A Cloud-centric Survey on Distributed Database Evaluation

Daniel Seybold and Jörg Domaschka
Ulm University, Institute of Information Resource Management, Ulm, Germany
{daniel.seybold, joerg.domaschka}@uni-ulm.de

Abstract. The database landscape has significantly evolved over the last decade and distributed databases running in the cloud moved into the focus. This evolvement challenges the already non-trivial task of selecting and deploying a distributed database system. Database evaluation frameworks aim at easing this task by guiding the database selection and deployment decision. The evaluation of databases has evolved as well, now considering not only performance evaluation but also distributed database aspects such as scalability or elasticity. This paper presents a classification for distributed database evaluation frameworks based on evaluation tiers and framework requirements with the focus on exploiting cloud computing. The classification is applied for eight well adopted evaluation frameworks. The results points out that the evaluation tiers performance, scalability, elasticity and consistency are well supported, while (cloud) resource selection and availability lack support. Further, the analysed frameworks support common database evaluation requirements in varying extend and lack the support of cloud-centric requirements.

Keywords: NoSQL, benchmarking, distributed database, database evaluation, cloud databases

1 Introduction

Relational database management systems (RDBMS) have been the common choice for persisting data for many decades. While traditional RDBMS installations can be considered as static in terms of one-RDBMS-installation-fits-all or even enterprise-based database appliances, including not only the RDBMS software but also the hardware to run the RDBMS.

Yet, the database landscape has evolved over the last decade and a plethora of new database systems came into the focus of academia and industry, namely NoSQL and NewSQL databases [18], promising persistence solutions not only for web-oriented applications but also for new application domains such as “Big-Data” or “IoT”. These database systems are designed to provide common requirements such as high performance or scalability by running on commodity

* A shorter version of this paper appeared in the 21. European Conference on Advances in Databases and Information Systems, ADBIS, Nicosia, Cyprus, September 2017 [24].
hardware and are typically built upon a distributed architecture, enabling the
dynamic adaptation of the database systems size.

An enabler of the database system evolution is the cloud computing paradigm,
providing easy access to commodity hardware by on-demand, self-service re-
source provisioning [20]. As resources can be elastically provisioned [20], the
deployment and adaptation of distributed database systems can be eased by
cloud resources.

With the plethora of available NoSQL and NewSQL systems and the variety
of cloud resource offerings, the decision which database to choose and how to
deploy the database in the cloud is a challenging task. In this respect, database
evaluation is a common approach to guide these decisions. With the evolution
of the database systems, the landscape of database evaluation frameworks has
evolved as well: from single node database evaluation, e.g. TPC-W 1 of the
Transaction Processing Performance Council (TPC), to distributed database
systems evaluation, e.g. the Yahoo Cloud Serving Benchmark (YCSB) [9], adding
new evaluation tiers such as scalability, elasticity or consistency. Yet, available
evaluation frameworks aim at different evaluation tiers and differ significantly
towards common database evaluation framework requirements [17][7], especially
with respect to cloud computing.

In order to facilitate the selection and deployment of distributed databases in
the cloud, we present a classification for distributed database evaluation frame-
worxks with the focus on cloud resources. Our contribution is threefold, (1) we
define relevant evaluation tiers for distributed databases running the cloud; (2)
we extend common requirements towards database evaluation with cloud spe-
cific requirements; (3) we classify common evaluation frameworks based on the
evaluation tiers and evaluation requirements.

The remainder of this paper is structured as follows: Section 2 introduces the
background on distributed databases and database evaluation. Section 3 defines
the evaluation tiers while Section 4 defines the requirements towards evaluation
frameworks. Section 5 presents the classification of common frameworks and
Section 6 discusses the analysis results. Section 7 concludes.

2 Background

The evolution of distributed database systems has led to novel architectures and
storage models, which are presented in the following Section 2.1, while Section 2.2
presents the background on database and cloud-centric evaluation frameworks.

2.1 Distributed Databases in the Cloud

Distributed databases provide a single database system to the user which is
spread over multiple nodes. A single database instance as part of the overall
distributed database system is termed database node in the following. The overall
distributed database system is termed database cluster.

1 http://www.tpc.org/tpcw/
The architecture of distributed databases is typically categorised into the three distribution models [23]: a single server handles all read and write requests; in a master-slave distribution the data is replicated across multiple nodes. One node represents the designated master node, executing all write requests and synchronizing the slave nodes, executing only read requests. In a multi-master all nodes are equal, i.e. replication and sharding [23] of the data is applied to spread write and read requests across all nodes of the cluster.

With respect to the storage model, distributed databases can be classified into three top-level categories with additional sub-categories: relational, NoSQL and NewSQL data stores. Relational data stores target transactional workloads, providing strong consistency guarantees based on the ACID paradigm. Hence, relational data stores are originally designed as single server database systems, (e.g. MySQL\(^2\) or PostgreSQL\(^3\)), which have been extended lately to support distribution, e.g. MySQL Cluster\(^4\). NoSQL data stores follow the eventual consistency paradigm [29], providing weaker consistency guarantees but easing a distributed architecture and the support for features such as scalability and elasticity. NoSQL storage models can be further classified [23] into key-value stores (e.g. Riak\(^5\)), document-oriented stores (e.g. MongoDB\(^6\)), column-oriented stores (e.g. Apache HBase\(^7\)) and graph-oriented stores (e.g. Neo4j\(^8\)). The latest data store category, NewSQL (e.g. VoltDB\(^9\) or NuoDB\(^10\)), targets strong consistency in conjunction with a distributed architecture.

Nowadays, the Infrastructure as a Service (IaaS) cloud service model is the preferable way to deploy database cluster, which require a high degree of flexibility on the resource level, as IaaS offers on-demand self-service and rapid elasticity [20]. IaaS provides processing, storage, networks resources to run arbitrary software [20]. The processing and storage resources are typically encapsulated in a virtual machine (VM), including as well the operating system (OS). VMs run on the virtualised physical infrastructure of the IaaS provider.

### 2.2 Evaluation Frameworks

Evaluating databases imposes challenges for the evaluation frameworks itself, which are discussed over decades. A first, but still valid guideline for evaluating relational database systems is presented by Gray [17], defining the requirements for: relevance to an application domain, portability to allow benchmarking of different systems, scalability to support benchmarking large systems, and simplicity to ensure that the results are easy to understand. A more recent guideline

\(^2\) [https://www.mysql.com/](https://www.mysql.com/)
\(^3\) [https://www.postgresql.org/](https://www.postgresql.org/)
\(^4\) [https://www.mysql.com/products/cluster/](https://www.mysql.com/products/cluster/)
\(^6\) [https://www.mongodb.com](https://www.mongodb.com)
\(^7\) [https://hbase.apache.org/](https://hbase.apache.org/)
\(^8\) [https://neo4j.com/](https://neo4j.com/)
\(^9\) [https://www.nuodb.com/](https://www.nuodb.com/)
\(^10\) [https://www.voltdb.com/](https://www.voltdb.com/)
is provided by Bermbach et al. [7], adding the context of distributed databases and the resulting challenges for supporting different deployment topologies and coordination of distributed experiments.

By adopting these challenges, several distributed database evaluation frameworks have been established over the years, which are analysed in Section 5. Yet, comprehensive summaries and classifications of database evaluation frameworks are rare, especially with respect to the distributed database and cloud aspects.

An overview of existing evaluation frameworks is presented by Friedrich et al. [14], focusing on the tiers availability and consistency. Yet, general requirements for database evaluation frameworks are not introduced and cloud specific characteristics are not considered.

Evaluating the dimensions of consistency in distributed databases, is also analysed by Bermbach et al. [6], introducing client-centric and data-centric consistency metrics. Related evaluation frameworks for consistency are presented and missing features for fine-grained consistency evaluation are outlined. Yet, no additional evaluation tiers are considered.

Khazaei et al. [19] present a recommendation compendium for distributed database selection based on functional and non-functional requirements, including an overview of database evaluation frameworks. Yet, the framework overview considers only the performance tier without considering cloud resources.

Besides database-centric evaluation frameworks, there are frameworks for evaluating multi-tier distributed applications, including databases. The RUBiS benchmark [8] evaluates a 3-tier application by simulating web-based shopping applications with the focus on the scalability tier. A more cloud-centric evaluation framework is Cloudstone [26], comprising a 3-tier web application and a workload generator. As the focus of this framework relies on cloud applications it provides tools for the automated deployment, execution and monitoring of the evaluation experiments. Yet, CloudStone does not focus on database specific evaluation tiers as its focus relies on cost metrics. Certainly, both frameworks provide valuable workloads and methodologies, but rely on single node RDBMS and do not consider distributed databases.

3 Distributed Database Evaluation Tiers

With the evolving heterogeneity in distributed database systems their evaluation becomes as well more challenging and heterogeneous. Typically, database evaluation is driven by a workload domains (WD), such as transactional workloads [11], web-oriented workloads [11], Big Data workloads [15] or synthetic workloads [9]. These workload categories provide a brief overview and can be further classified and extended, which is not in the scope of this paper.

In conjunction with the various workload domains, the need for new evaluation tiers for distributed databases arose. While performance is still an essential database evaluation tier, distributed databases require the consideration of additional evaluation tiers due to their architectures and storage models. In
the following common evaluation tiers of established database evaluation frameworks (cf. Section 5) and cloud-centric evaluation frameworks (cf. Section 2.2) are characterised. Evaluation tiers can built upon each other, which is shown in Figure 1, depicting dependencies between evaluation tiers, e.g. the scalability tier (S) builds upon the performance tier (P) metrics while the elasticity tier (E) builds upon the scalability tier with an additional explicit dependency to the resource selection tier (RS). Below the evaluation tiers are described in detail.

![Evaluation Tiers Dependencies](image)

**Fig. 1. Evaluation tiers dependencies**

**Resource selection (RS)** is a crucial decision, determining the best matching resources to run a database system. With respect to traditional RDBMS the focus relies on single database node resources, i.e. CPU, memory and storage while for distributed databases the network and the locality of the database nodes are in the focus as well. While IaaS resources are abstracted by VMs with limited control over the networking [20], recent development in cloud computing tends to offer more heterogeneous resources such as VMs with dedicated storage architectures\(^{11}\) or container based resources\(^{12}\) and locality aware placement of VMs is an ongoing research topic [21]. Hence, the resource selection drives significantly the overall behaviour of the database system, affecting as well evaluation tiers such as elasticity and availability with respect to resource provisioning time or locality aspects [28].

**Performance (P)** represents the basic tier to evaluate the behaviour of a single database system (cf. Section 2.1) against a specific kind of workload. Typical performance metrics are performance and latency, which are measured on the evaluation framework side. Throughput represents the number of requests that are processed per time unit. Latency is measured per request and represents the round trip time for one request.

**Scalability (S)** builds upon the performance tier, taking into account throughput and latency with respect to resources and database cluster size. A well-known definition of scalability for distributed databases in the context of cloud computing is provided by Agrawal et. al. [1], defining the terms scale-up, scale-out and scale-in in order to manage growing workloads. Scale up or *vertically scaling* applies by adding more computing resources to a single

\(^{11}\) [https://aws.amazon.com/de/ec2/instance-types/](https://aws.amazon.com/de/ec2/instance-types/)

\(^{12}\) [https://wiki.openstack.org/wiki/Magnum](https://wiki.openstack.org/wiki/Magnum)
node. The actions scale out (i.e. adding nodes to a cluster) and scale in (i.e. removing nodes from a cluster) apply to distributed databases and are commonly referred to as horizontal scaling. As this work focuses on distributed databases, the term scalability implies horizontal scalability in the following. Measuring scalability is typically performed by correlating throughput and latency for growing cluster sizes. A high scalability rating is represented by constant latency and proportionally growing throughput with respect to the number of nodes and the workload size [9].

Elasticity (E) While the scalability tier targets the general ability to process arbitrary workload sizes, the elasticity tier targets the ability to cope with sudden workload fluctuations, i.e. “the ability to deal with load variations by adding more resources during high load or consolidating the tenants to fewer nodes when the load decreases, all in a live system without service disruption, is therefore critical for these systems.” [1]. Common elasticity metrics are speedup and scaleup [9]. Speedup refers to the required time for a scalability action (scale-in/scale-out), i.e. adapting the cluster size, redistributing the data and stabilising the cluster. Scaleup refers to the benefit of this action, i.e. the throughput/latency development with respect to the workload fluctuation. The scaleup metric is affected by the locality of the cloud resources that are exploited for the new database nodes, i.e. is VM for the node on the same physical host, on the same virtual network or in the same availability zone?

Availability (A) represents the degree to which a system is operational and accessible when required for use. The availability of a database system can be affected by overloading at the database layer, i.e. a high number of requests issued concurrently by clients overloads the server such that the requests of clients cannot be handled or are handled with a latency > ∆t [12] or failures at the resource layer, i.e. a node failure due to a resource failure, e.g. network outage or disk failure. With respect to the database layer availability, an intuitive way to deal with overload is to scale-out the system. Hence, the availability tier builds upon scalability and elasticity as these tiers are exploited to handle request fluctuations [25]. Hereby, common metrics are the throughput/latency development in correlation with the growing cluster size to determine the efficiency in resolving the overload situation.

With respect to resource failures, distributed databases apply the replication of the data to multiple database nodes. A common metric to measure availability with respect to node failures are the takeover time, i.e. how long does it take until a replica takes over the requests, and the performance impact. Again the locality of the database nodes can affect the results in terms of network connection or even physical resource failures.

Consistency (C) With the support of high availability, the need to evaluate consistency comes up, as data can be replicated across multiple nodes. Yet, there is trade-off between consistency guarantees, availability and partitioning, i.e. the CAP theorem [16], which leads to different kind of consistency guarantees offered by the database systems. According to Bermbach et al. [6] consistency can be evaluated from two perspectives, client-centric, i.e. from
the application developer perspective and data-centric, i.e., from the database administrator perspective. In the context of this paper, we only consider the evaluation of client-centric consistency tier.

Client-centric consistency can be classified into staleness and ordering [6]. Staleness defines the lagging of replica behind its master. Staleness can be measured either in time or versions. Ordering defines that all requests must be executed on all replicas in the same chronological order.

4 Evaluation Frameworks Requirements

In order to evaluate distributed databases based on the presented evaluation tiers, evaluation frameworks have to fulfill certain requirements to ensure meaningful results. As there are already well-adopted requirement definition [17], [7], we briefly present existing requirements and extend them with cloud-centric requirements.

Usability (U) eases the framework configuration, execution and extension by providing sufficient documentation and tools to run the evaluation. Hence, the evaluation process has to be transparent to provide objective results [17].

Distribution/Scalability (D/S) is provided by distributed workload generation, i.e., clients can be distributed across multiple instances in order to increasing the workload by utilising an arbitrary amount of clients [7].

Measurements Processing (MP) defines that measurements are gathered not only in an aggregated but also in a fine-grained manner for further processing [7]. As the amount of measurements can grow rapidly for multiple or long running evaluation runs, file-based persistence might not be sufficient. Hence, advanced persistence options such as time series databases (TSDBS), e.g., InfluxDB[^13], will ease the dedicated processing and visualisation.

Monitoring (MO) data improves the significance of evaluation results. Hence, monitoring of the involved resources, clients and database systems should be supported by the evaluation framework to provide the basis of a thorough analysis. Again an advanced persistence solution is beneficial.

Database Abstraction (DA) enables the support of multiple database systems by abstracting database driver implementations. Yet, the abstraction degree needs to be carefully designed as a too high abstraction might limit databases and distort results. Hence, the abstraction interface should be aligned with the specified workload scenarios [5].

Client Orchestration (CO) enables automated evaluation runs. Therefore, the framework should provide tools that orchestrate evaluations, i.e., provision (cloud) resources, create, execute and collect the results and clean-up the clients. Hence, CO eases the creation of arbitrary load patterns and the simulation of multi-tenant workload patterns.

Database Orchestration (DO) enables the management of the database cluster to facilitate repetitive evaluation for different resources, database cluster

[^13]: https://docs.influxdata.com/influxdb/v1.2/
configurations and the adaptation of the database cluster based on predefined conditions. Hence, the evaluation framework should provide tools to automatically orchestrate databases, i.e. provision resources, setup, configure and adapt generic database clusters.

**Multi-phase Workloads (MpW)** define the support of multiple workloads that run in parallel, which is crucial to execute advanced evaluation scenarios. Further, the specification of the load development over a certain time frame per workload is required to simulate real world scenarios.

**Extensibility (E)** defines the need to provide an architecture, which eases the extension of the framework capabilities, e.g. by adding support for additional database systems or workload types.

## 5 Analysis of Evaluation Frameworks

In this section we analyse common evaluation frameworks focusing distributed databases. Hereby, we consider only evaluation frameworks, which have been published within the evolvement of the distributed databases, i.e. from 2007 on. In addition, only the original evaluation frameworks and no minor extensions or evaluations based on these frameworks are considered in the scope of this work. Each framework is analysed in a 2-dimensional process.

First, we classify each framework based on its workload domain and the supporting evaluation tiers (cf. Section 3). The results of this classification are shown in Table 1. Second, we analyse for each framework its capabilities against the presented evaluation framework requirements (cf. Section 4). The results are shown in Table 2. In the following the analysed frameworks are briefly introduced with respect to their results and cross-cutting aspects. The frameworks presented in chronological order of their release date

**TPC-C / TPC-E** The TPC organisation provides evaluation frameworks for various domains\footnote{14 http://www.tpc.org/default.asp}, where the transactional domain is covered by TPC-C\footnote{15 http://www.tpc.org/tpcc/default.asp} and its evolution TPC-E frameworks\footnote{16 http://www.tpc.org/tpce/default.asp}. Both frameworks target the performance and scalability tier and their specifications consider the resource tier with respect to physical resources. Both frameworks offer rich documentation based on their specifications but the database abstraction is limited to RDBMS as both frameworks require a relational data model. Evaluation results can be gathered in different detail level, yet there is no support for advanced measurements processing.

**YCSB** The YCSB [9] enables the evaluation of performance, scalability and elasticity based on synthetic create-read-update-delete (CRUD) workloads, supporting >20 database systems. The framework eases its usage by providing a rich documentation\footnote{17 https://github.com/brianfrankcooper/YCSB/wiki}. The granularity of the measurements is config-
A Cloud-centric Survey on Distributed Database Evaluation

<table>
<thead>
<tr>
<th>Evaluation Framework</th>
<th>Evaluation Tier</th>
<th>WD</th>
<th>RS</th>
<th>P</th>
<th>S</th>
<th>E</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-C/TPC-E</td>
<td>transactional</td>
<td>(✓)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>YCSB</td>
<td>synthetic</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>YCSB++</td>
<td>transactional</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>BG</td>
<td>synthetic</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>BigBench</td>
<td>transactional</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OLTP-bench</td>
<td>synthetic</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>YCSB-T</td>
<td>transactional</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>LinkBench</td>
<td>synthetic</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

✓= not supported, (✓)= partially supported, ✓= supported

Table 1. Distributed database evaluation tiers

... unreadable, but only file based storage is supported and measurements of distributed clients have to be merged manually.

**YCSB++** The YCSB++ framework [22] is an extension to the YCSB, enabling the evaluation tier consistency with the focus on staleness by adding transactional and Big Data workloads based on Apache Hadoop[^18]. The framework targets column-oriented stores but as it builds upon YCSB, it can be easily extended. The orchestration of clients is supported without the resource provisioning. In addition, it supports the evaluation of column-based access control.

**BG** The BG framework [3] builds upon the core of the YCSB but with the focus on the performance, scalability and consistency (staleness and ordering) by simulating a social media platform. It offers a rich documentation[^19] and supports the evaluation of custom service level agreements (SLA) specifications.

**BigBench** The BigBench framework [15] builds upon the TPC-DS[^20] specification, focusing transactional and Big Data workloads. By now, it is adopted by the TPCX-BB benchmark[^21]. The framework targets the performance tier. As the framework heavily relies on SQL queries, its abstraction is limited to RDBMS.

**OLTP-Bench** The OLTP-Bench framework [11] targets the performance and scalability evaluation tier with the focus on RDBMS. The framework pro-

[^18]: http://hadoop.apache.org/
[^19]: http://bgbenchmark.org/BG/overview.html
[^20]: http://www.tpc.org/tpcds/default.asp
[^21]: http://www.tpc.org/tpcx-bb/default.asp
vides an extensive set of 15 workloads, covering transactional, web-based and synthetic workloads. Its usage is described in a basic documentation\footnote{https://github.com/oltpbenchmark/oltpbench/wiki/Quickstart}. The measurements can be gathered in different granularity levels and the automated generation of graphs (including monitoring data) is supported. Yet, measurements are stored file-based and no advanced storage is supported. Database abstraction is provided on RDBMS and NewSQL level, but NoSQL databases are not supported.

**YCSB+T** The YCSB+T framework\footnote{https://github.com/facebookarchive/linkbench} is an extension to the YCSB, focusing transactional workloads. It extends the performance tier by measuring transactional overhead and adds the evaluation tier consistency with the focus on ordering. Besides the original YCSB documentation, no further guidelines are provided.

**LinkBench** The LinkBench framework\footnote{https://github.com/facebookarchive/linkbench} targets the performance tier by applying web-oriented workloads, which are designed according to real world workloads of Facebook. The framework provides an adequate documentation. The scalability of the framework is provided but each client is stateless and the coordinated workload execution across multiple clients is not supported. Measurements can be gathered in a fine-grained manner, but there is no support for advanced processing. The framework is extensible and provides a generic database abstraction layer, but in its original version it only supports one RDBMS.

## 6 Discussion

The first insight of our analysis shows that the performance, scalability, elasticity and consistency tiers are well covered by the evaluation frameworks but the resource selection and availability tier lack support. In the first part of this

<table>
<thead>
<tr>
<th>Evaluation Framework</th>
<th>U</th>
<th>D/S</th>
<th>MP</th>
<th>MO</th>
<th>DA</th>
<th>CO</th>
<th>DO</th>
<th>MpW</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-C/TPC-E</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>✓/</td>
<td>✓</td>
</tr>
<tr>
<td>YCSB</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>✓/</td>
<td>✓</td>
</tr>
<tr>
<td>YCSB++</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
</tr>
<tr>
<td>BG</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
</tr>
<tr>
<td>BigBench</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓/</td>
<td>X</td>
<td>X</td>
<td>✓/</td>
<td>✓</td>
</tr>
<tr>
<td>OLTP-bench</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
</tr>
<tr>
<td>YCSB+T</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>✓/</td>
<td>✓</td>
</tr>
<tr>
<td>LinkBench</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓/</td>
<td>✓</td>
</tr>
</tbody>
</table>

\(\times\) = not supported, \(\checkmark\) = partially supported, \(\checkmark\) = supported

Table 2. Distributed database evaluation tiers
section we discuss these results in more detail. The second insight points out that the traditional evaluation framework requirements [17] are well supported throughout the frameworks, where the monitoring and cloud related orchestration features lack support. A more detailed discussion of these results is provided in the second part of this section. In the third part we provide an outlook, how database evaluation can become more cloud-centric.

6.1 Results: Evaluation Tiers

The resulting table (cf. Table 1) shows that the earlier evaluation frameworks focus on performance, scalability and elasticity, while more recent frameworks focus as well consistency. Hereby, only the performance tier has established common rating indices such as throughput, latency or SLA based rating [3]. While multiple frameworks target the scalability and elasticity tier, a common methodology and rating index has not yet been established. Yet, the need for a common evaluation methodology [25] and rating index [13] is already carved out.

Evaluation tier, which are currently not supported are resource selection and availability. While resource selection is partially supported by TPC-C/TPC-E, by considering physical hardware configurations, cloud-centric resource selection is not in the scope of any of the frameworks. As the heterogeneity of cloud resources has increased over the last years from diverse virtual machines offering to even container based resources, the cloud-centric resource selection the consideration of this evaluation tiers needs to move into the focus of novel evaluation frameworks. Especially, as other evaluation tiers have an implicit or explicit dependency on the resource selection (cf. Figure 1). Yet, the existing frameworks can be applied to evaluate database systems running on heterogeneous cloud resources, but the frameworks do not offer an explicit integration with cloud resources. Hence, the impact on the manual tasks for managing resource monitoring and client/database orchestration hinders cloud-centric evaluations, which is discussed in the following sections.

While high availability is a major feature of distributed databases it is rather surprising that none of the existing evaluation frameworks considers the availability evaluation. Especially as cloud resources can fail, availability concepts for applications running on cloud resources is widely discussed topic, which has already led to frameworks to simulate failures in generic cloud applications [27]. Again, the support of database orchestration can enable database specific availability evaluations.

6.2 Results: Evaluation Framework Requirements

The analysis of the evaluation framework requirements (cf. Table 2) shows that usability, scalability, database abstraction and extensibility are covered by all frameworks. Measurement processing is covered as well but only a few frameworks support advanced features such as visualisation and none of the frameworks supports advanced storage solutions such as TSDBs. Multi-phase workloads are partially covered by the frameworks, especially by the frameworks of
the transactional and web-oriented workload domains. The monitoring of client resources is partially covered as well, but only OLTP-bench considers database resources monitoring as well. While all frameworks support the distributed execution of evaluations, only two support the orchestration of clients, which complicates the distributed evaluation runs. Further, none of the frameworks supports database orchestration. This fact leads to high complexity only for setting up the evaluation environment, especially when it comes to heterogeneous cloud resources. Further, dynamic database cluster transitions for evaluating tiers such as elasticity or availability, always require custom implementations, which impedes the comparability and validity of the results.

6.3 Looking ahead: Cloud-Centric Database Evaluation

The results of this analysis show, that existing frameworks can be applied to evaluate distributed databases in the cloud, but there are still unresolved issues on the evaluation tier side, i.e. the support for resource selection and availability evaluation, and on the framework requirement side, i.e. the orchestration of clients and databases and exploitation of advanced storage solutions. This hinders repeatability [17] of evaluations on heterogeneous cloud resources as well as dynamic transition in the cluster. Yet, cloud computing research already offers approaches to enable automated resource provisioning and application orchestration in the cloud based on Cloud Orchestration Tools (COTs) [4]. Integrating COT into evaluation frameworks can be an option to ease the distributed execution of evaluation runs as well as orchestrating database clusters across different cloud resources. As COTs provide monitoring and adaptation capabilities, they can ease the evaluation of dynamic cluster transitions by defining advanced evaluation scenarios with dynamic database cluster adaptations.

7 Conclusion and Future Work

In the last decade the landscape of distributed database systems has evolved and new database systems such as NoSQL or NewSQL, moved into the focus of industry and academia. In parallel, cloud computing evolved as well and became a novel deployment option for database systems. Yet, these evolutions raise the complexity in choosing an appropriate database system and its resources to run on.

In order to ease such decisions, several evaluation frameworks for distributed databases have been developed. In this paper we present a classification for distributed database evaluation frameworks based on evaluation tiers and requirements towards the frameworks itself.

We apply this classification to eight common evaluation frameworks and provide a thorough analysis of their evaluation tiers and capabilities. The results of this analysis shows that the performance, scalability, elasticity and consistency tiers are well covered, while resource selection and availability are not considered by existing evaluation frameworks. With respect to the framework requirements,
traditional requirements are covered [17], while cloud-centric requirements such as orchestration are only partially supported.

The discussion analyses the impact of these missing capabilities with respect to cloud-centric database evaluation and outlines a possible solutions by exploiting cloud orchestration tools (COTs) to advance database evaluations.

Future work will comprise the analysis of COTs with respect to their exploitation in database evaluation frameworks. In addition, we aim at the design and implementation of a cloud-centric evaluation framework.

Acknowledgements The research leading to these results has received funding from the EC’s Framework Programme HORIZON 2020 under grant agreement number 644690 (CloudSocket) and 731664 (MELODIC). We thank Moritz Keppler and the Daimler TSS for their valuable and constructive feedback on earlier versions of this paper.

References


