Submission date: 28/Apr/2017
Resubmission date: 22/Dec/2017
Camera ready submission date: 06/Mar/2018

1st round notification: 09/Apr/2018
Available online: 12/May/2018
Publication date: 30/Jun/2018

Section: survey

DBaaS Multitenancy, Auto-tuning and SLA Maintenance in Cloud Environments: a Brief Survey

Vinicius da S. Segalin[1], **Carina F. Dorneles**[1], **Mario A. Dantas**[2]

[1]Departamento de Informca e Estattica - INE/CTC
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianis – SC – Brazil

[2]Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brazil

vinicius.segalin@posgrad.ufsc.br, carina.dorneles@ufsc.br,
mario.dantas@ice.ufjf.br

Abstract. *Cloud computing is a paradigm that presents many advantages to both costumers and service providers, such as low upfront investment, pay-per-use and easiness of use, delivering/enabling scalable services using Internet technologies. Among many types of services we have today, Database as a Service (DBaaS) is the one where a database is provided in the cloud in all its aspects. Examples of aspects related to DBaaS utilization are data storage, resources management and SLA maintenance. In this context, an important feature, related to it, is resource management and performance, which can be done in many different ways for several reasons, such as saving money, time, and meeting the requirements agreed between client and provider, that are defined in the Service Level Agreement (SLA). A SLA usually tries to protect the costumer from not receiving the contracted service and to ensure that the provider reaches the profit intended. In this paper it is presented a classification based on three main parameters that aim to manage resources for enhancing the performance on DBaaS and guarantee that the SLA is respected for both user and provider sides benefit. The proposal is based upon a survey of existing research work efforts.*

1. Introduction

The service oriented computing paradigm known as cloud computing has been growing quickly in the past years and changing the way computing is perceived and used by people and corporations [1]. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are the most popular paradigms in the cloud nowadays and are already consolidated on the market [2]. However, they can be extended to other types, such as Database as a Service (DBaaS) [3]. The latter, in the same manner as the traditional database, presents several complex questions to be worked on. In this

Cite as: Segalin, V. S., Dorneles, C. F. and Dantas, M. A. (2018). DBaaS Multitenancy, Auto-tuning and SLA Maintenance in Cloud Environments: a Brief Survey. iSys: Revista Brasileira de Sistemas de Informa (Brazilian Journal of Information Systems), 11(2), 30-42.

scenario, the resources and performance can reach the ideal level for each situation using the advantages the cloud provides, like elasticity, pay-per-use and low upfront investment.

The importance of considering performance and resource management aspects in DBaaS and the need for a continuous research for effective solutions become clear in the recent emergence of numerous initiatives and applications. One of the first initiatives in DBaaS field worried about the emerging scenario of enterprise services using **multi-tenant** databases [4]. At that time, the experiments showed database vendors did not support well multitenancy and would need to enhance their products to better handle new requirements regarding databases as a service. Years later, some works tried to find the best way to manage resources among tenants [5, 6, 7], since DBMSs had already consolidated multi-tenant products and handled well this model. Although researches regarding multitenancy are far from complete, much work has been done with different focuses to complement the model and improve DBaaS performance.

Cloud computing uses Virtual Machines (VMs) to host the hired service [9], and many works [10, 11, 12, 13, 14, 15, 16, 17, 7, 18, 19] focus on finding different approaches for virtualization, implying that it limits the performance serving as a bottleneck. Other works [17, 20, 21, 7, 6, 5, 22] try to solve the issue of **auto-tuning**. These works state that the workload coming from the cliente may vary at any time, and if the cloud is not prepared for this variation, performance might decay, and as a consequence, the SLA will get compromised, harming either the client or the provider.

Another situation that may cause problems to both sides of the cloud service is the misunderstanding from the **service contracting** party (the costumer) [8, 23]. In this case, one can hire a service that finds suitable for the occasion when, in fact, it is not, and ends up hiring a service that does not meet the requirements and causes monetary loss for client and server. In this scenario, the big issue is how to separate cloud users from the details of compute resources behind a cloud management service.

This paper presents a survey of some important works on performance and resource management in DBaaS described in literature and provides a qualitative analysis about them. As a result of our literature analysis, we present a brief discussion about existing works and a taxonomy that explains their common and specific features. We expect the survey to be helpful to a researcher who is designing new approaches or who is looking for a library of approaches.

The existing works are organized through a high level classification, in three categories we consider relevant to the topic. These categories are still in research and in development to improve the way resources are attributed to each customer in order to improve its performance and generate more profit to the provider. The first one, regarding *multitenancy*, discusses some ways of handling multiple customers within the same software instance, such as using private virtual machines to each tenant, or sharing the machines, but isolating the tenants in a table level. The second category discussed in this overview is the *automatic tuning* of the cloud database in order to achieve its maximum potential with the least interference possible from the user. The last one is about different types of approaches to guarantee *SLA maintenance* other than worrying with multitenancy and auto-tuning, providing the user a higher-level SLA proposal, or performing query se-

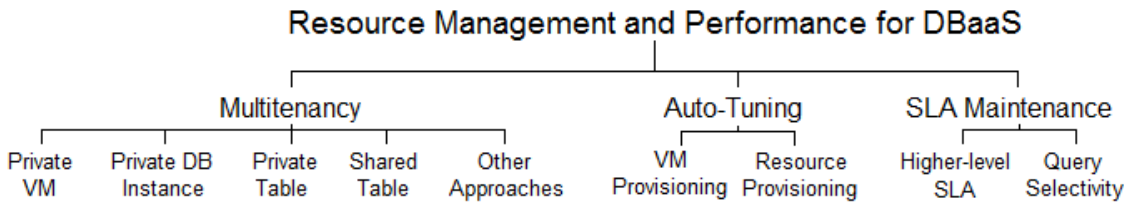

Figure 1. Classification of resource management and performance for DBaaS

lectivity in order to obey SLA conditions. To illustrate the categories mentioned above, we propose a taxonomy, presented in Figure 1, which will be used as a guide to present the related approaches.

This paper is organized as following. Section 2.1 discusses multitenancy in order to escape from the traditional VM approach. Some studies to improve auto-tuning and prevent SLAs to be broken are discussed in Section 2.2. Section 2.3 presents different approaches to guarantee SLA still not mentioned in previous sections. Section 3 discusses challenges and suggests future research directions, while Section 4 concludes this survey.

2. Multitenancy, Auto-Tuning and SLA Maintenance

In this section, we present existing proposals that aim at developing solution on DBaaS focusing on multitenancy architecture, autotuning task or SLA guarantee.

2.1. Multitenancy

Multitenancy is a software architecture in which a single instance of a software application serves multiple customers, and each customer is called a tenant. There are several types of multitenancy providing different levels of isolation[24, 4], and for each isolation level a different abstraction will be given to the tenant. In this section, some of these approaches will be addressed.

2.1.1. Private Virtual Machine

Using a private virtual machine to host database tenants provides total isolation among them, since it virtualizes the machines where the databases are hosted on and it works for the tenant as it had a physical machine for itself. Each virtual machine hosts only a single database giving total isolation across tenants, isolating its data and resources, providing security to it, with the drawback of performance overhead and waste when the system is underutilized. Private VM can be used for each tenant in order to provide database replicas more easily and address the challenges of provisioning shared-nothing replicated databases in the cloud [25]. This scheme, where each tenant runs its own virtual machine, is used by SmartSLA [5], a cost-aware resource management system to intelligently manage the resources in a shared cloud database system. Amazon RDS[26] uses this approach to provide a fully-managed SQL database service. According to Amazon's RDS pages, Amazon Relational Database Service is a web service that makes it easy to set up, operate, and scale a relational database in the cloud, and is one of the most used database services in the world.

2.1.2. Private DB Instance

Similar to the traditional private VM approach regarding to isolation, with a private database instance there may be multiple database instances within one single physical, or virtual, machine, and each tenant uses one instance. CasJobs [10] is an example of private database instance used to allow local analysis to several users, each one considered as a tenant, on a very large science database. Although it escapes from the private completely isolated virtual machine approach, private database instance still has a high performance overhead, limiting the system scalability. Also, it imposes high costs for maintenance, data backup and hardware, since the number of tenants that can share the same database is limited by the server. However, this approach presents some improvements comparing to private VM in relation to resource sharing, as now the tenants are hosted in the same machine and can use the same disk and processing power. A similar approach is Private Database, or Shared Process, where multiple databases may be created within a single instance, saving memory from starting many database instances. This shared process approach is used by Relational Cloud [11] to provide efficient multitenancy, elastic scalability and database privacy, problems that are not well addressed by the main database cloud providers. Microsoft Azure SQL Database[27] uses this scheme to provide a relational database service in the cloud based on Microsoft SQL Server, and along with Amazon is one of the market leaders nowadays.

Another alternative is a framework [7] that helps balancing multitenancy with performance-based SLOs (Service Level Objectives). This framework uses the Private Database Instance approach, but concerning on how to schedule the tenants on each hardware resource in order to guarantee both performance goals and cost reduction for the provider. An improvement to the Shared Process approach is presented in [18], where a single database server process hosts databases of different tenants. In this method, service providers do not give any assurances to a tenant in terms of isolating its performance from other co-located tenants, not assuring the performance required. In this case, they propose a mechanism called SQLVM, which is an abstraction for performance isolation built on a promise of reservation of key database server resources, such as CPU, I/O and memory. Another work that proposes an enhancement to the Shared Process approach is described in [19]. The proposal focuses on buffer pool memory, stated as an essential resource for good performance of a tenant's workload, since it serves as a cache of database pages.

2.1.3. Private Table

In this approach the tenants share the same database process, but each of them have their own tables, either in the same schema or not, depending on the system architecture. Using schemes has a lower complexity over not using them, since the SQL statements can be the same, only redirecting to the correct schema, and also backup and migration scripts can be reused. This approach is better than the previous ones at pooling memory, where studies indicate that it can scale up to thousands of active tenants per server, a two orders of magnitude improvement over the shared machine approach. Also, private table allows tenants to share connection pools, enhancing the performance of executing commands

on the database, with the downside of vulnerability, leaving the security to be handled at the application layer. Focusing on minimizing SLA violation penalties, [12] adopts this scheme to propose a solution to the tenant placement problem to maximize the provider's profit, and consequently also the customer's satisfaction. The work presented in [13] proposes a virtualization advisor through the private table approach to assist with design and configuration of the cloud such that machine utilization is optimized and service quality is ensured.

2.1.4. Shared Table

For this approach all the tenants share the same database process and the same tables, without schema separation. This is the cheapest of all the approaches presented, with the lowest disk storage and the best pooling resources. It is the most scalable scheme and is limited only by the number of rows the database can hold, but for every big advantage there must be a drawback. For this approach security may be easily compromised if, for example, one table becomes corrupted, causing all the tenants to be affected. Optimization also becomes a big problem, since all the queries have to deal with data from all the costumers. Force.com [14] uses the shared table model to deliver robust, reliable, Internet-scale application to over 50 thousand organization and have become the global leader in CRM. Megastore [15] is a storage system mixing the scalability of NoSQL to the convenience of a traditional RDBMS, which uses the shared table scheme in order to ensure strong consistency and high availability.

2.1.5. Other Approaches

An adaptive database schema design [16] is proposed to serve as a middle ground between Private Table and Shared Table approaches. The authors state that the former has poor scalability since it needs to maintain large numbers of tables, and the latter achieves good scalability at the expense of poor performance and high space overhead, thus an adaptive schema is needed to find a balance between them to achieve good scalability and high performance with low space requirement.

Alternative mechanisms must be provided to avoid the performance drawbacks of VM-based hard isolation, which means allowing for resource sharing when the system is underutilized, but enforcing quotas when necessary. A solution would be a framework called DBSeer [17], that automates the throttle of certain transactions or tenants, which is already done by some database systems, but manually.

2.1.6. Multitenancy Remarks

Table 1 compares the approaches discussed in this section. The comparison features were defined based on their importance in a multitenancy software architecture. The table compares (i) the type of isolation of each approach, whether it is hard, medium or soft, (ii) the complexity of implementation and deployment, (iii) the disk storage needed, (iv)

Table 1. Multitenancy approach comparison

	Isolation	Complexity	Disk storage	SLA	Scheme	Main contribution
Dolly [25]	Hard	Low	High		PVM	Database replication for big queries
SmartSLA [5]	Hard	Low	High		PVM	Resource allocation decision
Amazon RDS [26]	Hard	Low	High		PVM	World leading DBaaS
CasJobs [10]	Hard	Low	High		PDB	Allow local analysis for big queries
Relational Cloud [11]	Hard	Low	High/Medium	X	PDB	Elasticity and privacy
Mic. Azure [27]	Hard	Low	High/Medium		PDB	Competitive prices
SLOs [7]	Hard	Low	High	X	PDB	Performance and cost reduction
SQLVM [18]	Hard	Low	High/Medium	X	PDB	Reservation of key resources
Shared Buffer [19]	Hard	Low	High/Medium	X	PDB	Buffer pool memory
PMAX [12]	Medium	Medium	Medium	X	PT	Maximize profit
DB Virtualization [13]	Medium	Medium	Medium	X	PT	Service quality
Force.com [Weissman et al. 2009]	Soft	High	Low	X	ST	Robustness
Megastore [15]	Soft	High	Low	X	ST	Consistency and availability
Adap. Schema [16]	Medium	Medium	Low		Other	Scalability and high performance
DBSeer [17]	Soft	Medium	Medium	X	Other	Resource sharing among tenants

PVM : Private Virtual Machine
PDB: Private Database
PT: Private Table
ST: Shared Table

if it minimizes SLA violation, and (v) its main contribution. Each approach and scheme has its advantages and drawbacks, so the client should choose which one to use according to his business needs.

2.2. Auto-tuning

Database tuning is the activity of making a database application run more quickly [28], which means the database should response within the least possible time to the client/ application request. To make it possible, the database should be tuned in order to work together with the request source in a manner that they can make the most of each other's capabilities. Therefore, parameters of the database and configuration of the operating system and hardware must be frequently optimized, preferably in an automatic fashion. In conventional DBMSs, DBAs have an important work on tuning the databases to achieve their optimal performance since they have total access and control of it. However, on the

cloud, with its paradigm of reducing the client burden, neither the DBA nor the tenant owner have this level of access to the service hired, making the problem of tuning much harder. With this in mind, the following works propose solutions to auto-tuning in the cloud to improve the database's performance and reach both provider and client goals.

2.2.1. Virtual Machine Provisioning

The framework proposed in [20] continuously monitors the database workload, tracks the satisfaction of the application-defined SLA, evaluates the condition of the action rules and takes the necessary actions when required, either by scaling out or in. Horizontal scalability, which means adding and removing computing resources according to the application workload and requirements, together with the master-slave replication strategy, where updates are sent to a single master node and lazily replicated to slave nodes, are the keys to make the framework able to automatically configure the database service and avoid SLA violation.

Current solutions for auto-tuning result in heavy performance impact during elastic scaling, and since one of the main reasons for using cloud services is elasticity, this should be solved. A technique for live migration in a multitenant database is thus presented, named Albatross [21], where the database cache and state of active transactions are migrated to ensure minimal impact on transaction execution during migration.

A framework that shows the service provider, how much hardware to provision to each tenant and how to schedule them based on the tenants workload, their performance SLOs and the hardware that is available in the server, is described in [7]. With this, it is promised to achieve hardware provisioning policy, which specifies how many machines to provision for a given set of tenants, as well as tenant scheduling policy, that finds an efficient mapping of the tenants to the provisioned machines in order to minimize the overall cost of doing so.

A framework that addresses the problem of resource provisioning for databases in the cloud is described in [6]. With a query workload, the framework is responsible for continually optimizing the system's operational cost based on the service provider's pricing model and the client's QoS expectation. With this in mind, the proposed framework identifies a set of infrastructure resources that satisfy the SLA and chooses the one that will best suit the client's needs. It also routes, at run-time, incoming queries to specific machines in order to make the best out of reserved resources.

An SLA-driven dynamic strategy [22] aims to calculate the optimal number of VMs for future requests subject to the unavailability probability, which is defined as an SLA metric, below a desired threshold. Although this method is not specifically made for DBaaS, it is suitable for the case. Experiments show that the strategy achieves its goal of saving cost and guaranteeing SLA.

Table 2. Auto-tuning approach comparison

	Dynamic tuning	VM	Resource	Main contribution
DBseer [17]			X	Resource sharing among tenants
Consumer-Centric [20]	X	X		Dynamically database tuning
Albatross [21]	X	X		Lightweight elastic scaling
SLOs [7]		X		Cost-optimization for performance SLOs
Generic framework [6]	X	X		Generic framework for cloud databases
SmartSLA [5]	X		X	Dynamic resource allocation
SLA-driven dynamic [22]	X	X		Minimizing the unavailability probability

2.2.2. Resource Provisioning

Some works, such as [17], state that the current solution for tuning databases in the cloud is offering one or a small number of fixed configurations to the user to choose from, which means the user will have to choose, before running his workload, one fixed configuration for the database without the possibility of personalizing it. Thus, they propose an automated solution that achieves workload-specific DBMS tuning by observing multiple workloads running on a pool of machines, extracting performance/resource characteristics of each workload, and assigning them to a set of different tuned DBMSs, solving workload-to-DBMS placement and DBMS tuning problems.

SmartSLA [5], a cost-aware resource management system, can intelligently manage the resources in a shared cloud database system. They developed two modules. The first uses machine learning to find the profit margins for each client using different resources available in the provider, such as CPU and memory. The second module dynamically adjusts the resource allocations to achieve the optimum profits. With this, it is said that SmartSLA can achieve optimal resource allocation in a dynamic and intelligent fashion.

2.2.3. Auto-tuning Scenarios

Table 2 summarizes the approaches discussed in Section 2.2. The comparison features were defined based on their importance in cloud auto-tuning scenarios. The table shows if (i) the approach dynamically tunes the database, whether it uses (ii) virtual machine or (iii) resource provisioning to tune it, and (iv) the approach's main contribution. Again, it is the user's choice to decide which one to use, specially regarding VM provisioning (scaling out) or resource provisioning (scaling up).

2.3. SLA maintenance

In previous sections we have discussed two big areas of performance and resource management in DBaaS: multitenancy and auto-tuning. The former showed a few different ways of dealing with multitenant databases in the cloud, each of them with different proposals and benefits. The latter talked about ways of tuning the database in the cloud in an automatic fashion with different objectives, but focusing on guaranteeing SLA. This final section of the survey addresses different types of approaches also to guarantee SLA, but in alternative and innovative ways.

2.3.1. Higher-level SLA

A new interface to change the way the client interacts with the provider and chooses the resources [8] is proposed because, as the authors have stated, the customer's lack of knowledge usually results in a poor choice of resources causing problems for himself and for the provider. For instance, the wrong choice may harm the provider through waste of resources and financial loss. The client, on the other hand, may suffer with the SLA not being complied, having a poor performance for his data. To avoid this kind problem due to misunderstanding they propose a new interface where the client provides the database schema and basic statistics, and as return their approach shows which kind of query the client will be able to request with its monetary cost and runtime.

Bazaar [23], similarly to the previous approach, tries to stop the client from making bad decisions regarding the resource choice. The main goal is avoiding resource waste, so the provider will be able to accept future requests from other tenants. In this approach the clients submit the specification of their jobs and high-level goals, such as the job completion time and/or desired cost. The framework will then provide a set of resource combination that obey the SLA and choose, among them, the one that suits better the provider. With this, the client is benefited since the use is simplified and more intuitive, and the provider has a higher profit.

2.3.2. Query Selectivity

An admission control framework named ActiveSLA [29] allows DBaaS providers, through a prediction module, to estimate the probability for a new query to finish the execution before its deadline. Then, based on that, a decision module decides whether or not to admit the given query into the database system. With this, it is said that ActiveSLA is able to make accurate and profit-effective admission control decisions to help the provider to meet the SLA.

iCBS [30] is an optimization of an existing scheduling algorithm called CBS, which is responsible for making online decisions regarding query scheduling. iCBS can, thus, help the provider to handle the highly demanding query volumes presented in the cloud from diverse customers and schedule them in order to minimize the SLA cost. iCBS takes the query costs derived from the service level agreements (SLAs) between the service provider and its customers into account to make cost-aware scheduling decisions.

A vision and initial design for a new workload management framework is presented in [31], which offers a new grammar to capture performance goals and constraints for data processing applications. With this new grammar, called XCLang, the authors propose an extensible query admission control that decides which queries should be allowed in for execution, aiming to reduce the chance of overload. After the query execution is allowed by the admission control, query scheduling is responsible for minimizing SLA violation by routing the queries for execution in different database servers.

Table 3. SLA maintenance approach comparison

	Higher-level SLA	Resource	Query Selectivity	Main contribution
Myria [8]	X			Intuitive interface
Bazaar [23]	X	X		High-level goals to resource allocation
ActiveSLA [29]			X	Avoid query rejection scheduling
iCBS [30]			X	Optimize CBS
SLA-Driven [Stamatakis et al. 2014]			X	Query admission and scheduling

2.3.3. SLA Maintenance Scenarios

In Table 3 we summarize the approaches presented in Section 2.3. The comparison features were defined based on their importance in SLA maintenance scenarios. It shows if the approaches (i) propose a new interface for the service, (ii) optimize resource selection, use (iii) query selectivity, and (iv) their main contribution. The works presented in this section propose different approaches that could be combined to meet the user's requirements, thus one should not need to choose between one of them exclusively.

3. Challenges and Perspectives

Despite performance and resource management is one of the main concerns when it comes to cloud services, there are several research gaps in this area. Therefore, in this section we highlight some of the gap aspects.

- **Higher-level SLA:** as already mentioned in this paper, customers tend to choose resources badly for their databases in the cloud. Some works [17, 8, 23] point out this problem and propose solutions, but we consider this matter has not yet been fully addressed and is a promising challenge, not only for DBaaS, but for all the other cloud services. Also, as seen in Table 3, no approach proposes a solution for the problem mentioned above using query admission control or any other solution for enhancing the performance, thus we consider this a challenge that can be addressed in the future.
- **Auto-tuning:** another theme discussed in this paper, we consider auto-tuning a key task that still has a lot to progress. As stated before, the customer has few or no access to administrative tasks in a database as a service, such as resource provisioning and performance control, so they all need to be automated to deal

with the challenges the cloud benefits propose, such as resource elasticity and multitenancy. Table 2 shows us dynamic tuning is hardly achieved with resource provisioning. An idea for the future might be finding an approach that dynamically tunes the database by provisioning VMs and resource to the tenants as they need.

- **Green Cloud:** cloud computing can offer energy saving in the provision of computing and storage services [32] by slowing down CPU speeds, turning off entire machines and scheduling jobs efficiently, for instance, but the constant need for resources and better performance has become a challenge for energy-aware services. Knowing that saving energy is a very important task not only for cloud services, but for every activity in our lives, this is an important field that must be addressed in the near future.
- **Resource reservation:** the cloud can give us, sometimes, a false impression of endless resources due to the high processing power and high availability the providers have. This is not true, though, and users must take that into account before taking some decisions, such as changing their workload dramatically without previous warning. When the workload variation is predictable, users can reserve resources for the moment they will need them, as made by [33, 34, 35].

There are many other challenges regarding DBaaS, mostly non-related to our focus in this paper, such as security and data management. Researches like [36, 37] expose those and many other challenges to the future.

4. Conclusions and Future Research

This research work presented a brief survey, and a classification proposal, aiming at providing a scenario awareness related to parameters, such as multitenancy, auto-tuning and SLA maintenance. These parameters may improve considerably the performance and the resource management of DBaaS in cloud configurations taking into account the point of view from both users and providers. Performance and resource management are important questions that must be well addressed in order to have a fully optimized application running on a database, and when it comes to Database as a Service these questions are even more important, considering the power the cloud provides. Thus, this survey proposes a taxonomy that covers a few areas to clarify the readers and propose a summary of what has been done in academy to help service providers and customers to take full advantage of what the cloud can offer regarding resource management and performance.

References

[1] D. Puthal *et al*. Cloud Computing Features, Issues, and Challenges: A Big Picture. In *International Conference on Computational Intelligence and Networks (CINE)*, 2015, pp. 116-123.

[2] I. Hashem *et al*. The rise of "big data" on cloud computing: Review and open research issues. In *Information Systems*, Volume 47, January 2015, pp. 98-115.

[3] D. Agrawal *et al*. A. Big data and cloud computing: new wine or just new bottles?. In *Proceedings of the VLDB Endowment*, 2010, pp. 1647-1648.

[4] D. Jacobs *et al*. Ruminations on multi-tenant databases. In *Datenbanksysteme in Business, Technologie und Web (BTW)*, 2007, pp. 514-521.

[5] P. Xiong *et al*. Intelligent management of virtualized resources for database systems in cloud environment. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE)*. 2011, pp. 87-98.

[6] J. Rogers *et al*. A Generic Auto-Provisioning Framework for Cloud Databases. In *Proceedings of the 5th International Workshop on Self-Managing Database Systems (SMDB)*, 2010.

[7] W. Lang *et al*. Towards Multi-tenant Performance SLOs. In *IEEE 28th International Conference on Data Engineering*, 2012, pp. 702-713.

[8] J. Ortiz *et al*. Changing the Face of Database Cloud Services with Personalized Service Level Agreements. In *Proceedings of the Conference on Innovative Data system Research (CIDR)*, 2015.

[9] L. Wang *et al*. Scientific Cloud Computing: Early Definition and Experience. In '*10th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2008, pp. 825-830.

[10] W. O'Mullane *et al*. Batch is Back: CasJobs, Serving Multi-TB Data on the Web. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2005, pp. 33-40.

[11] C. Curino *et al.*. Relational Cloud: A Database Service for the Cloud. In *CIDR*, 2011, pp. 235-240.

[12] Z. Liu *et al*. PMAX: tenant placement in multitenant databases for profit maximization. In *Proceedings of the 16th International Conference on Extending Database Technology (EDBT)*, 2013. pp 442-453.

[13] T. Kiefer *et al*. Private Table Database Virtualization for DBaaS. In *Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, 2011, pp 328-329.

[14] C. Weissman *et al*. The design of the force.com multitenant internet application development platform. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD)*, 2009, pp. 889-896.

[15] J. Baker *et al*. Megastore: Providing Scalable, Highly Available Storage for Interactive Services. In *CIDR*, 2011, pp. 223-234.

[16] J. Ni *et al*. Adaptive Database Schema Design for Multi-Tenant Data Management. In *IEEE Transactions on Knowledge and Data Engineering*, 2014, pp. 2079-2093.

[17] C. Curino *et al*. DBSeer: Resource and Performance Prediction for Building a Next Generation Database Cloud. In *CIDR*, 2013.

[18] V. Narasayya *et al*. SQLVM: Performance Isolation in Multi-Tenant Relational Database-as-a-Service. In *CIDR*, 2013.

[19] V. Narasayya *et al*. Sharing buffer pool memory in multi-tenant relational database-as-a-service. In *Proc. VLDB Endow*, 2015, pp. 726-737.

[20] L. Zhao *et al*. A Framework for Consumer-Centric SLA Management of Cloud-Hosted Databases. In *IEEE Transactions on Services Computing*, 2015, pp. 534-549.

[21] S. Das *et al*. Albatross: lightweight elasticity in shared storage databases for the cloud using live data migration. In *Proc. VLDB Endow*, 2011, pp. 494-505.

[22] Y. Ran *et al*. SLA-driven dynamic resource provisioning for service provider in cloud computing. In *IEEE Globecom Workshops (GC Wkshps)*, 2013, pp. 408-413.

[23] V. *et al*. Bridging the tenant-provider gap in cloud services. In *Proceedings of the Third ACM Symposium on Cloud Computing (SoCC)*, 2012, 14 pages.

[24] F. Chong *et al*. Multi-Tenant Data Architecture [online]. In *https://msdn.microsoft.com/en-us/library/aa479086.aspx*. Accessed: October 2016.

[25] E. Cecchet *et al*. Dolly: virtualization-driven database provisioning for the cloud. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE)*, 2011, pp. 51-62.

[26] Amazon Relational Database Service (RDS) [online]. In *https://aws.amazon.com/pt/rds*. Accessed: October 2016.

[27] Microsoft Azure [online]. In *https://azure.microsoft.com*. Accessed: October 2016.

[28] D. Shasha *et al*. Database Tuning: Principles, Experiments, and Troubleshooting Techniques. In *The Morgan Kaufmann Series in Data Management Systems*. 2002.

[29] P. Xiong *et al*. 2011. ActiveSLA: a profit-oriented admission control framework for database-as-a-service providers. In *SOCC*, 2011, 14 pages.

[30] Y. Chi *et al*. iCBS: incremental cost-based scheduling under piecewise linear SLAs. In *Proc. VLDB Endow*, 2011, pp. 563-574.

[31] D. Stamatakis *et al*. SLA-driven workload management for cloud databases. In *International Conference on Data Engineering Workshops (ICDEW)*, 2014, pp. 178-181.

[32] J. Baliga *et al*. Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport. In *Proceedings of the IEEE*, 2011, pp. 149-167.

[33] E. Gomes *et al*. An Advance Reservation Mechanism to Enhance Throughput in an Opportunistic High Performance Computing Environment. In *IEEE 13th International Symposium on Network Computing and Applications (NCA)*, 2014, pp. 221-228.

[34] H. He. Virtual resource provision based on elastic reservation in cloud computing. In *International Journal of Networking and Virtual Organisations*, 2015, pp. 30-47.

[35] D. Funke *et al*. Towards truthful resource reservation in cloud computing. In *6th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, 2012, pp. 253-262.

[36] C. Abadi *et al.* The Beckman Report on Database Research2014. In *SIGMOD*, 2014, pp. 61-70.

[37] Q. Zhang *et al*. Cloud computing: state-of-the-art and research challenges. In *Journal of Internet Services and Applications*, 2010, pp. 7-18.