Tim Kiefer, Hendrik Schön, Dirk Habich, Wolfgang Lehner

A Query, a Minute: Evaluating Performance Isolation in Cloud Databases

Erstveröffentlichung in / First published in:

Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou, 01.-05.09.2014. Springer, S. 173–187. ISBN 978-3-319-15350-6.

DOI: http://dx.doi.org/10.1007/978-3-319-15350-6_11

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6.
http://dx.doi.org/10.1007/978-3-319-15350-6_11

A Query, a Minute: Evaluating Performance Isolation in Cloud Databases

Tim Kiefer(✉), Hendrik Schön, Dirk Habich, and Wolfgang Lehner

Database Technology Group, Technische Universität Dresden, Dresden, Germany
{tim.kiefer,hendrik.schon,dirk.habich,wolfgang.lehner}@tu-dresden.de

Abstract. Several cloud providers offer reltional databases as part of their portfolio. It is however not obvious how resource virtualization and sharing, which is inherent to cloud computing, influence performance and predictability of these cloud databases.

Cloud providers give little to no guarantees for consistent execution or isolation from other users. To evaluate the performance isolation capabilities of two commercial cloud databases, we ran a series of experiments over the course of a week (a query, a minute) and report variations in query response times. As a baseline, we ran the same experiments on a dedicated server in our data center. The results show that in the cloud single outliers are up to 31 times slower than the average. Additionally, one can see a point in time after which the average performance of all executed queries improves by 38 %.

1 Introduction

Cloud computing has been a hype for several years. Motivations for moving to the cloud range from high flexibility and scalability to low costs and a pay-as-you-go pricing model. From a provider's point of view, consolidation of applications on a shared infrastructure leads to increased infrastructure utilization and reduced operational costs.

Virtually all major vendors offer relational databases as part of their cloud ecosystem, e.g., Amazon Relational Data Services [1], Microsoft Azure SQL Databases [11], or Oracle Database Cloud Services [13]. A common use-case for a database in the cloud is as storage tier for a website or application running in the same cloud. Storing application data outside the infrastructure of the provider is often unfeasible or prohibitively expensive with respect to data transfers or performance. However, relational databases traditionally had strict performance requirements and users have certain expectations when it comes to database performance. The service provider has to balance his interest in a high degree of resource sharing (which leads to an economic use of the available resources) and the customers' interest in a predictable, yet cheap service.

We refer to any database in a system that offers flexible provisioning, a pay-as-you-go pricing model, and resource sharing (by means of resource virtualization and usually transparent to the user) as a *cloud database*. Cloud databases have a considerably shifted focus on requirements compared to classic relational

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

databases. Throughput, having been the number one metric in the past, is still of interest, though many applications that run in a cloud infrastructure do not have the highest throughput requirements. However, new quality measures like *predictability*, *scalability*, *fairness*, and *performance isolation* determine the way, a customer perceives a cloud database.

In this work, we concentrate on the problem of *performance isolation*. Performance isolation refers to the customer's requirement that his performance is not influenced by other customers' activities. Performance isolation directly affects predictability, i.e., whenever different users influence one another it leads to variation in query response times and hence to bad predictability.

Depending on the implementation of the cloud database system, performance isolation is hard to achieve. Moreover, performance isolation is a goal that conflicts with high resource utilization. Service providers acknowledge that and state, e.g. "Each physical machine can service many databases, and performance for each database can vary based on other activities on the physical hosting machine."[1] Providers in the past have made little promises with respect to performance or predictability. However, this seems to change and providers started to add performance guarantees to their products or product road maps.

The research community also showed interest in performance isolation on different levels. For example, Gupta et al. [4] and Krebs et al. [10] investigated performance isolation in Xen based systems using different applications including databases. Narasayya et al. [12] and Das et al. [3] worked on the problem of performance isolation in shared-process cloud database implementations and provided a prototype implementation of their solution.

The evaluation of performance isolation in commercial cloud database offerings is inherently difficult. Many aspects of the implementation, especially the placement of different cloud databases, are by design hidden from the user. It is (from the outside) not possible to force the co-location of different databases which would allow us to artificially generate concurrency to evaluate the performance isolation. Hence, the only way we see to evaluate a cloud database system from the outside is to consider it a black box and observe the behavior in different situations. Our approach is to generate a constant load over a long period of time and to "hope" for other users to generate concurrent load that ultimately influences our query execution times. More specifically, we query the database every minute over a period of seven days and observe the variation of the response times for the query. In our experiments, we compare two different commercial cloud database providers with a baseline collected on a dedicated server.

To summarize, our key contributions in this paper are:

- An analysis of cloud database implementation options and their performance isolation challenges.
- An overview of currently available commercial cloud offerings.
- An experimental comparison of two commercial cloud databases with respect to performance isolation.

[1] From the Microsoft Azure documentation at http://msdn.microsoft.com/en-us/library/azure/dn338083.aspx.

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

The rest of the paper is organized as follows. Section 2 discusses performance isolation in different cloud database implementation classes. Section 3 continues with an overview of currently available commercial cloud offerings. Our experiments are detailed in Sect. 4 before we discuss related work and conclude our work in Sects. 5 and 6, respectively.

2 Performance Isolation in Database Clouds

In this section, we discuss the problem of performance isolation in general. Furthermore, we analyze possible cloud database implementation classes and the challenges for performance isolation related to each class.

2.1 Design Decisions

The degree of performance isolation in a cloud database system is a design decision for the service provider to make. Independent of the ability to implement it in the given system, the desired degree of performance isolation is not obvious.

Offering *strong isolation* leads to better predictability of the database performance. Resources are constantly assigned to users to ensure consistent behavior. At the same time, assigned resources that are currently not used by a certain user cannot be given to other users or else there is high chance of interference and degraded performance. Consequently, resources are often idle and the global utilization in the system is bad.

In contrast, designing for *weak isolation* gives the service provider the freedom to assign idle resources to active users, potentially above the amount they are actually paying for. Shared infrastructures in other domains deal with such higher assignments with a notion of *bonus resources* to indicate that the performance is at times better than the booked service level. With weak isolation, systems can also be oversubscribed to further increase utilization and lower service prices. However, depending on the global load, the performance that a single user observes may be unpredictable.

From personal communication with one of the cloud database providers, we know that customers seem to value predictability higher than bonus resources or cheap service. However, whether or not there is a best decision on performance isolation is up for debate.

2.2 Cloud Database Implementation Classes

The layered system stack of a DBMS—from the database schema to the operating system—allows for consolidation at different levels. Previous works have classified virtualization schemes, leading to classes like *private OS*, *private process*, *private database*, or *private schema* [5,7]. The classes differ in the layer that is virtualized and consequently in the resources that are private to a user or shared among several users. Figure 1 shows four possible implementation classes for cloud databases.

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

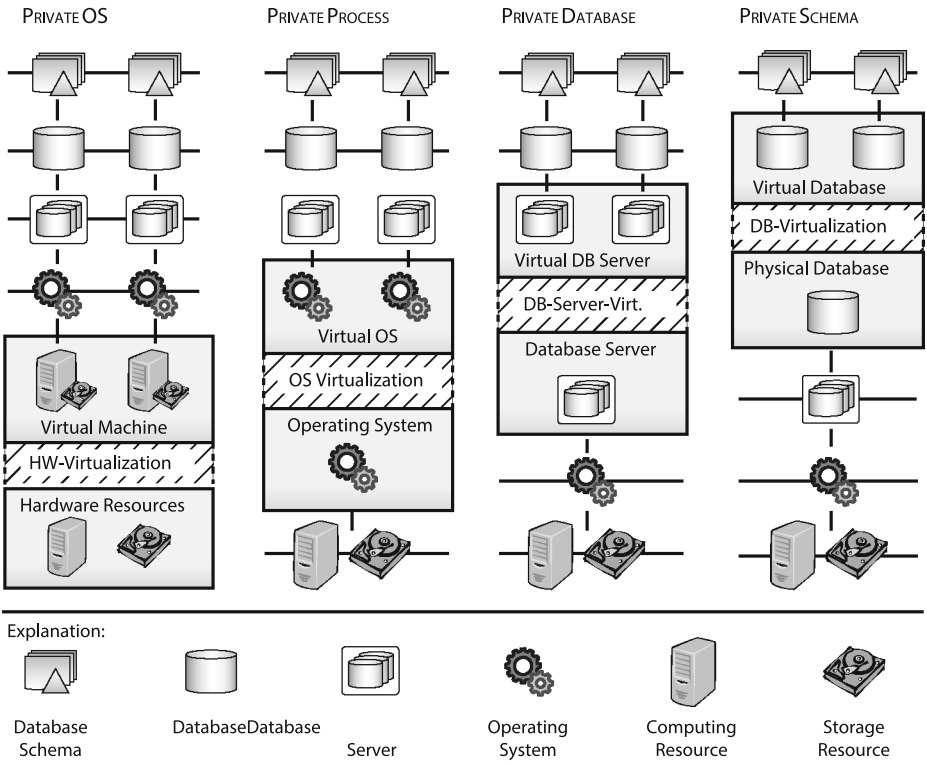


Fig. 1. Cloud database implementation classes

The means as well as the extent of performance isolation depend on the chosen implementation class. We differentiate different kinds of resources, *system resources* and *DBMS resources*. System resources are, e.g., CPU, main memory, storage, or network. DBMS resources are, e.g., page buffers, sort buffers, logging facilities, or locking facilities. Although DBMS resources ultimately map to system resources, their isolation and predictability depend on the ability to assign access to them and to prevent congestion on them.

Private Operating System: The system shown in Fig. 1 on the left implements a private OS virtualization where each user is given a complete stack of virtual machine, operating system, and database management system. The virtual machine hypervisor is responsible for balancing and isolating multiple virtual machines that run database workloads.

The private OS implementation class offers strong isolation of the system resources. The virtual machine hypervisor can be used to assign virtual CPU cores, main memory, storage capacity, and network bandwidth to each virtual machine. Depending on the system setup (disk setup), storage performance may or may not be isolated. Multiple virtual machines that access the same set of

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

hard disks may have quotas with respect to size, but they still rival for IOPS. Oversubscription of the system can be used to increase the system utilization, but may harm performance isolation. DBMS resources are strongly isolated in this class.

Private Process: The second system shown in Fig. 1 implements a private process scheme. In this implementation class, each user is given a private database server process and several such processes share the operating system.

In the private process implementation class, operating system facilities can be used to isolate certain system resources. The operating system scheduler assigns CPU time to the various competing processes and priority levels and sometimes quotas can be used to increase or decrease the share of any process. The main memory is per default not limited by the operating system. Each process can use the same virtual address space. The operating system only takes care of mapping the virtual memory to physical memory and of paging in case of memory shortage. In the Linux operating system, the `ulimit` system tool can be used to set system wide limits for certain resources (including memory). Similarly, a process can limit its resources with the system call `setrlimit`. Control groups (`cgroups`) are a Linux kernel feature to limit, police, and account the resource usage of certain process groups.

Storage and network resources are hard to isolate on the OS level. There are no integrated means in standard operating systems to assign, e.g., IOPS or network bandwidth, to processes. DBMS resources are per process and hence strongly isolated in this implementation class.

Private Database: The third system shown in Fig. 1 is a private database system. Here, a single server process hosts a number of private databases—a common setup for many database servers. The database management system needs to provision resources and balance loads between different databases, i.e., users. The different users are usually indistinguishable by the operating system.

In the private database class, all users share a database process and hence all system resources (from the operating system's point of view). The only way to isolate the users' performances is by means of managing and isolating DBMS resources. In our experience, buffers (e.g., for pages or sorts) can usually be split and assigned to different databases. Other resources, such as the logging facility are usually shared and can hence lead to congestion and inevitably weak performance isolation.

The detailed isolation options in currently available systems are dependent on the DBMS implementation and usually very limited. In the research community, Narasayya and Das et al. [3,12] investigated the problem of performance isolation in shared-process cloud database implementations. They presented *SQLVM*, an abstraction for performance isolation in the DBMS. Furthermore, they implemented and tested a prototype of SQLVM in Microsoft Azure.

Private Schema: The system shown in Fig. 1 on the right is an example for a private schema virtualization. Each user is implemented as a schema in a

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

single physical database. The performance isolation characteristics of the private schema class are similar to the private database class. However, some resources like page buffers may be even harder or impossible to isolate in a shared database.

3 Commercial Cloud Database Offerings

In this section, we present currently available commercial cloud database providers and their service characteristics and conditions. We analyzed three commercial database cloud offerings, Amazon Relational Data Services [1], Microsoft Azure SQL Databases [11], and Oracle Database Cloud Services [13]. The selection of these products is based on their availability and visibility but without intention of promoting any particular one. The various usage options and conditions are complex and detailed in the providers' documentations. In this work, we concentrate on a few key aspects such as pricing, resource provisioning, and (if available) performance guarantees. Though different in detail, the aspects we focus on are quite similar across the different service providers. Again, our intention is to provide an overview of available services, not to compare or rank them.

Amazon Web Services: Relational Database Service (RDS) is the part of Amazon Web Services that provides relational databases in the cloud. The user can select from four different database products (MySQL, PostgreSQL, Oracle, and Microsoft SQL Server). To fit the database performance to the application needs, users can select from different instance classes which differ in the provided number of virtual CPUs, the amount of memory, and the network performance. The available classes for MySQL instances in the region US East at this time are summarized in Table 1.[2] For high availability, a database can be deployed in multiple availability zones (Multi-AZ) so that there are a primary and a standby version for failover. Amazon guarantees 99.95 % monthly up time for Multi-AZ databases. Prices for the different database instances range from $0.025 per hour (db.t1.micro) to $7.56 per hour (db.r3.8xlarge Multi-AZ) (again, prices are exemplary for MySQL instances and the deployment region). The storage for the database can be as large as 3 TBs, where each GB is charged with $0.1 per month ($0.2 for Multi-AZ instances). Amazon offers the use of provisioned IOPS storage for fast and consistent performance at an additional cost. This provisioned storage can help to increase performance isolation and hence predictability.

Microsoft Azure: Microsoft offers its cloud ecosystem Azure with SQL Databases, a service to easily set up and manage relational databases. Databases can grow up to 150 GB but are charged based on their actual size, starting from $4.995 per month (up to 100 MB) and ranging to $225.78 per month for 150 GB. Microsoft maintains two synchronous copies in the same data center for failover. Geo-replication for further availability is a preview feature at this

[2] see http://aws.amazon.com/rds/pricing/.

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

Table 1. DB instance classes in RDS (exemplary for MySQL instances in region US East)

Instance type	vCPU	Memory [GB]	Network performance	Price/hour (Single-AZ) [$]	Price/hour (Multi-AZ) [$]
db.t1.micro	1	0.613	Very low	0.025	0.050
db.t1.small	1	1.7	Low	0.055	0.110
db.m3.medium	1	3.75	Moderate	0.090	0.180
db.m3.large	2	7.5	Moderate	0.185	0.370
db.m3.xlarge	4	15	Moderate	0.370	0.740
db.m3.2xlarge	8	30	High	0.740	1.480
db.r3.large	2	15	Moderate	0.240	0.480
db.r3.xlarge	4	30.5	Moderate	0.475	0.950
db.r3.2xlarge	8	61	High	0.945	1.890
db.r3.4xlarge	16	122	High	1.890	3.780
db.r3.8xlarge	32	244	10 gigabit	3.780	7.560

time. The Microsoft SQL Azure Service Level Agreement[3] contains so called Service Credit in case of a monthly uptime percentage below 99.9 %. As of now, Microsoft provides two different database classes, *web* as backend for websites and for testing and development purposes and *business* for production systems. Details about the configurations of both classes are not known. As a preview feature for future releases, Azure also contains additional classes *basic*, *standard*, and *premium*. With these classes, Microsoft introduces so called Database Throughput Units (DTU) that represent the performance of the database engine as a blended measure of CPU, memory, and read and write rates. It seems as if Microsoft is aiming for more predictable database performance and better performance isolation with DTUs and the new classes.

Oracle Cloud: Oracle's cloud ecosystem, the Oracle Cloud, provides several services including one for relational databases. A user can rent a schema in an Oracle 11 g instance. Thereby, the user selects between databases of up to 5 GB, up to 20 GB, or up to 50 GB. The prices range from $175 per month for 5 GB to $2000 for 50 GB. As a preview feature, users can rent virtual machines with fully configured (and optionally managed) Oracle database instances. As the only provider in our overview, Oracle lets users decide between different cloud database implementation class, i.e., Private Schema or Private OS.

4 Experimental Evaluation

In this section, we describe the setup and results of our experimental evaluation.

[3] see http://azure.microsoft.com/en-us/support/legal/sla/.

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

Two fundamental problems of evaluating cloud databases' performance isolation from a user's point of view are a lack of control and a lack of insight. A user can only control a database by means of starting, stopping, and using a database. Furthermore, it is possible to configure a database in the boundaries of what the service provider allows. It is however not possible to influence aspects of placement (beyond the selection of a region or data center) or co-location of several databases. The lack of insight refers to situations where a database's observed behavior changes. It is near impossible to reason about such changes without knowing the infrastructure and possible events that may have caused the change.

To overcome the problem of not being able to generate concurrency between cloud databases, we decided to run a steady workload for several days. Assuming that other users are actively using their cloud databases, we collect and report variations in execution times. These variations can have several reasons that are beyond our control and knowledge but are likely influenced by concurrency and the degree of performance isolation. As mentioned before, we can only speculate about reasons that may have caused certain changes in response times.

Since we are interested in performance isolation (and not absolute performance), we only report relative execution times. We also do not disclose which cloud database providers were used for our experiments but will refer to them as CloudA and CloudB.

4.1 Experiment Setup

Cloud Databases: We ran our experiments on databases from two different cloud providers. Additionally, we ran the same workload on a dedicated server in our data center as a baseline. The cloud databases were provisioned to fit the size of our data (1 GB). Beyond that requirement, we only used the most basic configuration and did not book any additional guaranteed performance (if available at all). We used the MulTe benchmark framework [8] to set up and fill our databases as well as to execute the workload. The workload driver that queries the cloud database was executed on a virtual machine in the cloud. We ensured that the virtual machine and the cloud database are in the same providers cloud and in the same region or data center.

Database Configuration: We used a TPC-H [16] database of scale factor 1 (equals 1 GB raw data) for our experiments. Primary key constraints as well as indexes that benefit the selected queries were created on the database. We did not modify the cloud database's configuration in any way.

Workload Configuration: As workload, we picked a subset of TPC-H queries, namely queries 2 (Minimum Cost Supplier), 13 (Customer Distribution), 17 (Small-Quantity-Order-Revenue), and 19 (Discounted Revenue). These queries were selected for their different characteristics and different execution times. The workload driver was configured to pick one of the four queries randomly and execute it. Afterward, the workload driver slept for one minute before it

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

started the next query. Since the execution times vary for different queries and cloud databases, we collected between 7632 and 9145 values over seven days.

Metric: The query execution times collected in our experiments are normalized to the average execution time (per query type and cloud database). Thereby, we are able to compare the two cloud databases although the absolute execution times differ. Additionally, we report the coefficient of variation (i.e., standard deviation divided by mean) and certain percentiles of the query execution times. We consider execution times below the 1st percentile and above the 99th percentile outliers and refer to the remaining execution times as being *without outliers*.

4.2 Result Discussion

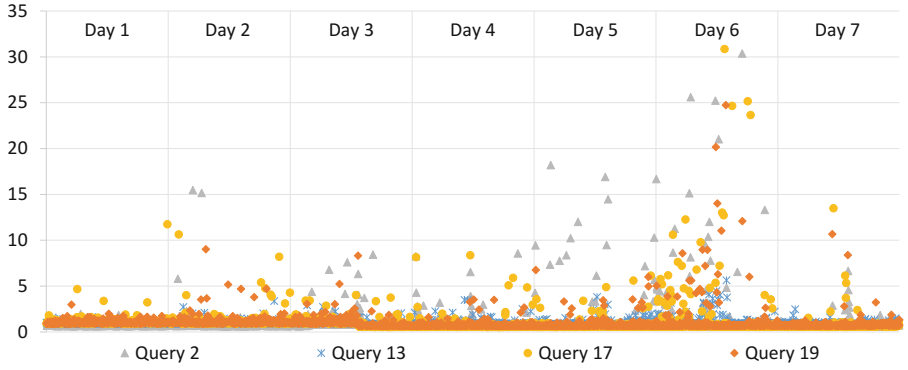
The relative execution times over seven days are plotted in Fig. 2. The charts provide a high-level first impression of the execution time variations. One can see that CloudA (Fig. 2a) has many queries that executed slowly compared to the average. Single queries needed as long as 31 times the average execution time of this query type. The figure also shows that there are more slow query executions on day 6 of the experiment compared to the other days. Figure 2b shows that query execution times in CloudB are closer around the average execution times. There are no execution times above 5 times the average. CloudB does not show any major changes of behavior over the duration of the experiment. Finally, Fig. 2c shows that our on-premise database has the least variations in execution times.

Table 2 lists the coefficient of variation of the query execution times. The impression that the on-premise database varies least is confirmed by the values. The table also shows that the variations are almost always higher in CloudA, except for Query 2 where the difference is negligible.

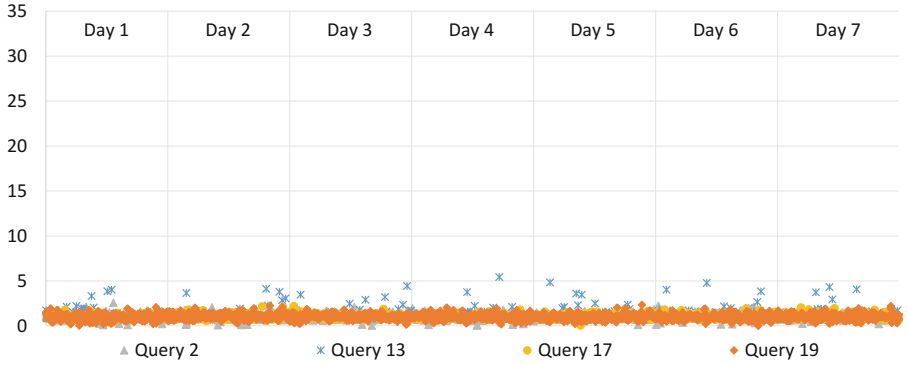
Table 3 lists the 90th, 95th, and 99th percentile of the relative query execution times. One can see that 95 % of all queries in CloudA finish within 1.46 times the average execution time. Accordingly, within 1.44 times the average in CloudB, and 1.27 times the average in the dedicated database. The high value of the 99th percentile in CloudA is caused by the few very long query executions.

Figures 3 and 4 show the relative query execution times (without outliers) of both cloud databases, split up by query and plotted in the range from 0 to 3. This more detailed plots shows several interesting things. First of all, there is no obvious daily rhythm visible. Even if the data centers are differently utilized at different times of the day, it does not show in our experiment results. Second, Fig. 3 shows that there must have been a distinct event in CloudA on day 3. After that event, all queries show an improved query execution time for the rest of the experiment (the execution time drops by 38 % for Query 17). Figure 3a gives the impression that there may have been more events on days 5 and 6 that increased the execution times for short periods of time. We have no way of knowing what caused the sudden change in performance, especially since we did not change the setup whatsoever. The third conclusion from Figs. 3 and 4 is that

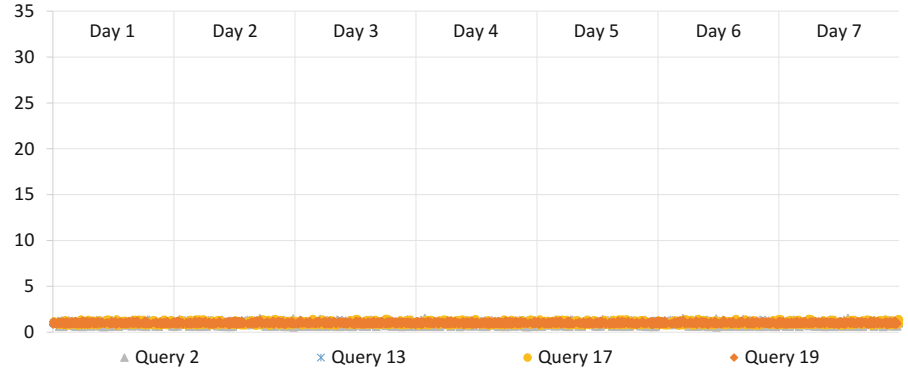
Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11



(a) CloudA



(b) CloudB



(c) On-Premise

Fig. 2. Experiment overview—relative execution times over seven days

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

Table 2. Coefficient of variation of query execution times (without outliers in parantheses)

	CloudA	CloudB	On-premise
Query 2	1.65 (0.81)	0.25 (0.22)	0.21 (0.20)
Query 13	0.31 (0.19)	0.36 (0.20)	0.01 (0.01)
Query 17	1.39 (0.59)	0.22 (0.19)	0.16 (0.15)
Query 19	1.04 (0.44)	0.29 (0.26)	0.11 (0.11)

Table 3. Percentiles of relative execution times over all queries

	CloudA	CloudB	On-premise
90th percentile	1.17	1.29	1.20
95th percentile	1.46	1.44	1.27
99th percentile	5.13	1.89	1.37

in CloudA the execution times show a rather distinct baseline with variance above that line while execution times in CloudB vary above and below the average.

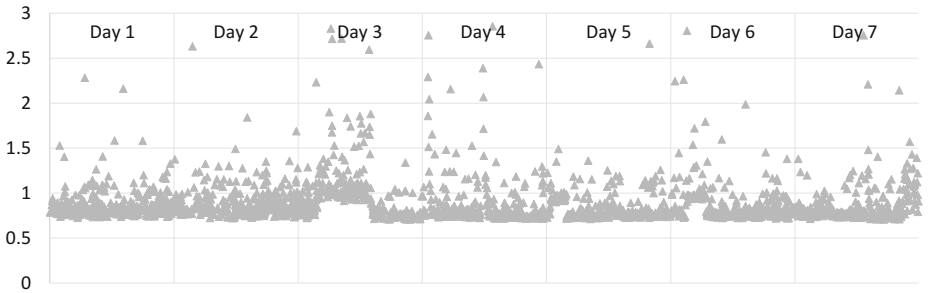
5 Related Work

Several works have investigated implementation and performance of cloud databases or performance isolation in general (not specific to databases).

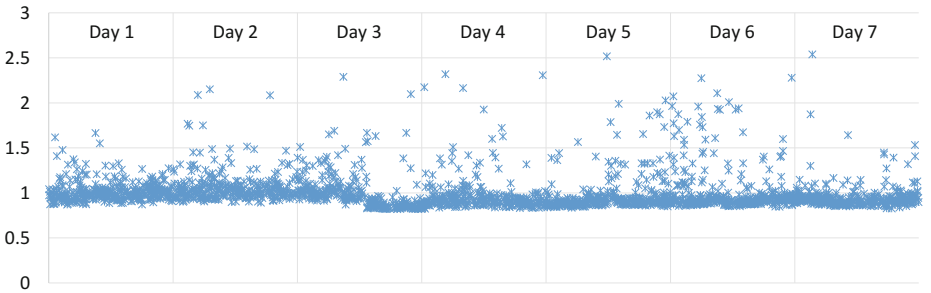
Kossmann et al. [9] analyzed different cloud databases when they were still very new and partly immature. They described different cloud database architectures and compared different providers with respect to performance, i.e., mainly scalability, and cost. Shue et al. [14] propose a system for per-tenant fair sharing and performance isolation in multi-tenant, key-value cloud storage services. However, key-value stores differ from relational databases and the results cannot easily be transferred between the two systems. Curino et al. [2] present *Relational Cloud*, a Database-as-a-Service for the cloud. Unlike commercial providers, the authors provide insights in their architecture and propose mechanisms for consolidation, partitioning, and security. While they try to consolidate databases such that service level objectives are met, performance isolation is not in the focus of their research.

Performance isolation in private OS systems has also been studied in the past. Somani and Chaudhary [15] investigated performance isolation in a cloud based on the Xen virtual machine monitor. They use different application benchmarks simultaneously to evaluate the isolation strategy provided by Xen. Gupta et al. [4] analyzed performance isolation in Xen based systems. Furthermore, they presented a set of primitives implemented in Xen to monitor per-VM and aggregated resource consumption as well as to limit the amount of consumed

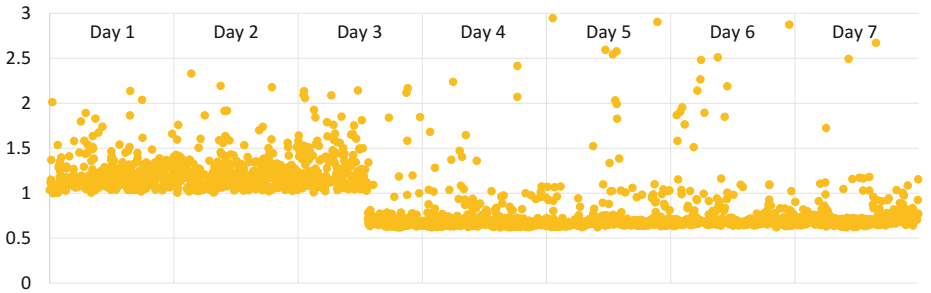
Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11



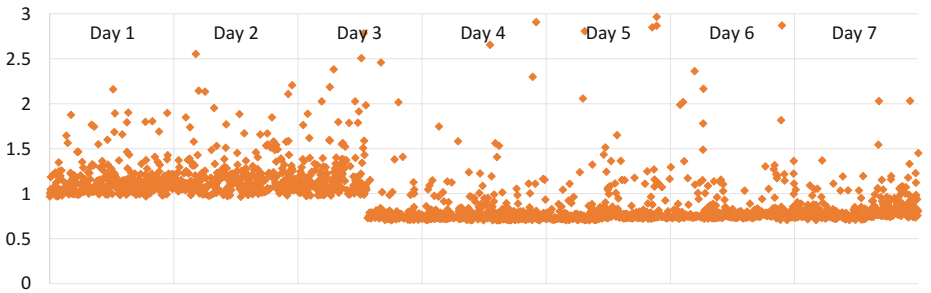
(a) Query 2, Minimum Cost Supplier



(b) Query 13, Customer Distribution



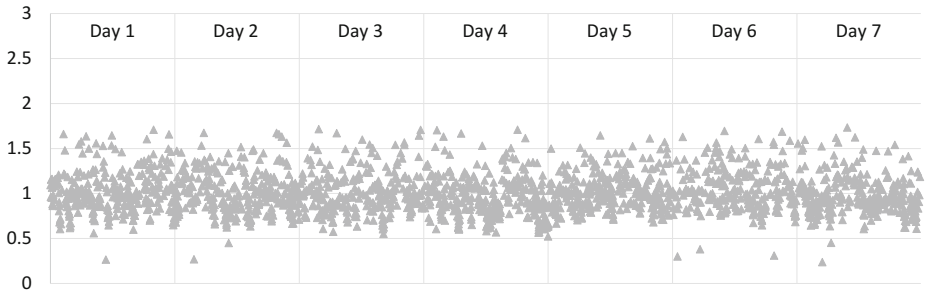
(c) Query 17, Small-Quantity-Order Revenue



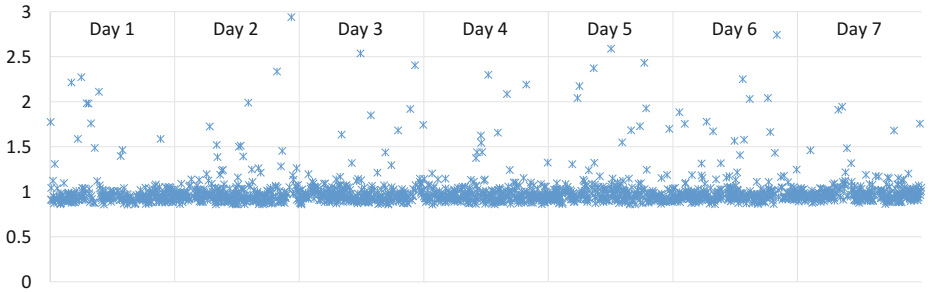
(d) Query 19, Discounted Revenue

Fig. 3. CloudA—relative execution times over seven days in the range from 0 to 3

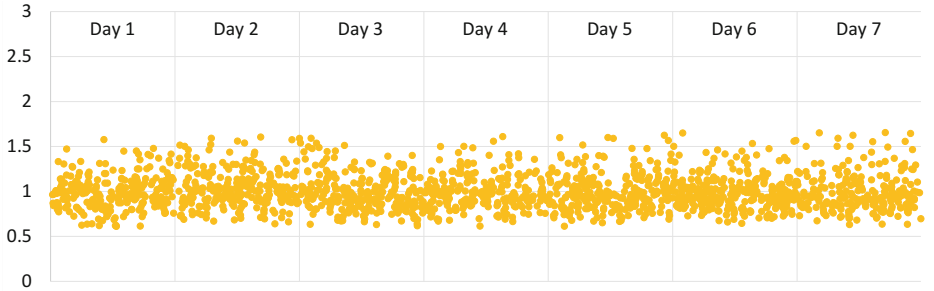
Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11



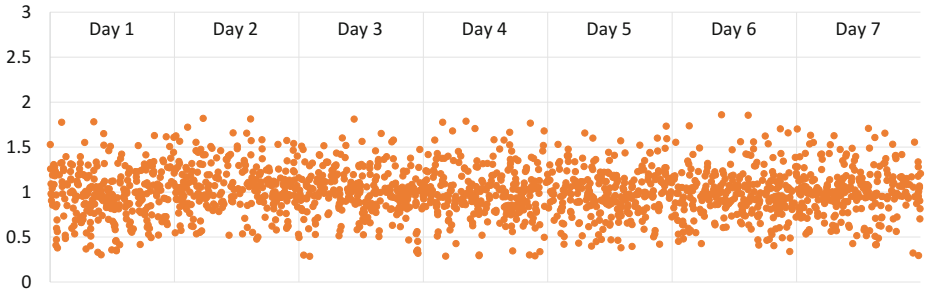
(a) Query 2, Minimum Cost Supplier



(b) Query 13, Customer Distribution



(c) Query 17, Small-Quantity-Order Revenue



(d) Query 19, Discounted Revenue

Fig. 4. CloudB—Relative Execution Times over seven days in the range from 0 to 3

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

resources. Krebs et al. [10] propose metrics for quantifying the performance isolation of cloud-based systems. Furthermore, they consider approaches to achieve performance isolation in Software-as-a-Service offerings. Their experimental evaluation uses several instances of the TPC-W benchmark in a controlled environment with a system running the Xen hypervisor.

Other works focus on performance isolation in private process systems, e.g., Kaldewey et al. [6] virtualize storage performance, a particularly hard resource to isolate. They used disk time utilization as the aspect of disk performance to focus on and implemented a prototype that uses utilization based I/O scheduling.

Finally, there is work in the area of shared process cloud systems. Narasayya et al. [12] investigated the problem of performance isolation in shared-process cloud database implementations. They presented SQLVM, an abstraction for performance isolation in the DBMS. Furthermore, they implemented and tested a prototype of SQLVM in Microsoft Azure. In [3], Das et al. further detail performance isolation in SQLVM with focus on the CPU as a key resource.

6 Summary

In this work, we gave an overview on performance isolation in cloud databases. We analyzed different implementation classes for cloud databases and the challenges on performance isolation that each class poses.

A black-box analysis of two commercial cloud databases gave us insights in their behavior. Our experiments revealed that variations in query execution times, which are influenced by the degree of performance isolation, differ in the two cloud databases. Moreover, we learned that both cloud databases showed constantly higher variations than our dedicated database.

References

1. Amazon: Amazon Relational Database Service (2014). http://aws.amazon.com/rds/
2. Curino, C., Jones, E.P.C., Popa, R.A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., Zeldovich, N.: Relational cloud: a Database-as-a-Service for the cloud. In: CIDR 2011, Asilomar, California, USA (2011). http://dspace.mit.edu/handle/1721.1/62241
3. Das, S., Narasayya, V., Li, F., Syamala, M.: CPU sharing techniques for performance isolation in multi-tenant relational Database-as-a-Service. In: VLDB 2014, Hangzhou, China, vol. 7 (2014). http://www.vldb.org/pvldb/vol7/p37-das.pdf
4. Gupta, D., Cherkasova, L., Gardner, R., Vahdat, A.: Enforcing performance isolation across virtual machines in Xen. In: van Steen, M., Henning, M. (eds.) Middleware 2006. LNCS, vol. 4290, pp. 342–362. Springer, Heidelberg (2006). http://link.springer.com/chapter/10.1007/11925071_18
5. Jacobs, D., Aulbach, S.: Ruminations on multi-tenant databases. In: BTW 2007, Aachen, Germany, pp. 5–9 (2007). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.6429&rep=rep1&type=pdf

Final edited form was published in "Performance Characterization and Benchmarking. Traditional to Big Data : 6th TPC Technology Conference, TPCTC 2014. Hangzhou 2014", S. 173–187, ISBN 978-3-319-15350-6. http://dx.doi.org/10.1007/978-3-319-15350-6_11

6. Kaldewey, T., Wong, T.M., Golding, R., Povzner, A., Brandt, S., Maltzahn, C.: Virtualizing disk performance. In: 2008 IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 319–330. IEEE, April 2008. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4550803
7. Kiefer, T., Lehner, W.: Private table database virtualization for DBaaS. In: UCC 2011, Melbourne, Australia, vol. 1, pp. 328–329. IEEE, December 2011. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6123516
8. Kiefer, T., Schlegel, B., Lehner, W.: MulTe: a multi-tenancy database benchmark framework. In: Nambiar, R., Poess, M. (eds.) TPCTC 2012. LNCS, vol. 7755, pp. 92–107. Springer, Heidelberg (2013). http://link.springer.com/chapter/10.1007%2F978-3-642-36727-4_7
9. Kossmann, D., Kraska, T., Loesing, S.: An evaluation of alternative architectures for transaction processing in the cloud. In: SIGMOD 2010 - Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, p. 579 (2010). http://portal.acm.org/citation.cfm?doid=1807167.1807231
10. Krebs, R., Momm, C., Kounev, S.: Metrics and techniques for quantifying performance isolation in cloud environments. Sci. Comput. Program. **90**, 116–134 (2014). http://linkinghub.elsevier.com/retrieve/pii/S0167642313001962
11. Microsoft: Microsoft Windows Azure (2014). http://www.windowsazure.com/en-us/
12. Narasayya, V., Das, S., Syamala, M., Chandramouli, B., Chaudhuri, S.: SQLVM: performance isolation in multi-tenant relational Database-as-a-Service. In: CIDR 2013 (2013)
13. Oracle: Oracle Database Cloud Service (2014). https://cloud.oracle.com/database?tabID=1383678914614
14. Shue, D., Freedman, M.J., Shaikh, A.: Performance isolation and fairness for multi-tenant cloud storage. In: OSDI 2012 (2012). https://www.usenix.org/system/files/conference/osdi12/osdi12-final-215.pdf
15. Somani, G., Chaudhary, S.: Application performance isolation in virtualization. In: CLOUD 2009, pp. 41–48. IEEE (2009). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5284105
16. TPC: Transaction Processing Performance Council, TPC-H (2014). http://www.tpc.org/tpch/