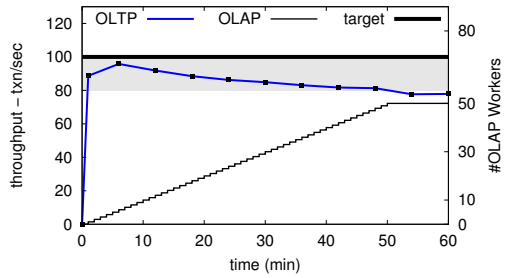
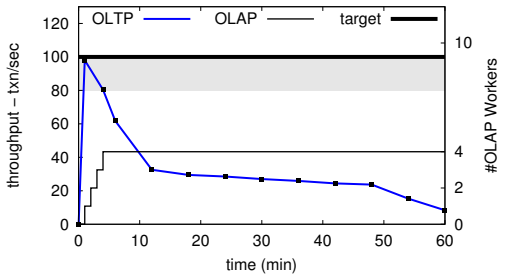
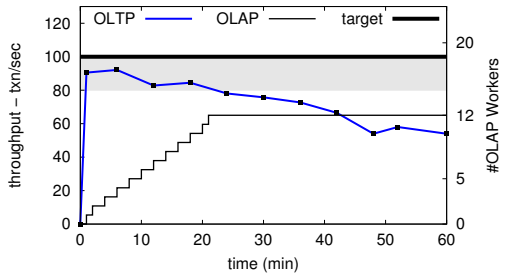


(a) OLTP SUT.



(b) OLAP SUT.



(c) Hybrid SUT.

	# OLAP workers	QpH	QpHpW
OLTP	50	7	0.14 @ 756
OLAP	4	123	30.75 @ 217
Hybrid	12	169	14.14 @ 530

(d) Overview of results for different SUT.

Figure 4: Registered progress between OLTP throughput and induced OLAP scalability. The optimal trade-off between OLTP and OLAP performance is achieved when the OLTP throughput goes below the gray area, at which point the OLAP activity is not further increased. The table in (d) summarizes the OLAP results, showing that the Hybrid SUT is more balanced for an HTAP workload.

3.1 OLTP System

The current experiment used a server with an Intel Xeon E5-2670 v3 CPU with 24 physical cores (48 virtual) at 2.3 GHz and 72GB of memory, running Ubuntu 12.04 LTS as the operating system. We deployed an OLTP row-oriented engine and configured the level of memory that would allow the required number of clients. The Client Balancer within HTAPBench was configured to consider the default error margin of 20%. The admissible loss of OLTP throughput induced by the configuration of the error margin is depicted as a gray area in Figures 4(a), 4(b) and 4(c). HTAPBench was launched from the same machine.

The results in Figure 4(a) depict that HTAPBench was able to launch and sustain the required target *tps* throughout the entire execution time. This is read by looking at the OLTP line that is the linear interpolation of the plotted points. The required target *tps* was reached on the first minute of execution and, from that moment on, the Client Balancer started its configuration by launching an OLAP worker at each minute. The evolution regarding when and how many OLAP workers exist is read by looking at the plotted line resembling a staircase.

With a configured error margin of 20%, the bottom *tps* barrier was only surpassed after 50 minutes, saturating the system at that point in time, and therefore not launching further OLAP workers. Please consider that by default HTAPBench's Client Balancer introduces a single OLAP worker in each evaluation period (Δt). Even though we do not present this evaluation for lack of space, it is possible to configure HTAPBench's Client Balancer to deploy more than one OLAP worker at a time.

Throughout the test, a declining trend in the OLTP throughput becomes evident. In what strictly concerns the OLAP activity, the engine under test was able to hold up to 50 OLAP clients. As previously stated, the main premise in HTAPBench's evaluation scheme is to discover how many OLAP workers are required to stress out a given SUT, while ensuring that the transactional activity does not degrade beyond a configurable threshold. In addition to providing a temporal evaluation of both OLTP and OLAP workers within the system, HTAPBench outputs the unified metric we propose, but also the standard TPC-C *tpmC* and TPC-H *QphH* metrics. Within this setup, this SUT was able to sustain 756 *tpmC* and 7 *QphH*, resulting in 0.14 QpH in each OLAP worker (7 QpH / 50 OLAP workers). As such, the unified metric, *QpHpW*, amounts to 0.14 @ 756 *tpmC*.

3.2 OLAP System

The current experiment reused the previous configuration. We deployed an OLAP column-oriented engine, only setting up the required level of clients allowed in the engine.

The results in Figure 4(b) depict that the SUT sustained the OLTP throughput for a shorter period. This behavior is not unreasonable since the focus of this engine is not on OLTP activity. From the moment the threshold was broken (6th minute), the Client Balancer stopped releasing new OLAP clients. From this time on, albeit at a lower throughput, the OLTP activity was kept stable until the end of the run time. The SUT was able to register 217 *tpmC* while sustaining 4 individual OLAP clients. Cumulatively, the OLAP activity reached a peak of 123 *QphH*. As such, the unified metric we propose, *QpHpW*, reaches 30.75@217 *tpmC*.

Compared with the previous system, we observe that the latter system can enable a larger number of analytic queries, despite the fact that this particular system was only able to launch 4 OLAP streams, compared to 50 OLAP streams with the first SUT. This may seem counter-intuitive, considering that we are working atop a column-based engine specifically designed for an analytical workload. However, the 4 OLAP streams in the current SUT are much more efficient when compared with the 50 OLAP streams in the OLTP SUT.

3.3 Hybrid System

Next, we use HTAPBench to characterize a Hybrid engine. As such systems are designed to scale out, they are usually made available over a distributed architecture.

We deployed the Hybrid system over 10 nodes, 9 of which are responsible for handling and storing data, and the remaining node provides coordination and other global services. Each node has an Intel i3-2100-3.1GHz 64 bit processor with 2 physical cores (4 virtual), 8GB of RAM memory and 1 SATA II (3.0Gbit/s) hard drive, running Ubuntu 12.04 LTS as the operating system and interconnected by a switched Gigabit Ethernet network.

The results depicted in Figure 4(c) reveal that the OLTP throughput reached the assigned target in the first minute of execution. From the first minute onward, the Client Balancer started to deploy OLAP streams until the OLTP throughput degraded beyond the considered error margin, which happened after 20 minutes, registering a grand total of 12 OLAP streams. From that moment on, the OLTP throughput slowly degraded over the remainder of the test duration. Cumulatively, the SUT was able to sustain 530 *tpmC* and 169 QphH. Therefore, our unified metric $QpHpW$, presents as 14.14@530 *tpmC*.

3.4 Discussion

The results of these three tests are summarized in the table in Figure 4(d). Although the OLTP SUT achieved the highest number of OLAP workers, the results in Figure 4(d) also show that it achieved the lowest OLAP performance (0.14). The OLAP workers in the OLTP engine spend most of their time waiting for the OLAP queries to be processed and therefore complete relatively few OLAP queries. On the other hand, the OLAP engine processes significantly more analytical queries, but fails to cope with the required OLTP throughput that is used by the Client Balancer to launch additional OLAP workers. Overall the OLTP engine completed 7 QpH and the OLAP engine completed 123 QpH. This results show that the OLAP SUT was better at handling the OLAP workload, but its inability to cope with the OLTP workload in the hybrid configuration harmed the overall scalability. The favored hardware configuration for the Hybrid SUT is significantly different from the OLTP system or the OLAP system, which undermines a direct comparison among them. Nevertheless, HTAPBench was able to evaluate the Hybrid SUT, enabling 12 OLAP streams and achieving a total of 169 OLAP queries. The results reveal that the Hybrid SUT can sustain a considerable OLTP throughput with a moderate OLAP scalability.

4. VALIDATION

In order to validate the expressiveness of our metrics, we studied the system's representativeness, workload accuracy, homogeneity, error margin variability and cost. Moreover, we discuss the benchmark portability, reproducibility and repeatability. For that matter, we used all the previous SUT configurations.

First, we show that the proposed metric can either identify systems with similar or very different goals. Second, we analyze HTAPBench's accuracy by verifying the storage traces produced in three distinct scenarios. Third, we verify that the produced query result sets are homogeneous as specified. Fourth, we conduct a variability analysis which presents the consequences of varying HTAPBench's error rate. Finally we discuss the cost of using the suite.

4.1 HTAP Unified Metric

The unified metric we propose allows us to establish a comparison across two dimensions. Figure 5 uses a quadrant field plot to visually compare the relationship between our proposed metric and the target OLTP throughput.

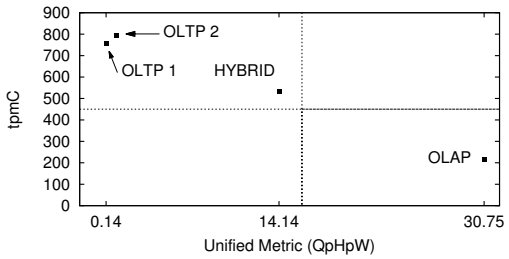


Figure 5: Quadrant field plot for the unified metric.

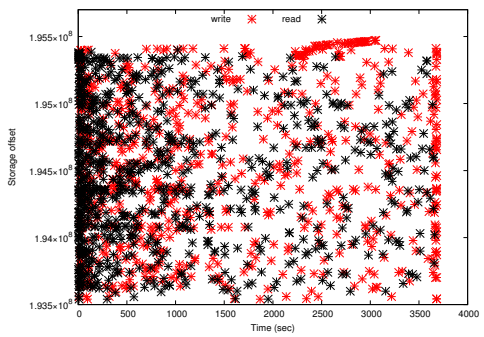
While an increase in the vertical axis translates into a higher OLTP throughput, the horizontal axis evaluates the capability to perform more analytical queries per OLAP worker. The best result would be a reading on the upper right quadrant. The analysis depicts that while the SUT labeled OLTP 1 registered a $QpHpW$ of 0.14@756 *tpmC*, the OLAP observed 30.75@217 *tmpC* and the Hybrid 14.14@534.1 *tpmC*. Overall, we can state that the analyzed systems follow the trend in which the increase of OLTP activity diminishes the OLAP capability. The hybrid system reached a position close to the middle of the plot, indicating better support for the mixed workload with simultaneous OLTP and OLAP activity.

So far, the presented results allow us to conclude that the benchmarking suite is able to compare systems with very distinct work plans. In order to verify if the suite is able to distinguish systems designed for similar workloads, we reproduced the same experimental scenario as in Section 3.1 but changed the SUT to a different engine designed for OLTP workloads. We evaluated its performance and plotted it in Figure 5 with the label "OLTP 2". The results place OLTP system 2 very close to system OLTP 1 with enough precision so as not to overlap both results.

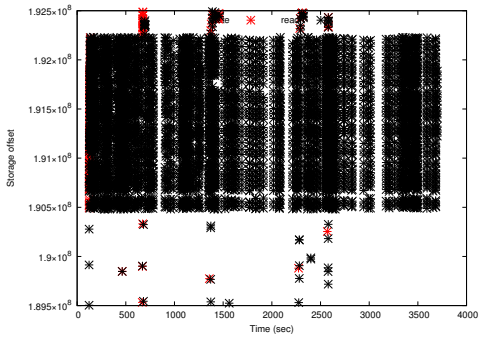
All of the previous experiments were integrated in a statistical study from where we were able to compute the variation coefficient of the computed metrics, pointing to a variation of 1.2%. As such, if the evaluation of 2 different systems, or system configurations produce variabilities with less than 1.2%, the systems should be considered indistinguishable.

4.2 Error Margin Variability

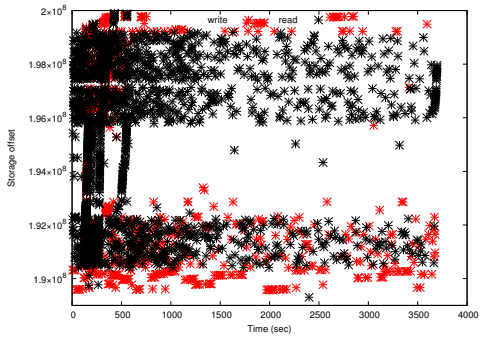
The Client Balancer in HTAPBench controls whether fur-



(a) OLTP access pattern.



(b) OLAP access pattern.



(c) HTAP access pattern.

Figure 6: Traces collected through the blktrace tool during a HTAPBench execution over an (a) OLTP, (b) OLAP and (c) HTAP SUT. The trace represents read (black marks) and write (red marks) patterns induced on the storage medium by the workload. The vertical axis represents the storage medium offset. The horizontal axis represents time in seconds.

ther OLAP streams are deployed or not, evaluating at each point in time if the current OLTP throughput does not go below a threshold defined by the error margin configuration. Intuition leads us to believe that by increasing the error margin, the Client Balancer will naturally allow more OLAP clients to be deployed, thus increasing the overall amount of analytical queries performed.

Error Rate	*QpH*	*tpmC*	*OLAP Clients*
10 %	70.99	130.64	1
20 %	168.9	131.36	5
40 %	265.99	138.30	11

Table 3: Client Balancer error rate variance.

The current experiment assesses if in fact such behavior translated into the actual behavior of the benchmark. For this purpose, we set up the OLAP SUT as in Section 3.2 but varied the error margin across runs.

First, by analyzing Table 3, the reader should be aware that the consecutive executions registered similar OLTP throughputs as expected. Moreover, as we increase the allowed error rate, we verify that more analytical queries were performed, which is a consequence of having more OLAP clients on the system, thus increasing the registered QpH and consequently the QpHpW.

4.3 Workload Representativeness

Database systems must be evaluated with representative workloads that test realistically the storage back-end capabilities. To better understand the representativeness of HTAPBench we analyzed the storage traces produced. The 3 previously tested SUT were evaluated with these workloads and the resulted storage traces were collected and analyzed with the *blktrace*[2] tool. This tool allows us to collect storage traces for a given block device. With these traces it is possible to extract useful information, such as the access patterns of storage requests, the ratio of storage reads and writes, the throughput and latency of each request, etc. Each SUT was deployed in a single machine as in Section 3.1.

Figure 6 depicts the access patterns registered by the storage medium for an HTAP SUT. The figures plot the offset of the storage medium (vertical axis) being accessed during execution time of the benchmark (horizontal axis). The analysis of the traces collected allowed us to confirm that OLTP workloads are dominated by random storage accesses (Figure 6(a)) while OLAP workloads do mostly sequential storage accesses (Figure 6(b)) dominated by read-only requests. On the other hand, mixed workloads generated by HTAPBench present a mix of these access patterns, as expected (Figure 6(c)). Moreover, the ratio of storage reads and writes is according to the specification of each workload. In summary, these results show that HTAPBench is able to simulate a realistic storage load for a hybrid SUT that pro-

[2]blktrace manual: http://linux.die.net/man/8/blktrace.

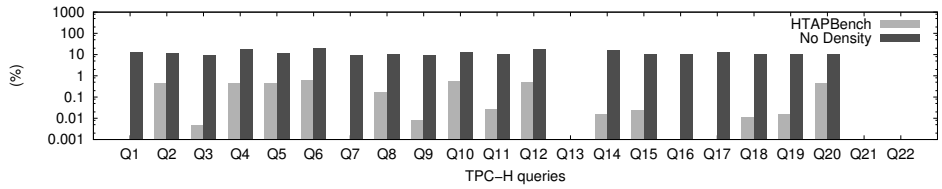


Figure 7: Result set execution cost as a percentage of the baseline across independent runs. The vertical axis is presented in logarithmic scale. The absence of columns represents null variability.

cess both OLAP and OLTP workloads simultaneously. Consequently, HTAPBench's representativeness of the analytical capability on top of a sustained transactional operation is ensured by the experimental data, but also by the individual representativeness of the TPC workloads used. Moreover, the results show that current proposals for hybrid SUT must be aware that processing simultaneously OLAP and OLTP workloads generates a mix of random and sequential storage accesses. This challenge can drive novel research proposals for both hybrid engines and the back-end storage systems supporting them.

4.4 Homogeneity and Reproducibility

The present experiment validates the effectiveness of our density mechanism. Figure 7 depicts 2 different scenarios, where we compare a distribution using random field generation with the density approach we propose. The depicted results relate to a baseline comparison where all queries were executed without any of their filtering operators, thus allowing us to extrapolate the total universe of rows considered in each analytical query. The value presented for each query is the absolute difference in produced result set rows in two consecutive executions: first, with random parameters that do not follow any distribution and, second, with the density mechanism proposed in HTAPBench.

In the first scenario, the analytical queries ran with the introduction of randomized parameters; the results show that consecutive executions of such queries produced variable result sets. The measured variability of result set lines across all query executions for this setup amounted to 12%, with a registered maximum of 20%. For the second scenario, we used HTAPBench's density mechanism to generate the random parameters to be used in all analytical queries. Queries Q13, Q21 and Q22 produce variabilities of 0% as the consecutive runs always produced the same number of result set rows.

The results show that by using our approach, we were able to produce query result sets with comparable costs. Consecutive executions of the same query registered a variability of just 0.17% on average across the different SUT and different configuration settings. These results confirm a very high level of reproducibility of results for a given configuration.

4.5 Benchmark Adoption Effort

The benchmarking suite proposed in this paper derives from two industry standard benchmarks with added functionality. The available suites offer implementations that are individually deployed to collect separate metrics, and they typically require the implementation of connector mechanisms between the benchmarking suite and the engine driver.

HTAPBench follows the trend of a few other suites, relying in the native JDBC driver of the SUT to communicate all the operations of the workload. Concerning the required time to run the benchmark, the setups considered in this paper are the result of 5 independent runs. In each one of them, the SUT had to be populated with data and then tested with the hybrid workload. The time spent during the populate stage greatly depends on the configured target of transactions per second and the SUT used. In the setups considered, we observed an average of 4 hours to populate the files for the engine. Afterwards, before subsequent executions, the engine would dump and restore the initial populate data, taking an average of 15 minutes. This technique is also valid for setups with terabytes of data. Regarding the execution stage, each setup ran for 60 minutes. Consequently, the overall cost for each SUT was 10 hours. Considering other suites, the total cost per SUT is relatively the same.

5. RELATED WORK

There are several organizations that propose benchmarking standards to assess either transactional or analytical systems, namely: the Transaction Performance Council (TPC) [12], the Standard Performance Evaluation Council (SPEC) [7] and the Storage Performance Council (SPC) [6]. We categorize several systems into OLTP, OLAP or Hybrid and we briefly present their main characteristics and shortcomings.

5.1 OLTP

Online Transactional Benchmarking systems, such as TPC-C [8], focus on assessing a system's ability to cope with a large amount of small-sized transactions. Usually, OLTP systems rely on row-oriented data stores, in which the transactions operate over a restricted space of tuples, ensuring at the same time properties such as consistency and isolation in what is commonly referred to as ACID [17].

The TPC-C specification models a real-world scenario where a company, comprised of several warehouses and districts, processes orders placed by clients. The workload is defined over 9 tables operated by a transaction mix comprised of five different transactions, namely: New Order, Payment, Order-Status, Delivery and Stock-Level. Each transaction is composed of several read and update operations, where 92% are update operations, which characterizes this as a write heavy workload. The benchmark is divided into a load and an execution stage. During the first stage, the database tables are populated and, during the second stage, the transaction mix is executed over that dataset. TPC-C defines how these tables are populated and also defines their size and scaling requirements, which is indexed to the number of configured warehouses in the system. The outcome of this benchmark is a metric defined as the maximum qualified throughput of the system, *tpmC*, or the number of New Order transactions per minute.

The TPC-E [11] benchmark builds on TPC-C, introducing a variable number of client terminals. It models a scenario where a brokerage firm receives stock purchase requests from customers, trying to acquire the correspondent stock bonds from a Stock pool. The purchase orders placed by clients are based on asynchronous transactions while stock requests between the Brokerage firm and the Stock Exchange are based on synchronous transactions. Compared with TPC-C, this benchmark builds a much wider and more complex system as it is composed of 33 tables and 10 transactions, 6 of which are read-only and the remainder are read-write transactions; the latter accounting for 23% of all requests.

Both specifications build benchmarking suites strictly intended to evaluate OLTP engines. Despite their representativity of OLTP workloads, the characteristic short operational transactions prevent the workload from exercising OLAP requirements. Engines designed to provide a high OLTP throughput are typically row oriented as the nature of a single transaction induces I/O bound operations in several attributes per transaction. This behavior translates into random access patterns to the storage medium and does not produce CPU and memory bound operations as do OLAP workloads.

5.2 OLAP

Online Analytical Systems such as TPC-H [9, 25] or TPC-DS [10] focus on assessing a system's ability to perform multi-dimensional operations, usually on top of column-oriented data stores.

TPC-H builds a workload that does not require ETL, modeling a real world-scenario where a wholesale supplier must perform deliveries worldwide. The business queries perform complex data operations (e.g., aggregations) across large sets of data. The workload defines a query mix comprised of 22 business queries that access 8 different tables. The execution is divided into three stages. The first stage loads the database. During the second stage, a single user executes all 22 business queries, while during the third stage, a multi-user setup is used in order to evaluate the system's ability to schedule concurrent operations. TPC-H does not consider any growth factor during runtime, which means that the dataset does not change in terms of its total size. The outcome of this benchmark is computed through a metric that accounts for the total number of completed queries per hour ($QphH$).

TPC-DS builds a workload that requires ETL, namely to ensure data freshness. It models a scenario where orders must be processed from physical and online stores of a wholesale supplier, mapping it into a star schema composed of 7 fact tables and 18 dimensions. The workload holds 4 different query types, namely Reporting, Iterative, Data Mining and Ad-hoc queries. The database populated for TPC-DS, as in TPC-H, does not consider any growth factor; still, the initial population is regulated in terms of a scale factor that has direct influence over the data size. The output metric is defined as the number of queries per hour at a given scale factor, *QphDS@ScaleFactor*.

TPC-DS is seen as an evolution of TPC-H, addressing oversimplifications that prevent the proper evaluation of OLAP systems. However, the need to use ETL to promote updates on the star schema prevents us from using it as an OLAP agent in our hybrid workload.

The high prevalence of read operations in these workloads mostly generate sequential accesses to storage mediums, and therefore are not able to simulate the short and cross attribute nature of OLTP workloads that also live in a hybrid workload.

5.3 Hybrid

Hybrid workloads should capture both access patterns observed in the previous individual specifications [15]. There are a few benchmarking suites that use both access patterns, namely CH-benCHmark [5] and CBTR [4, 2, 1, 3].

CH-benCHmark creates a mixed workload also based on TPC standard benchmarks, enabling two types of clients. A transactional client provides a TPC-C agent, while an analytical client provides a TPC-H agent. To allow the analytical workload across the transactional relations, CH-benCHmark merged both schema into a single one, comprising relations from TPC-C and TPC-H. The relations accessed by the OLTP execution scale according to TPC-C's specification. Relations accessed by the OLAP execution are kept unchanged. However, CH-benCHmark neglects aspects within TPC-H's specification. Namely, the analytical queries should hold random parameters in order not to constantly transverse the same regions of the dataset. It also disregards the required distributions for date fields, impacting the produced analytical results.

CBTR defines a benchmarking system aimed at mixed workloads, which does not account for any previous standardized specifications, considering them too predictable. It introduces a workload built from real enterprise data that models an order-to-cash workflow. For that matter, a new schema and the respective transactional activity is presented. By using real data, CBTR bypasses the need to use numerical distributions to populate or to generate data during benchmark execution.

The major differentiator of HTAPBench when compared with the previous approaches lies specifically in its Client Balancer module that governs how both workloads coexist. Both CH-Benchmark and CBTR use naive approaches to find the maximum qualified throughput for both the transactional and analytical workloads. The main concern that arises is that neither workload agent in each benchmarking suite is aware of the other agent, creating a dispute for resources as each agent tries to saturate the SUT. The Client Balancer in HTAPBench follows Gartner's recommendation to specify how the transactional and analytical agents coexist; instructing the transactional agent to sustain a configured throughput and allowing the analytical agent to saturate the SUT up to the point when the transactional throughput is affected. Moreover, HTAPBench also addresses the issues found in CH-Benchmark relating to non-uniform result sets by using the density mechanism to regulate that behavior.

6. CONCLUSION

This paper proposed HTAPBench, a benchmarking suite for engines that support hybrid workloads composed of high levels of transactional activity and, at the same time, provide business analytics directly over production data.

As our main contributions, we proposed a unified metric that enables the quantitative comparison of very similar systems, as well as very different systems. We also propose solutions to the key challenges of a hybrid benchmark, such as the Client Balancer.

Moreover we provide homogeneous and comparable results across executions. We validated our prototype on top of OLTP, OLAP and Hybrid systems, demonstrating the inherent expressiveness of HTAPBench's metrics to characterize such systems. We show that HTAPBench is able to distinguish different classes of systems (i.e., OLTP from OLAP), as well as distinguish systems within the same class with high precision. We ensured that HTAPBench exercised the storage layout as expected for each workload type. The results allowed us to conclude that by using the proposed approach we were able to introduce the required workload randomness while keeping the results comparable by ensuring equal query execution costs across the whole dataset.

7. ACKNOWLEDGMENTS

The authors would like to thank Marco Vieira for all the constructive reviews and comments on the final stage of this work, but also to Martin Arlitt and the anonymous reviewers for their helpful comments. This work is financed by: (1) the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalization - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia as part of project UID/EEA/50014/2013; (2) the European Union's Horizon 2020 - The EU Framework Programme for Research and Innovation 2014-2020, under grant agreement No. 653884.

8. REFERENCES

[1] A. Bog, J. Krüger, and J. Schaffner. A composite benchmark for online transaction processing and operational reporting. In *Advanced Management of Information for Globalized Enterprises, 2008. AMIGE 2008. IEEE Symposium on*, pages 1–5. IEEE, 2008.

[2] A. Bog, H. Plattner, and A. Zeier. A mixed transaction processing and operational reporting benchmark. *Information Systems Frontiers*, 13(3):321–335, July 2011.

[3] A. Bog, K. Sachs, and H. Plattner. Interactive performance monitoring of a composite oltp and olap workload. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 645–648, New York, NY, USA, 2012. ACM.

[4] A. Bog, K. Sachs, and A. Zeier. Benchmarking database design for mixed oltp and olap workloads. In *Proceedings of the 2Nd ACM/SPEC International Conference on Performance Engineering*, ICPE '11, pages 417–418, New York, NY, USA, 2011. ACM.

[5] R. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper, S. Krompass, H. Kuno, R. Nambiar, T. Neumann, M. Poess, et al. The mixed workload ch-benchmark. In *Proceedings of the Fourth International Workshop on Testing Database Systems*, page 8. ACM, 2011.

[6] S. P. Council. *Storage Performance Council*. 2015.

[7] S. P. E. Council. *Standard Performance Evaluation Council*. 2015.

[8] T. P. P. Council. *TPC Benchmark C*. 2010.

[9] T. P. P. Council. *TPC Benchmark H*. 2010.

[10] T. P. P. Council. *TPC Benchmark DS*. 2012.

[11] T. P. P. Council. *TPC Benchmark E*. 2015.

[12] T. P. P. Council. *The Transaction Processing Performance Council*. 2015.

[13] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudré-Mauroux. Oltp-bench: An extensible testbed for benchmarking relational databases. *PVLDB*, 7(4):277–288, 2013.

[14] C. D. French. "one size fits all" database architectures do not work for dss. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, SIGMOD '95, pages 449–450, New York, NY, USA, 1995. ACM.

[15] F. Funke, A. Kemper, and T. Neumann. Benchmarking hybrid oltp&olap database systems. In *BTW*, pages 390–409, 2011.

[16] A. K. Goel, J. Pound, N. Auch, P. Bumbulis, S. MacLean, F. Färber, F. Gropengiesser, C. Mathis, T. Bodner, and W. Lehner. Towards scalable real-time analytics: An architecture for scale-out of olxp workloads. *Proc. VLDB Endow.*, 8(12):1716–1727, Aug. 2015.

[17] T. Haerder and A. Reuter. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317, Dec. 1983.

[18] R. Kimball, M. Ross, et al. The data warehouse toolkit: the complete guide to dimensional modelling. *US: John Wiley & Sons*, 2002.

[19] S. S. Lightstone, T. J. Teorey, and T. Nadeau. *Physical Database Design: The Database Professional's Guide to Exploiting Indexes, Views, Storage, and More*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[20] S. Machine. Splice machine. Technical report, Splice Macine, 2015.

[21] N. Mukherjee, S. Chavan, M. Colgan, D. Das, M. Gleeson, S. Hase, A. Holloway, H. Jin, J. Kamp, K. Kulkarni, T. Lahiri, J. Loaiza, N. Macnaughton, V. Marwah, A. Mullick, A. Witkowski, J. Yan, and M. Zait. Distributed architecture of oracle database in-memory. *Proc. VLDB Endow.*, 8(12):1630–1641, Aug. 2015.

[22] M. Nakamura, T. Tabaru, Y. Ujibashi, T. Hashida, M. Kawaba, and L. Harada. Extending postgresql to handle olxp workloads. In *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on*, pages 40–44. IEEE, 2015.

[23] R. O. Nambiar and M. Poess. The making of tpc-ds. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 1049–1058. VLDB Endowment, 2006.

[24] NuoDB. Hybrid transaction and analytical processing with nuodb. Technical report, NuoDB, 2014.

[25] M. Poess and C. Floyd. New tpc benchmarks for decision support and web commerce. *SIGMOD Rec.*, 29(4):64–71, Dec. 2000.

[26] J. Rossberg and R. Redler. *Pro Scalable .NET 2.0 Application Designs*. Expert's Voice in .Net. Apress, 2005.

[27] VoltDB. Voltdb - whitepaper. Technical report, VoltDB, 2014.

[28] M. Waclawiczek. *HTAP - What it Is and Why it Matters*. 2015.