUATION

To evaluate the efficiency of our approach, we use the results of experiments carried out in one of our previous works [Moreira et al. 2012]. In this work, we show that some tenants could generate interference that reduce the QoS of others. In this regard, we will apply our approach to decrease the number of SLA violations during the occurrence of such interference. Our experiments aim to show how our approach can minimize the SLA violations between tenants, thereby improving the QoS for tenants. In our experiment we used the RDBMS MySQL 5.5 and OS Ubuntu 12.04.1 LTS with InnoDB engine and 128MB of buffer. We use OLTPBenchmark [OLTPBenchmark 2013] to simulate different workloads to the RDBMS in the experiments. OpenNebula [OpenNebula 2013] was the provider used to perform the experiments. In this work we used only VM instances that have 2 CPUs, 2GB RAM, 30GB of disk. For these experiments, we used the quality model that considers the response time metric. For our experiment environment, we create a VM image with the setup described at the beginning of this section. This will be used in cases where our quality strategy requires a provisioning of a VM. To simulate the tenants, we use the OLTPBenchmark [OLTPBenchmark 2013]. The OLTPBenchmark is a framework to evaluate the performance of different RDBMSs facing settings of OLTP workloads. The framework has several benchmarks representing different applications, such as TPC-C, Twitter, YCSB, AuctionMark, Wikipedia. As tenants we use TPC-C (400MB), YCSB (800MB) and Wikipedia (600MB). For this experiment were specified the following SLAs: TPC-C_{SLA}= 100.0ms, YCSB_{SLA}= 50.0ms, Wikipedia_{SLA}= 50.0ms. All tenants were placed in a VM and OLTPBenchmark in another VM.

3.1 Experimental Results

The objetive of this experiment is to analyze the number of SLA violations in a situation of increasing workload in only one tenant, the TPC-C. In order to bring the experiments execution closer to that of a real environment, only values after the first 120 seconds of obtained measures were considered to avoid that delays in the startup of tenants could impact the results. In our experiment was performed varying the rate of transactions over a period of time. In the this experiment, the rates of YCSB and Wikipedia were fixed at 60, and the rate of the TPC-C was changed according to the sequence (50, 100, 150, 200, 250), every 300 seconds. Figure 2 show the variation of the average response time for the experiment without QoSDBC system. Already the Figure 3 show the variation of the average response time for the experiment with QoSDBC system.

In [Moreira et al. 2012] we show that these three tenants, TPC-C is what causes more interference. Therefore, we increased the rate of the TPC-C to that had SLA violation and thus show how our approach behaves. In our experiment, the TPC-C tenant, after the first SLA violation, has been migrated to a new VM. As shown in the experiments, we can see that our approach reduces the number of SLA violations for tenant TPC-C. However, during the migration process has been SLA

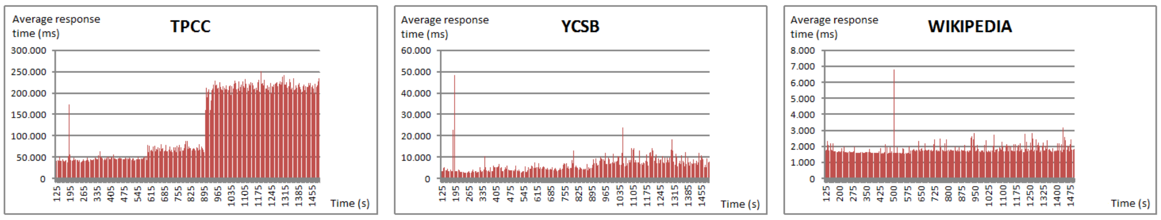

Fig. 2. Average response time without QoSDBC system

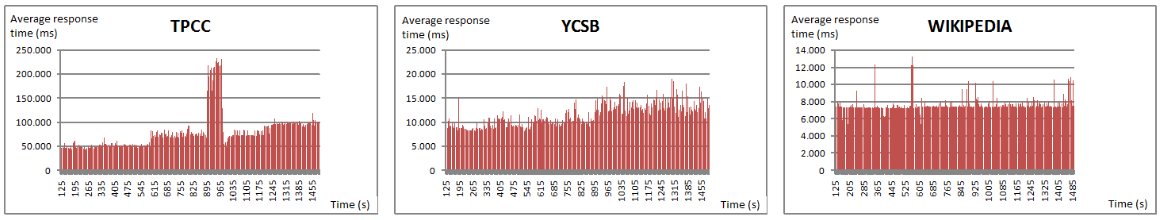

Fig. 3. Average response time with QoSDBC system

violations. The best way would be to anticipate the migration process that had not SLA violations. Furthermore, it was found that there was an increase in the average response time of the transactions. A justification for this is that our approach intercepts queries and performs monitoring tasks, like adding an increase in transaction execution.

4. RELATED WORK

In [Elmore et al. 2011] various multi-tenant models are presented (shared PM, VM, OS, DBMS, database, database schema and database table) for the use of database in virtualized environments. However, only problems achieved when a DBMS has several databases are presented and, still, no evaluation or results of the study are shown. [Barker et al. 2012] presents a solution to databases migration that satisfies some workload constraints. It employs the shared DBMS model and aims to minimize the impact of decreased performance during the migration process of the databases. The relationship between the DBMS and the database is one to one. In this sense, a DBMS can only have one database. However, the focus of this work is the migration of databases, and it does not support other multi-tenant models. The work presented by [Xiong et al. 2011] discusses a solution for managing resources for shared DBMS in the cloud. However, the model presented in this work can not be used on public providers, such as Amazon, since it requires physical access to infrastructure resources.

[Schiller et al. 2013] presents an approach that implements live migration designed for multi-tenant RDBMS, where applications running OLTP workloads execute in tenants. This approach uses a better management of data pages and the buffer pool in the RDBMS. However, this approach is tightly integrated with the used RDBMS, workload type, concurrency control etc. Finally, one of the disadvantageous aspects of this approach is that it not ensures serializability criteria and does not prevent all concurrent access anomalies. In [Moon et al. 2013] it is shown a method to manage replicated multi-tenant databases. To reduce the workload of a given replica that is violating SLA, this work uses a load balancing strategy that finds a subset of replicas where the workload can be directed to. However, the load balancing strategies are appropriated to solve performance problems in replicated environments. Furthermore, it does not eliminate SLA violations when a replica of the tenant is overloaded, because this work does not perform provisioning. [Hatem A. Mahmoud and El-Abbadi 2012], on the other hand, aims to minimize the necessary amount of resources for a set of tenants, satisfying an SLA for each tenant. From the viewpoint of multi-tenant strategies, this work uses shared DBMS model. Besides, the presented solution is oriented to OLAP applications, and its focus is not to exceed the resources limits of the environment. Therefore, it does not aim the QoS through the SLA.

5. CONCLUSION

The work presents an approach to improve QoS for multi-tenant RDBMS in the cloud by migration techniques, covering its architecture and performance aspects. According to the results, our approach reduces the number of SLA violations through migration and provisioning of new resources. In future work, we intend to anticipate the provisioning and therefore further reduce the number of SLA violations. For this purpose, our prediction work that forecasts workload based on a short window of observation of the environment [Santos et al. 2013] will be added to our infrastructure. In addition, we intend to improve the allocation of tenants in the environment to avoid unnecessary provisioning. The monitored information are not being used in our work. We intend to add this information to our prediction work to improve its effectiveness.

REFERENCES

Ahmad, M. and Bowman, I. T. Predicting system performance for multi-tenant database workloads. In *Proceedings of the Fourth International Workshop on Testing Database Systems*. DBTest '11. ACM, New York, NY, USA, pp. 6:1–6:6, 2011.

Barker, S., Chi, Y., Moon, H. J., Hacigümüş, H., and Shenoy, P. "cut me some slack": latency-aware live migration for databases. In *Proceedings of the 15th International Conference on Extending Database Technology*. EDBT '12. ACM, New York, NY, USA, pp. 432–443, 2012.

Chaudhuri, S. What next?: a half-dozen data management research goals for big data and the cloud. In *Proceedings of the 31st symposium on Principles of Database Systems*. PODS '12. ACM, New York, NY, USA, pp. 1–4, 2012.

Elmore, A., Das, S., Agrawal, D., and Abbadi, A. E. Towards an elastic and autonomic multitenant database. In *NetDB 2011 - 6th International Workshop on Networking Meets Databases Co-located with SIGMOD 2011*, 2011.

Elmore, A. J., Das, S., Agrawal, D., and El Abbadi, A. Zephyr: live migration in shared nothing databases for elastic cloud platforms. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. SIGMOD '11. ACM, New York, NY, USA, pp. 301–312, 2011.

Hatem A. Mahmoud, Hyun Jin Moon, Y. C. H. H. D. A. and El-Abbadi, A. Towards multitenancy for io-bound olap workloads. Tech. Rep. UCSB-2012-02, CS Department, University of California, Santa Barbara. May, 2012.

Lang, W., Shankar, S., Patel, J. M., and Kalhan, A. Towards multi-tenant performance slos. *Data Engineering, International Conference on* vol. 0, pp. 702–713, 2012.

Moon, H. J., Hacümüş, H., Chi, Y., and Hsiung, W.-P. Swat: a lightweight load balancing method for multitenant databases. In *Proceedings of the 16th International Conference on Extending Database Technology*. EDBT '13. ACM, New York, NY, USA, pp. 65–76, 2013.

Moreira, L. O., Sousa, F. R. C., and Machado, J. C. Analisando o desempenho de banco de dados multi-inquilino em nuvem. In *XXVII Simpósio Brasileiro de Banco de Dados, SBBD 2012*. pp. 161–168, 2012.

OLTPBenchmark. *OLTPBenchmark*, 2013. `http://www.oltpbenchmark.com`.

OpenNebula. *OpenNebula - Open Source Data Center Virtualization*, 2013. `http://www.opennebula.org`.

Santos, G., Maia, J., Moreira, L., Sousa, F., and Machado, J. Scale-space filtering for workload analysis and forecast. In *Proceedings of the 6th IEEE International Conference on Cloud Computing*. IEEE CLOUD '13. IEEE, Santa Clara Marriott, CA, USA, pp. 677–684, 2013.

Schad, J., Dittrich, J., and Quiané-Ruiz, J.-A. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proc. VLDB Endow.* 3 (1-2): 460–471, Sept., 2010.

Schiller, O., Cipriani, N., and Mitschang, B. Prorea: live database migration for multi-tenant rdbms with snapshot isolation. In *Proceedings of the 16th International Conference on Extending Database Technology*. EDBT '13. ACM, New York, NY, USA, pp. 53–64, 2013.

Sousa, F. R. C., Moreira, L. O., Santos, G. A. C., and Machado, J. C. Quality of service for database in the cloud. In *CLOSER*, F. Leymann, I. Ivanov, M. van Sinderen, and T. Shan (Eds.). SciTePress, pp. 595–601, 2012.

Xiong, P., Chi, Y., Zhu, S., Moon, H. J., Pu, C., and Hacigumus, H. Intelligent management of virtualized resources for database systems in cloud environment. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*. ICDE '11. IEEE Computer Society, Washington, DC, USA, pp. 87–98, 2011.