MuTeBench: Turning OLTP-Bench into a Multi-Tenancy Database Benchmark Framework

Andreas Göbel
Institute of Computer Science
Friedrich Schiller University Jena
Jena, Germany
Email: andreas.goebel@uni-jena.de

Abstract—**Database benchmarks have been used for decades to load test systems and to compare systems or system configurations with each other. However, their methods and assumptions are hardly suitable for multi-tenant cloud database services. Those systems have to provide both performance isolation of a lot of tenants with dynamic workloads and cloud computing features like scalability, elasticity, and reliability. In this article, the open source benchmark framework MuTeBench is presented. It allows the creation of OLTP benchmarks for multi-tenant databases and combines extensibility, portability, and evolved workload support of the underlying OLTP-Bench with flexible scheduling, statistic gathering across tenants, and individual service level agreements.**

***Keywords*-Benchmarking; Multi-Tenancy; OLTP; Database System; Service Level Agreements.**

I. Introduction

Centralization of infrastructure due to cloud computing is an increasingly important attempt of IT enterprises. Multi-tenant architectures enable cloud service providers to share resources and costs across various tenants (organizations, customers, or companies). Particularly, multi-tenancy database management systems (MT-DBMSs) have become an important field of research for academia and industry.

In MT-DBMSs, several or even all tenants share a single DBMS instance and its available resources. Assuming that not all tenants are active simultaneously, high resource utilization can be achieved by avoiding an allocation of resources required for the peak load of each tenant [1]. Consolidation can be implemented by three different approaches [2]. In the *shared machine* approach, each tenant receives a dedicated database resulting in high tenant isolation at the expense of high costs per tenant. In the *shared process* approach, tenants share databases but operate on separate tables or separate schemas. This approach enables partial resource sharing across tenants and allows an appropriate isolation level. The *shared table* approach achieves the highest degree of sharing and efficiency by sharing tables and indexes among tenants. A special column associates each row with the appropriate tenant. Allowing customized database schemas and individual administration for each tenant is very challenging with this approach.

Resource competition of simultaneously active tenants bears a new challenge for DBMSs, in particular with high degree of resource sharing. The performance of tenants must not be affected by resource-intensive activities and volatile workloads of other tenants, for example, in order to meet performance service level agreements (SLAs) of tenants. Moreover, tenant data has to be protected against unauthorized access by other tenants and a MT-DBMS must provide tenant metering, low operating costs and tenant-specific database schemas. These requirements are supplemented by providing general cloud computing features such as zero downtime, elasticity, and scalability. For scalability reasons, a MT-DBMS should run on low cost commodity hardware and scale out to a lot of servers for many customers.

Classical benchmarks are not able to adequately assess MT-DBMSs with respect to the above-mentioned requirements. A new generation of database benchmarks is required, which is suitable for the difficult terrain of clouds and multi-tenancy. In this article, appropriate methods and metrics of MT-DBMS benchmarks are summarized. The purpose, architecture, and configuration of the benchmark framework MuTeBench are presented. By some experiments, its suitability to evaluate MT-DBMSs concerning their major challenges is illustrated.

This article is structured as follows. Section II outlines challenges in benchmarking of MT-DBMSs and compares them against conventional DBMSs. Section III presents MuTeBench, a framework for creating MT-DBMS benchmarks with evolving tenant workloads. After discussing experiments in Section IV and related work in Section V, the article ends with conclusions of our contributions and open issues in Section VI.

II. MT-DBMS Benchmarking

The best known representative of benchmarks for transaction processing systems and databases are benchmarks of the Transaction Processing Performance Council (TPC) [3], which simulate real-world application scenarios. Like most traditional benchmarks, they provide an infrastructure to run a representative workload against a static non changing software system in order to assess its average performance under maximum load. Furthermore, some benchmarks include cost-based metrics. The static setups of those traditional database benchmarks contradicts to MT-DBMSs which have to handle a varying number of active tenants with changeable workload mixes and rates as described in [4]. For this purpose, they may need to allocate additional hardware or save costs by releasing underutilized resources. Therefore, evaluating MT-DBMSs requires benchmarks with the ability to run changing workloads of several tenants in parallel. Such benchmarks can be used by service providers to improve their services.

Under certain circumstances, they may also assist customers in finding an optimal database service.

A. Metrics

Appropriate metrics are needed to assess the impact of other active tenants on the query processing performance of a single tenant. Typical performance metrics, such as request throughput, average latency, or latency percentiles of a single tenant are only partially suited for this purpose. We call them *absolute tenant performance metrics*.

Kiefer et al. [5] proposed a metric called relative execution time. We have adopted this approach and expanded it by relative throughput, relative average latency, and relative latency percentiles, summarized as *relative tenant performance metrics*. For their calculation, the best possible tenant performance in a simulated single-tenant environment has to be determined first. For this purpose, tenants may run workload in an initial baseline run without any resource competitors. Following this, during actual multi-tenancy runs their absolute performance will be gathered. According to formulas 1 and 2, these results are set in proportion to their baseline equivalents relatively, resulting in relative performance values.

$$throughput_{rel} = throughput_{abs}/throughput_{base} \quad (1)$$
$$latency_{rel} = latency_{base}/latency_{abs} \quad (2)$$

Performance SLAs and contract penalties due to non-fulfillment of them are not common in DBMSs and cloud database services yet [6]. They would cause a MT-DBMS to prioritize tenants, statically or even dynamically. The metric *SLA compliance* can be used to determine MT-DBMS capabilities concerning performance reliability and tenant prioritization. It is calculated as total penalty amount.

B. Experiments

The proposed metrics in conjunction with scheduling of evolved tenant workloads enable an evaluation of the main challenges of MT-DBMS by various kinds of experiments.

Scalability tests: These tests refer to the MT-DBMS performance with increasing load. On the one hand, single-tenant scalability can be measured by increasing a tenant's workload rate in a simulated single-tenant environment and gathering its absolute performance. On the other hand, system scalability can be quantified by continuously increasing the number of active tenants with constant workload rates and mixes. On closer consideration of the performance degradations of active tenants, the fairness of resource distribution can be measured.

Performance isolation tests: These tests estimate how well a MT-DBMS isolates tenant performances from each other. This can be achieved by calculating relative performance of a tenant which runs a static workload while one or more other tenants run a dynamic workload in parallel. Dynamic workloads can be achieved either by changing the transaction rate or its mix over time. Tenants may have equal importance or a MT-DBMS prioritizes them by agreed individual SLAs. Relative tenant performance can be used to assess resource allocation fairness of the MT-DBMS, while SLA compliance evaluates its performance reliability.

Further tests may determine database elasticity like warmup times on workload changes. They can also be used to evaluate database robustness on hardware failures or include costs from the perspective of a provider (cost per tenant) or a tenant (cost of delivered performance, etc.). Detailed explanations of these tests are beyond the scope of this article.

III. MuTeBench

MuTeBench [7] is an open-source framework that allows simple creation of highly diverse benchmarks for a variety of multi-tenant DBMSs and cloud database services. We have implemented it with the purpose of running scalability tests and performance isolation tests, but it is not limited to these experiments only. It provides flexible scheduling of various tenant workloads implementing diverse and evolving usage patterns. With the help of fine-grained statistic gathering, all metrics mentioned in Section II-A can be determined.

Instead of developing a new framework, we decided to extend an existing testbed called OLTP-Bench [8]. We will reason why we regard it as an ideal starting point for a MT-DBMS benchmark framework and point out our extensions resulting in MuTeBench.

A. OLTP-Bench

OLTP-Bench is an open-source benchmarking framework for relational databases. Currently it supports data generation and workload execution of 15 online transaction processing (OLTP) benchmarks consisting of classical OLTP benchmarks such as TPC-C [3], modern web benchmarks like Yahoo Cloud Serving Benchmark (YCSB) [9], generated synthetic micro-benchmarks as well as workload traces of real-world applications like Twitter. Due to central SQL dialect management and the use of standard database drivers, each benchmark can be applied to all major relational DBMSs and cloud database services. OLTP-Bench is able to simulate evolving usage patterns by varying its transaction rate and mix. It is determined in a configuration file in conjunction with connection settings and the number of concurrent worker threads. Controlled by a central workload manager, those threads execute requests in parallel and gather transaction latencies. The results of all workers are finally combined and aggregated for a given time window, providing information about average latency, latency percentiles, and throughput. This client-side database performance monitoring can be brought into accordance with server-side monitoring of its resource consumption. However, OLTP-Bench is not suitable for modeling a challenging large-scale multi-tenancy scenario, among other reasons, because it cannot run a benchmark several times in parallel. [8]

B. Architecture

Due to its features mentioned in Section III-A, OLTP-Bench represents an appropriate benchmarking framework for cloud databases. Hence, we decided to purposefully modify and expand it in order to allow benchmarking of MT-DBMSs. Figure 1 illustrates the resulting architecture based on [8], with added components marked in gray. To simplify updates on future OLTP-Bench versions, we changed its components as little as possible. The most impactful change is an added central controller. It schedules benchmark runs for all tenants according to a scenario description file (see Section III-C). For each benchmark run, it controls existing OLTP-Bench components which are necessary for running a workload or modifying data. The most notable changes of these components are related to enabling concurrent benchmark runs,

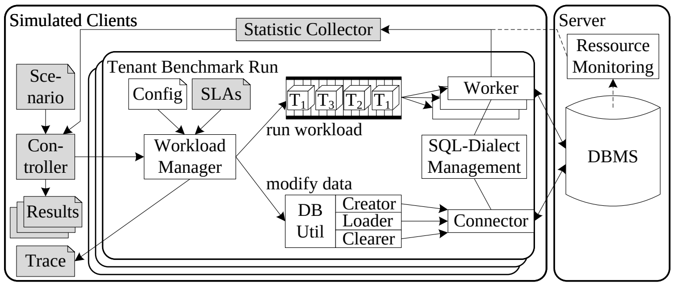


Figure 1: Architecture of MuTeBench

```
 1 <event> <!-- Time Format: hh:mm:ss -->
 2     <start>00:00:00</start>
 3     <repeat>00:05:00</repeat>
 4     <stopAfter>00:10:00</stopAfter>
 5     <tenantsPerExecution>3</tenantsPerExecution>
 6     <firstTenantID>2</firstTenantID>
 7     <benchmark>tpcc</benchmark>
 8     <actions>create, load, execute</actions>
 9     <configFile>tpcc_config.xml</configFile>
10     <slaFile>premium_service.xml</slaFile>
11 </event>
```

Figure 2: Scenario Event Definition

supporting service level agreements, calculating statistics after finishing all benchmark runs, and calculating them both for each tenant and across tenants. Statistics are stored as files and visualized by charts using a graphical user interface (GUI) if requested (see Section III-D).

C. Scenario Description

A scenario description file contains a number of events whose definition is shown in Figure 2. Each event is comparable to an OLTP-Bench run and can be executed by multiple tenants at different times. Event execution times are specified by time of first execution, time of last execution and a repetition time interval (lines 2–4). At any execution time, the benchmark may be performed for several tenants. The associated tenant ID will be incremented for each run, starting from an initial ID (lines 5–6). By Figure 2, benchmark runs for nine tenants will be started. The tenants 2–4 will start immediately, 5–7 after 5 minutes, and 8–10 after 10 minutes. Actions to be executed (table definition, data generation, benchmark execution, script execution, or table truncating) and an universal OLTP-Bench configuration file for the given benchmark have to be defined (lines 7–9). A single configuration file enables tenant specific settings by using wildcards, which will be replaced by the corresponding tenant ID. This enables, for instance, individual workload rates or custom connection settings to support shared machine, shared process, and shared table consolidation (see Section I). Combined with fain-grained workload control of OLTP-Bench, this scheduling flexibility allows running quite diverse large-scale experiments, despite compact definition.

D. Statistics and Service Level Agreements

MuTeBench collects statistics about the absolute performance of each workload run. This is based on OLTP-Bench statistic gathering and includes both performance of each single tenant and overall performance. In addition, relative performance (see Section II-A) may be determined by involving result files of previous tenant baseline runs. Results can be saved as raw data (list of transactions including start times and latencies) and aggregated data by using a given time interval. Furthermore, MuTeBench provides a GUI based on Java Swing and JFreeChart [10] to visualize results as charts.

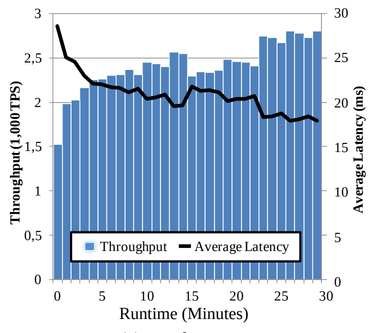
MuTeBench supports tenant-specific SLAs (line 10 in Figure 2). Each agreement is defined for an absolute performance metric (see Section II-A) and a time window like five minutes. Optionally, those agreements can be linked to specific transaction types only. Each agreement may include several service levels, which associate violations of performance targets with a penalty. For example, a service level may predefine a penalty of US$50 if the 99th latency percentile for 'Delivery' transactions is above 50 ms within an interval of five minutes. MuTeBench is able to measure DBMS reliability very conveniently by computing penalty amount for a given tenant or across all tenants. Because of a lack of standards and only marginal DBMS support of performance SLAs, we have developed a SQL extension to forward SLAs to our MT-DBMS prototype for further research purposes.

IV. Experiments

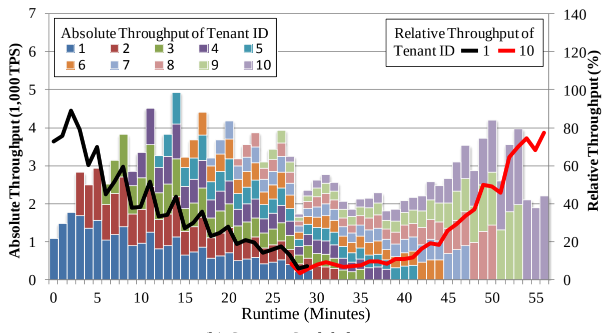
We have performed several experiments to analyze the suitability of MuTeBench for creating various MT-DBMS benchmarks. This article presents the results of two selected experiments using just two machines. Further tests may evaluate database clusters, compare different MT-DBMSs, consider creation, migration, and deletion of tenant data, or evaluate database predictability by using SLA compliance. We used computers of identical construction (Intel Core i7-2600 with 4 cores running at 3.4 GHz, 8 GB of memory, a 7200 rpm hard disk, Debian 4.7.2-4) for client and server, they were interconnected by 1 Gbps Ethernet. For each tenant, 50 parallel worker threads with separate database connections were executed running the YCSB benchmark [9] against MySQL 5.5.31. Due to a shared machine approach, each tenant uses a dedicated database with a size of about 600 MB (YCSB scale factor 500).

During an initial baseline run, a single tenant ran a YCSB workload mix (50% 'ReadRecord' and 10% of each other transaction type) without any rate limitations for 30 minutes. After initial overhead for connection buildup, the performance increased almost steadily due to improved buffer utilization. Figure 3a illustrates the results of this experiment with a maximum performance of about 2,800 transactions per second (TPS) and an average latency of 17 milliseconds at this time.

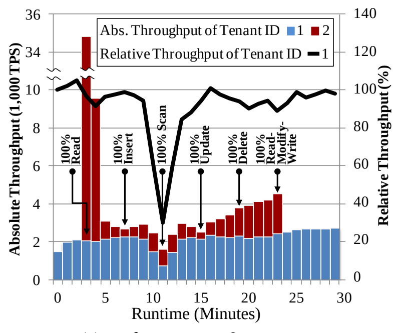
After re-establishing equivalent test conditions, we have performed a system scalability test with 10 tenants, which used the workload profile of the baseline run. They were run one by one at a starting interval of three minutes. The absolute throughput increased up to 4,900 TPS with five active tenants, respectively 250 parallel connections (see Figure 3b). However, the performance decreased with increasing number of connections. The situation was aggravated by exhausting the available memory after 28 minutes. It relaxed again by less active connections near the end of the test. The relative throughput of tenant 1 highlights periodical performance impact by incoming connections of other tenants. Altogether, MySQL distributed resources quite fair among tenants. Performance deviations of active tenants were relatively small at each point in time.



(a) Baseline Run



(b) System Scalability Test



(c) Performance Isolation Test

Figure 3: Experiment results

Another experiment shall give further explanation about tenant performance isolation. Tenant 1 used the workload profile of the baseline run once again, while tenant 2 started an evolving workload mix with unlimited rate after three minutes. At the beginning it only ran the transaction 'ReadRecord' and shifted to the next transaction type by changing the ratio (100:0 $\rightarrow$ 75:25 $\rightarrow$ 50:50 $\rightarrow$ 25:75 $\rightarrow$ 0:100) every minute. This shift was repeated for all YCSB transaction types. Figure 3c shows the tremendous throughput variations of tenant 2. Surprisingly, the achieved 32,758 point queries per second over one minute barely affected the performance of tenant 1. Only range queries resulted in a significant performance impact of tenant 1.

V. Related Work

To the best of our knowledge, *Multe* [5] is the only existing benchmark framework for MT-DBMSs so far. Its purpose is similar to MuTeBench. However, it was not designed for OLTP benchmarks exclusively. It is suitable for real workload simulation just to a limited extent because workload mixes of tenants cannot be changed dynamically and their workload can only be enabled or disabled instead of providing fine-grained rate control. In addition, its statistic gathering is limited and it provides only a sample implementation of a single benchmark for two supported DBMSs so far.

Aulbach et al. [2] presented a MT-DBMS benchmark called *MTD Benchmark*. It simulates an OLTP component of a hosted customer relationship management offering and has been primarily designed to compare the performance of different tenant consolidation approaches by providing schema variability. Hence, its purpose differs from benchmarks built by MuTeBench.

Krebs et al. [11] created a multi-tenancy benchmark to compare multi-tenancy and tenant isolation for dedicated virtual machines on cloud platforms based on TPC-W [3]. Therefore, they address a complete service infrastructure, consisting of web servers, application servers, and database servers.

TPC-VMS [3] requires parallel executions of identical workloads in separate virtual machines, consolidated onto one logical server. It describes an environment with static tenant workloads. By contrast, an intermediate state of TPC-V [12] describes a similar scenario, but with evolving usage patterns and database sizes of tenants. Both benchmarks specify workloads, scenarios, and their environment precisely in order to ensure comparability, while MuTeBench has been designed to perform in a wide range of scenarios.

VI. Conclusion and Future Work

In this article, we summarized our expectations towards capabilities of a MT-DBMS benchmarking framework and reasoned that classical database benchmarks cannot fulfill them. In our opinion, OLTP-Bench represents an ideal basis for such a framework because of its extensibility, portability, statistic gathering, and support of evolving workloads. Our benchmark framework MuTeBench benefits from that and combines it with flexible tenant workload scheduling, SLA support and new metrics to cope with challenges of MT-DBMSs.

However, MuTeBench has some limitations. For instance, compatibility with all OLTP-Bench features is not tested yet and its numerous parallel workers may limit its scalability. Hence, as an extension we plan to properly decompose a given scenario for distributed running a benchmark on several clients and combining their statistics. Transaction-specific SLAs are not yet evaluated and further tests with different database systems and cloud database services are required.

References

[1] D. Jacobs and S. Aulbach, "Ruminations on Multi-Tenant Databases," in BTW, 2007, pp. 514–521.

[2] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger, "Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques," in SIGMOD, 2008, pp. 1195–1206.

[3] Transaction Processing Performance Council, http://www.tpc.org/, retrieved: March 28, 2014.

[4] J. Schaffner and T. Januschowski, "Realistic Tenant Traces for Enterprise DBaaS," in ICDE Workshops, 2013, pp. 29–35.

[5] T. Kiefer, B. Schlegel, and W. Lehner, "MulTe: A Multi-Tenancy Database Benchmark Framework," in TPCTC, 2012, pp. 92–107.

[6] W. Lang, S. Shankar, J. M. Patel, and A. Kalhan, "Towards Multi-tenant Performance SLOs," in ICDE, 2012, pp. 702–713.

[7] MuTeBench, https://github.com/MuTeBench/MuTeBench, retrieved: March 28, 2014.

[8] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudré-Mauroux, "OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases," PVLDB, vol. 7, no. 4, 2014, pp. 277–288.

[9] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking Cloud Serving Systems with YCSB," in SoCC, 2010, pp. 143–154.

[10] JFreeChart, http://www.jfree.org/jfreechart/, retrieved: March 28, 2014.

[11] R. Krebs, A. Wert, and S. Kounev, "Multi-Tenancy Performance Benchmark for Web Application Platforms," in ICWE, 2013, pp. 424–438.

[12] P. Sethuraman and H. R. Taheri, "TPC-V: A Benchmark for Evaluating the Performance of Database Applications in Virtual Environments," in TPCTC, 2011, pp. 121–135.