FMKe: a Real-World Benchmark for Key-Value Data Stores

Gonçalo Tomás  
NOVA LINCS & DI, FCT,  
Universidade NOVA de Lisboa,  
Portugal  
ga.tomas@campus.fct.unl.pt

Peter Zeller  
University of Kaiserslautern,  
Germany  
p_zeller@cs.uni-kl.de

Valter Balegas  
NOVA LINCS & DI, FCT,  
Universidade NOVA de Lisboa,  
Portugal  
v.sousa@campus.fct.unl.pt

Deepthi Akkoorath  
University of Kaiserslautern,  
Germany  
akkoorath@cs.uni-kl.de

Annette Bieniusa  
University of Kaiserslautern,  
Germany  
bieniusa@cs.uni-kl.de

João Leitão  
NOVA LINCS & DI, FCT,  
Universidade NOVA de Lisboa,  
Portugal  
jc.leitao@fct.unl.pt

Nunó Preguiça  
NOVA LINCS & DI, FCT,  
Universidade NOVA de Lisboa,  
Portugal  
nuno.preguica@fct.unl.pt

ABSTRACT
Standard benchmarks are essential tools to enable developers to validate and evaluate their systems’ design in terms of both relevant properties and performance. Benchmarks provide the means to evaluate a system with workloads that mimics real use cases. Although a large number of benchmarks exist for database system, there is a lack of standard benchmarks for an increasingly relevant class of storage systems: geo-replicated key-value stores providing weak consistency guarantees. This has led developers and researchers to rely on ad-hoc tools, whose results are both hard to reproduce and compare.

In this paper, we propose the first standardized benchmark specially tailored for weakly consistent key-value stores. The benchmark, named FMKe, is modeled after a real application: the Danish National Joint Medicine Card. The benchmark is scalable, it can be parameterized to emulate a large number of access patterns, and it is also highly flexible, enabling its application on systems that offer different consistency guarantees and mechanisms.

CCS CONCEPTS
• General and reference → Evaluation;

KEYWORDS
Benchmark, Key-Value Store

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

PaPoC’17, Belgrade, Serbia
© 2017 ACM. 978-1-4503-4933-8/17/04 . . $15.00
DOI: http://dx.doi.org/10.1145/3064889.3064897

1 INTRODUCTION
Standard benchmarks provide a uniform way for evaluating and comparing different systems. The most used benchmarks for databases (e.g. TPC-C [1], TPC-W [2], etc.) model realistic applications. As such, this type of benchmarks is expected to provide a more realistic performance evaluation than synthetic benchmarks, where individual operations are generated randomly according to some distribution defined in the workload.

While the TPC-* benchmarks work well for the evaluation of relational, strongly consistent database systems, they are a bad fit for evaluating eventually consistent key-value stores. The main issue is that they do not reflect the way key-value stores are typically used. For example, some aggregation queries in TPC-W are very expensive to implement on top of a key-value data model respecting the specification, which can render the value of experiments useless.

Given the need of evaluating their systems, many system developers opt for implementing their own version of popular applications, like Twitter, FusionTicket [3], or even TPC-C/TPC-W. Yet, there is no standardized way of comparing these ad-hoc implementations due to different codebases and the lack of a common specification. The Yahoo! Cloud System Benchmark (YCSB) [4] addresses this problem by providing a set of standard benchmarks that can be used to evaluate key-value stores. However, the operations of the benchmark consists of simple read/write operations, while real-life applications often use more complex access patterns.

In this paper, we present FMKe, a new application benchmark tailored to the evaluation of key-value stores providing weak consistency. It is based on a subsystem of the Danish National Healthcare
This information is accessed concurrently by multiple entities, including medical facilities, such as hospitals, and pharmacies.

The benchmark can be used to evaluate any storage system providing weak consistency guarantees, but also includes variants for evaluating advanced database features, such as highly available transactions.

In the remainder of this paper we present the data model of the FMKe benchmark, describe the operations of the workload, and discuss its implementation and evaluation on top of Antidote [5], a key-value store that supports highly available transactions under geo-replication.
Table 1: Number of read and write operations per FMKe operation. For some operations in the normalized variant the number of reads depends on the current number of prescriptions associated to pharmacy, staff, etc. (denoted by N); this number varies over time.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Frequency</th>
<th>Non-normalized</th>
<th></th>
<th>Normalized</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td># reads</td>
<td># writes</td>
<td># reads</td>
<td># writes</td>
</tr>
<tr>
<td>Get pharmacy prescriptions</td>
<td>27%</td>
<td>1</td>
<td>0</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>Get prescription medication</td>
<td>27%</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Get staff prescriptions</td>
<td>14%</td>
<td>1</td>
<td>0</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>Create prescription</td>
<td>8%</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Get processed pharmacy prescriptions</td>
<td>7%</td>
<td>1</td>
<td>0</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>Process prescription</td>
<td>4%</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Update prescription medication</td>
<td>4%</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2: Number of entities for a workload targeted at performance evaluation

<table>
<thead>
<tr>
<th>Entity</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patients</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Hospitals</td>
<td>50</td>
</tr>
<tr>
<td>Pharmacies</td>
<td>300</td>
</tr>
<tr>
<td>Doctors</td>
<td>5,000</td>
</tr>
</tbody>
</table>

2.2). This module is implemented using Basho Bench [7], an open source benchmarking framework.

**FMKe application server** The FMKe application server receives client requests, and for each application operation issues a number of operation to modify the state of the database.

**Database** The data of the benchmark is stored in the database. In our current prototype, we only support Antidote [5].

We ran our experiments in the Amazon Web Service (AWS) infrastructure. Each data center instance consists of four m3.xlarge machines running the Antidote database servers, four m3.xlarge machines running the FMKe application server and four m3.xlarge running the Basho Bench workload generator. A m3.xlarge machine has 4 vCPUs, 15GB of memory and 80 GB of SSD disk. We used the Ireland, Frankfurt and N. Virginia AWS data centers, with the following mean round-trip-time between machines in those data centers: Ireland-Frankfurt: 22.4ms; Ireland-N.Virginia: 84.9ms; Frankfurt-N.Virginia: 89.7ms. The mean round-trip-time between two machines inside a DC was 0.55ms.

Figure 2 shows a throughput-latency plot for the FMKe benchmark on the Antidote system [5]. The plots shows measurements under three different deployments where we vary the number of data centers in each deployment. We based the measurements on a version of FMKe with normalized data layout. The results show that Antidote scales linearly with the number of DCs. The reason for this is that the majority of operations in the workload are read-only. As read-only operations involve only a single DC in Antidote, they do not generate any additional load on the other DCs. Operations that update the database generate additional load when forwarding updates, but the efficient mechanism for update propagation used in Antidote keeps this additional load low, allowing the throughput to almost double when we add the second DC.

Figure 3 shows the detailed results for a single experiment, where it is possible to observe the evolution of throughput and latency during the complete experiment. These graph are generated by Basho Bench, and are very useful to understand the behavior of the system as the database size increases.

**4 CONCLUSION AND FUTURE WORK**

In this paper we introduced a new benchmark for data stores providing weak consistency, which is modeled after a wide-area healthcare production system for managing medical prescriptions. We briefly presented the data model and operations for this benchmark. We have described our initial prototype and reported preliminary performance results obtained with Antidote database.

As next step we plan to provide a precise specification of the FMKe operations and their functional requirements. From this specification we will derive a reference implementation of the benchmark with bindings for multiple languages. Further, we will define a set of tests that developers can run to assess the consistency and availability properties of their system. For instance, these tests would allow checking whether operations executed atomically, or if the
system provides causality. We also aim for mechanisms to measure data staleness, which is a relevant trade-off for storage systems providing weak consistency guarantees and high availability.

Acknowledgements. This work was partially supported by FCT/MCTES: NOVA LINCS project (UID/CEC/04516/2013) and the European Union, through projects SyncFree (grant agreement number 609551) and LightKone (grant agreement number 732505). We would like to thank Kresten Thorup (Trifork) for his help in the design of the benchmark.

REFERENCES