

HTAP With Reactive Streaming ETL

Carl Camilleri, University of Malta, Malta
Joseph G. Vella, University of Malta, Malta
Vitezslav Nezval, University of Malta, Malta

ABSTRACT

In database management systems (DBMSs), query workloads can be classified as online transactional processing (OLTP) or online analytical processing (OLAP). These often run within separate DBMSs. In hybrid transactional and analytical processing (HTAP), both workloads may execute within the same DBMS. This article shows that it is possible to run separate OLTP and OLAP DBMSs and still support timely business decisions from analytical queries running off fresh transactional data. Several setups to manage OLTP and OLAP workloads are analysed. Then, benchmarks on two industry standard DBMSs empirically show that, under an OLTP workload, a row-store DBMS sustains a 1000 times higher throughput than a columnar DBMS, whilst OLAP queries are more than four times faster on a columnar DBMS. Finally, a reactive streaming ETL pipeline is implemented which connects these two DBMSs. Separate benchmarks show that OLTP events can be streamed to an OLAP database within a few seconds.

KEYWORDS

Column-Store, DBMS, ETL, HTAP, OLAP, OLTP, Reactive Streams, Row-Store

1. INTRODUCTION

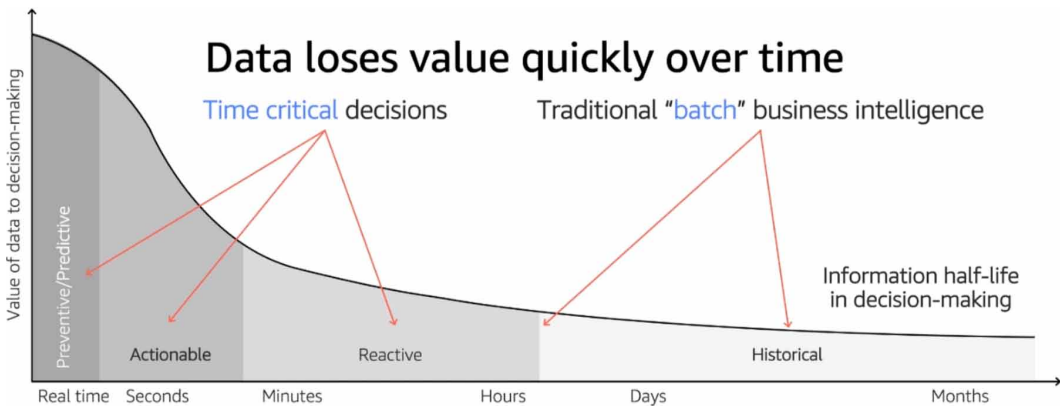
In database management systems (DBMS), query workloads are segmented into two broad modes (Elnaffar et al., 2002; Li et al., 2019). Online transactional processing (OLTP) workloads typically consist of write queries that modify small amounts of data, and queries that read a few records whilst projecting the majority of the attributes available (Bach & Werner, 2016). In OLTP, queries are expected to have short response times, often in the order of microseconds (Harizopoulos et al., 2018), in order to avoid user frustration and business impact (Poggi et al., 2014). At the other end of the spectrum, Online analytical processing (OLAP) workloads typically consist of read-only queries which traverse a large amount of records, performing aggregations and projecting a narrow set of attributes (Bach & Werner, 2016). A system dedicated to OLAP queries is also known as a Business Intelligence (BI) or Decision Support System (DSS), since such queries often aim to elicit information from a data warehouse to support making decisions.

Traditionally, longer response times for OLAP queries have been tolerated, and such queries tend to execute within a dedicated data warehouse which is periodically loaded by data coming from operational (OLTP) systems, typically via extract-transform-load (ETL) processes. On the other hand, modern business requirements are refusing the bounds of these assumptions. The phenomenon of perishable insights (E. A. Lee, 2018), as illustrated in Figure 1, indicates that, in some application

DOI: 10.4018/JCIT.20211001.0a10

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

Figure 1. Perishable Insights (E. A. Lee, 2018)



domains such as fraud detection, data might lose value for decision making as time passes. In such use cases, increasing the data freshness in the OLAP database is beneficial.

2. PROBLEM DEFINITION

Running transactional and analytical workloads efficiently on the same dataset is an open problem which attracts research and commercial interests (Yang et al., 2020). Referred to as Hybrid Transactional and Analytical Processing (HTAP), several approaches are proposed to tackle the ostensibly conflicting demands of preserving the performance of transactional workloads whilst at the same time running analytical queries efficiently on fresh data to facilitate time-critical business decisions.

Several HTAP systems presented in the literature are bespoke DBMSs. These vary from adopting the *Single System for OLTP and OLAP* approach (Yang et al., 2020) that typically rely on support from cutting-edge hardware (Appuswamy et al., 2017) to handle both OLTP and OLAP workloads on the same hardware, to those adopting the *Separate OLTP and OLAP Systems* approach, which deploy loosely-coupled OLTP and OLAP DBMSs.

Several problems are identified. Firstly, although data freshness is largely improved by taking the *Single System for OLTP and OLAP* approach, OLTP and OLAP workloads running on the same hardware conflict, with some systems reporting a reduction of OLTP throughput by three times when running OLAP queries concurrently (J. Lee et al., 2018).

Secondly, reliance on cutting-edge hardware, such as fast non-volatile memory (NVM), restricts DBMS users from exploiting commodity hardware for their workloads and may therefore be either an infeasible solution if the hardware is not available, or require a costlier hardware setup (Neumann & Freitag, 2020).

Lastly, an approach based on bespoke solutions forces the use of specific DBMSs, which might not be compatible with the rest of the software ecosystem or require specialised expertise on the database administrator (DBA) team, increasing the complexity of the information system (IS).

3. APPROACH

The primary objective of this article is to present and empirically evaluate an HTAP setup that takes the *Separate OLTP and OLAP Systems* approach and which is based on DBMSs that are commercially available and supported. This HTAP approach tackles the problems outlined in Section 2, namely by

ensuring that OLTP and OLAP workloads can run on separate hardware, and by using commercially-available DBMSs deployed on commodity hardware without the need of specialised DBA expertise.

To the authors' knowledge this study is a novel approach in presenting a solution to these problems, backed with empirical analysis. The article gives some background in Section 4, discussing different DBMS architectures, standard benchmarks quoted in the literature and used in this article to measure the performance of RDBMSs, as well as discussing other works in the literature in the same research area. Two contributions are subsequently presented.

In the first contribution, discussed in Section 5, empirical experiments are run, using repeatable workloads, on commercially available instances of two types of DBMSs, systematically quantifying and illustrating the difference between them. Besides allowing the calibration of a test bed for the subsequent benchmarks, these experiments quantify and illustrate the aptness of row-based and columnar DBMS to handle OLTP and OLAP workloads, and provide insights to IS DBAs as to the applicability of each type of DBMS when running on commodity hardware deployed in a public cloud infrastructure.

Section 6 discusses the second contribution, where ETL processes based on change data capture (CDC) and reactive streaming approaches are designed and implemented. Empirical experiments then measure the degree of *data freshness* of this approach, or the time taken for a change in the OLTP database to be visible in the OLAP database for decision support workloads. The results are then compared to state of the art systems in related literature, and provide insights on the trade-offs involved in taking the *Separate OLTP and OLAP Systems* approach (Yang et al., 2020) using OLTP and OLAP DBMSs that are commercially available and supported.

4. BACKGROUND

Relational DBMSs (RDBMS) have traditionally been put forward as general-purpose systems that can be deployed to handle different types of workloads. Nonetheless, seminal studies (Michael Stonebraker et al., 2007; Michael Stonebraker & Cetintemel, 2005) as well as recent literature (Butterstein et al., 2020; Rompf & Amin, 2019) support the notion that systems that allow the configuration of the DBMS to target a specific type of workload perform better.

4.1 DBMS Architectures

ACID-compliant databases that conceptually represent data in the relational model, and that can be queried using SQL (Bach & Werner, 2016; Dhindsa, 2012), although challenged by the rise of various newer technologies such as NoSQL (Strauch, 2011) and NewSQL (Pavlo & Aslett, 2016), have often been found to be apt technology for a variety of workloads (Sridhar, 2017; Mike Stonebraker et al., 2005). Table 1 illustrates a simple data model (Dhindsa, 2012). A DBMS's physical storage is typically a one-dimensional structure of pages having a pre-defined size, which defines the granularity of I/O operations (Sridhar, 2017).

4.1.1 Row-Store

In a row-store DBMS, each page stores several rows, or tuples, from a particular table, or relation. Values of the columns, or attributes, of a tuple stored in a page are serialised sequentially. This assertion holds in general for values of simple data types, although specific approaches are taken for values of types which are larger than the size of a page.

With this type of physical storage organisation, a request for the value of a column X within a row N results in a specific operation (Dhindsa, 2012; Ordóñez & Bellatreche, 2018; Sridhar, 2017) that:

1. Reads the whole page where N is stored from persistent storage;
2. Finds row N containing all attributes; and

3. Finds the value of column *X* and returns its value

The way a row-store DBMS would physically store the data is illustrated in Figure 2 (Dhindsa, 2012).

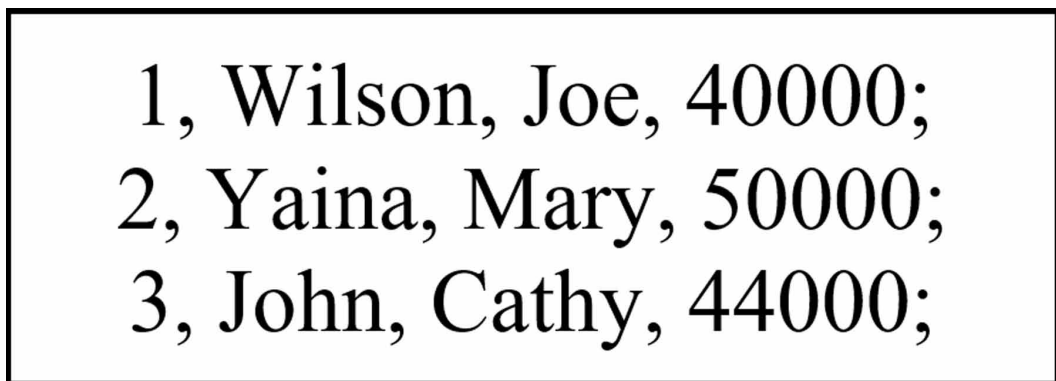
4.1.2 Column-Store

Column stores, or columnar DBMSs, were first introduced in the 1970s (D. Raab, 2007) and have recently become popular with DBMSs such as Vertica (Lamb et al., 2012). In contrast to a row-store RDBMS, a columnar RDBMS vertically partitions a table, physically storing each column in a separate page. Each page contains values organised in an order that reflects the order of the rows (Sridhar, 2017). Figure 3 illustrates the same data model referred to in Table 1, as stored by a columnar RDBMS (Dhindsa, 2012).

Table 1. The relational model (Dhindsa, 2012)

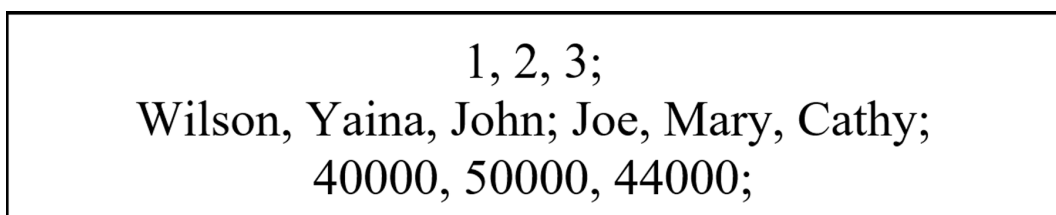
EmpId	LastName	FirstName	Salary
1	Wilson	Joe	40000
2	Yaina	Mary	50000
3	John	Cathy	44000

Figure 2. The relational data model as stored in one page by a row-store DBMS (Dhindsa, 2012)



1, Wilson, Joe, 40000;
2, Yaina, Mary, 50000;
3, John, Cathy, 44000;

Figure 3. The relational data model as stored across four pages by a columnar DBMS (Dhindsa, 2012)



1, 2, 3;
Wilson, Yaina, John; Joe, Mary, Cathy;
40000, 50000, 44000;

Taking the same example, a request for the value of a column X within a row N results in an operation that:

1. Reads the page where X of N resides; and
2. Finds the value of column X and returns its value.

Especially in large data sets and with data compression techniques employed efficiently, this results in much less I/O overheads than the equivalent operation in a row-store RDBMS (Dhindsa, 2012; Ordonez & Bellatreche, 2018; Sridhar, 2017).

Conversely, data modification operations (namely INSERT, UPDATE and DELETE operations) need to span multiple pages and hence tend to be more expensive than the equivalent operation in a row-store DBMS. Although, in principle, it is possible to vertically partition a table in a row-store RDBMS to achieve a columnar data organisation, a database engine aware of the columnar data storage can be finely tuned to achieve other benefits, such as improved compression capabilities and late materialization (Abadi et al., 2008; Macyna & Kukowski, 2020), that are not feasible to achieve in a database engine which is tuned for row storage.

4.1.3 Hybrid transactional/analytical processing (HTAP)

Besides performance considerations, HTAP RDBMSs introduce the axis of data freshness in DSS (Raza et al., 2020) in order to improve data freshness for DSS.

There are a number of approaches to HTAP (Raza et al., 2020), including:

1. **OLTP active instance switching**, where the OLTP DBMS is deployed in a Master/Slave (Active/Passive) configuration, with OLAP queries executing on the passive instance.
2. **Co-located OLTP and OLAP**, which can be like OLTP active instance switching but the two separate DBMS engines are deployed on the same infrastructure, sharing memory and CPUs. Active instance switching happens at hardware level, with OLAP queries executing in a segment of memory that is separate from OLTP, and that contains a version of the database at a particular point in time in memory. In general, these approaches build on the strengths of both row-stores and column-stores (Arulraj, Pavlo, et al., 2016; Makreshanski et al., 2017; Pavlo et al., 2017), such DBMSs rely on:
 - a. data being stored in shared memory, accessible by OLTP and OLAP requests;
 - b. multiple layers of caching and cache coherence algorithms to provide consistency guarantees; and
 - c. parallelism to concurrently execute OLTP and OLAP queries.
3. **Isolated OLTP and OLAP**, where the OLTP and OLAP DBMS engines run in complete isolation and are as de-coupled as possible.

4.2 Benchmark Workloads

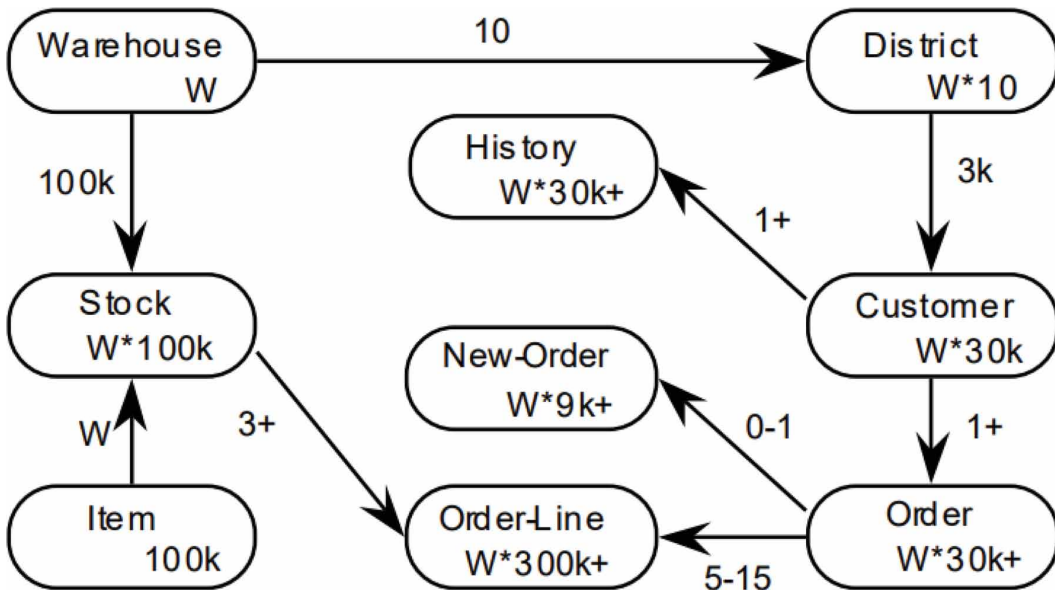
In this work, several empirical experiments to analyse the different approaches are performed. Standard workloads are used to ensure repeatability of these benchmarks, and to allow results to be comparable with other works in the literature that also present results based on the same workloads.

4.2.1 OLTP Workloads

Similar to several related works in the literature (Harizopoulos et al., 2018; Huang et al., 2020; Prasaad et al., 2020), in this article transactional workloads are simulated using the TPC-C synthetic benchmark (F. Raab, 1993). This benchmark simulates a database that models several geographically distributed brick and mortar warehouses, each associated to one or more districts. Several terminals

perform transactions on stock available in each warehouse. The logical schema for the database created by the TPC-C benchmark is illustrated in Figure 4.

Figure 4. TPC-C Logical Schema and Scale Factor for OLTP (Council, 2010)



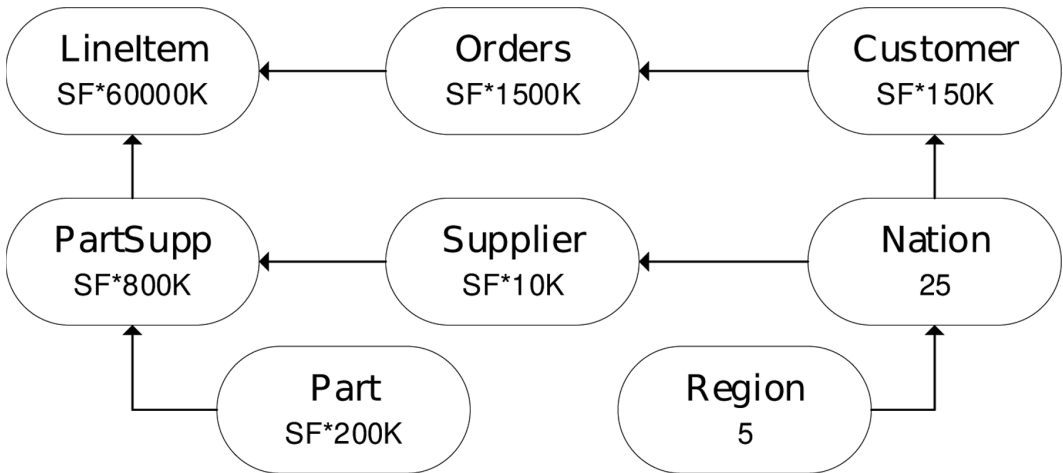
A TPC-C workload is characterised by five transactions, namely a) NewOrder (45%), b) Payment (43%), c) OrderStatus (4%), d) Delivery (4%) and e) StockLevel (4%). Each transaction consists of several queries. The workload can be throttled by two parameters. The scale factor (*sf*) specifies the number of records that are generated within the database. The *scale factor* determines the number of warehouses available, which in turn determines the number of records generated in the other tables, as per the multiplier factor shown in Figure 4. Conversely, the *number of terminals* determines the number of parallel threads that the workload generator spawns to execute concurrent transactions. Hence, larger scale factors imply that TPC-C queries are heavier (e.g. record selections operators applied to larger tables), but possibly less contentious (a larger number of records reduces the probability of contention), whilst a larger number of terminals increases the number of concurrent transactions, and thus the probability of contention.

4.2.2 OLAP Workloads

Analytical workloads are simulated using the TPC-H synthetic benchmark (Council, 1999), which is purposely defined by its authors to validate the performance of a decision support workload and is applicable to both row stores and a column stores. The choice of TPC-H to run OLAP workloads in this article also follows best practices adopted by recent, related works in the literature, that either use TPC-H (Halfpap & Schlosser, 2020; Raza et al., 2020), or CH-benCHmark (Huang et al., 2020), which is a combination of both TPC-C and TPC-H (Cole et al., 2011).

The TPC-H workload simulates a set of ad-hoc queries that aid decision support of a nationwide wholesale supplier. The logical schema for the database of this benchmark is illustrated in Figure 5 (Council, 1999).

Figure 5. TPC-H Logical Schema and Scale Factor for OLAP (Council, 1999)



The benchmark is throttled by a single parameter, the *scale factor* (sf). Like TPC-C, the scale factor determines the number of records that are generated within the OLAP database before the DSS workload queries are executed. In TPC-H, the scale factor roughly determines the size of the database in GB. For example, a scale factor of 1 means a database size of approximately 1 GB, whilst a scale factor of 10 means a database size of approximately 10 GB.

4.3 Related Work

A number of approaches to HTAP (Arulraj et al., 2016; Makreshanski et al., 2017; Pavlo et al., 2017) advocate a *Single System for OLTP and OLAP* approach, and are typically based upon an in-memory bespoke DBMS (Babeanu & Ciobanu, 2015).

Other literature is closer to the approach taken in this article in adopting the *Separate OLTP and OLAP Systems* approach, and some of these recent works are discussed here in more detail.

Integrated Synchronisation (Raza et al., 2020) is an HTAP solution for separate, shared-nothing OLTP and OLAP DBMSs. The solution is specific to the IBM DB2 DBMS engine, and employs engine-specific approaches to efficiently transfer transactional data stored in a row-store DBMS to a column-store DBMS that is efficient for OLAP workloads. The authors benchmark the solution using a tailored workload and report that it takes a maximum of 10 seconds to make transactional data available for OLAP queries.

F1 Lightning (Yang et al., 2020) is described as a loosely-coupled HTAP solution. It encapsulates three core components, namely: 1) the *Change Pump* that reads CDC events from the OLTP database; 2) *Lightning* that receives data from the Change Pump and is responsible to maintain Log-Structured Merge trees on distributed file systems, stored in a column-store format that is optimal for OLAP queries; and 3) the *SQL Processor*, which accepts F1 SQL queries and sends them to Lightning for processing. F1 Lightning supports OLTP workloads in Google Spanner and F1 DB and allows OLAP queries specifically through the F1 Query engine. The authors present metrics from Google-specific products, such as Adwords, and report that data could be available for OLAP queries within 10 minutes.

TiDB (Huang et al., 2020) is an approach to HTAP which uses a row-store DBMS and a column-store DBMS for OLTP and OLAP workloads respectively. In this approach, multiple row-store DBMS replicas, based on the TiKV distributed DBMS, are deployed as a cluster to handle transactional queries. The cluster is kept synchronised using a Raft-based algorithm. TiDB introduces TiFlash, composed of *learner* nodes in the Raft cluster that receive only asynchronous updates from the leader node of the Raft cluster. More importantly, the learner nodes transform the transactional

data to column-store, and therefore become available to handle OLAP queries efficiently. TiDB is benchmarked using CH-benCHmark, to run TPC-C and TPC-H workloads to simulate concurrent OLTP and OLAP queries. Using a scale factor of 10 and 100, and between 64 and 1024 terminals, the authors report that data is replicated between TiKV and TiFlash in less than 1.5 seconds.

5. ROW STORES VS. COLUMN STORES

Several works in the literature show that row-store DBMSs and column-store DBMSs are more apt for OLTP workloads and OLAP workloads respectively. Other works analyse the performance of a columnar design within a row-store DBMS (Andurkar, 2012). Here, the performance of two commercially available RDBMSs is empirically analysed and quantified, namely Postgres (Michael Stonebraker & Rowe, 1986), a row-store RDBMS, and Vertica (Lamb et al., 2012; Mike Stonebraker et al., 2005), a columnar RDBMS.

5.1 Methodology

The experiments were executed on PostgreSQL 11.4, and Vertica 9.2.1. PostgreSQL was deployed in a *n1-standard-16* machine in Google Cloud Platform (GCP), having 16 vCPUs, 60GB RAM and 2TB SSD disk, running with default configuration on Debian GNU/Linux 9. PostgreSQL is configured with the default READ COMMITTED transaction isolation level. Vertica was similarly deployed with default configuration on a *n1-standard-16* machine, with the management console deployed on a *n1-standard-4* machine (4 vCPUs, 15GB RAM and 15GB standard disk). Like PostgreSQL, the default READ COMMITTED transaction isolation level is kept for Vertica.

A third *n1-standard-16* machine was deployed to run benchmark workloads. OLTP-Bench (Difallah et al., 2013) was used to run different workloads, applied to both OLTP and OLAP, with minor modifications to support Vertica. Each experiment was executed three times, and the aggregate output of OLTP-Bench was retained in each run. The average duration from each run is reported in the results.

5.2 OLTP Performance Evaluation

OLTP performance of each DBMS was measured on the infrastructure using the TPC-C workload. Figure 6 and Figure 7 show the throughput achieved, in terms of requests/second when running the TPC-C benchmark via OLTP-Bench against PostgreSQL and Vertica respectively.

5.3 OLAP Performance Evaluation

OLAP performance was analysed using TPC-H workloads, executed in two phases.

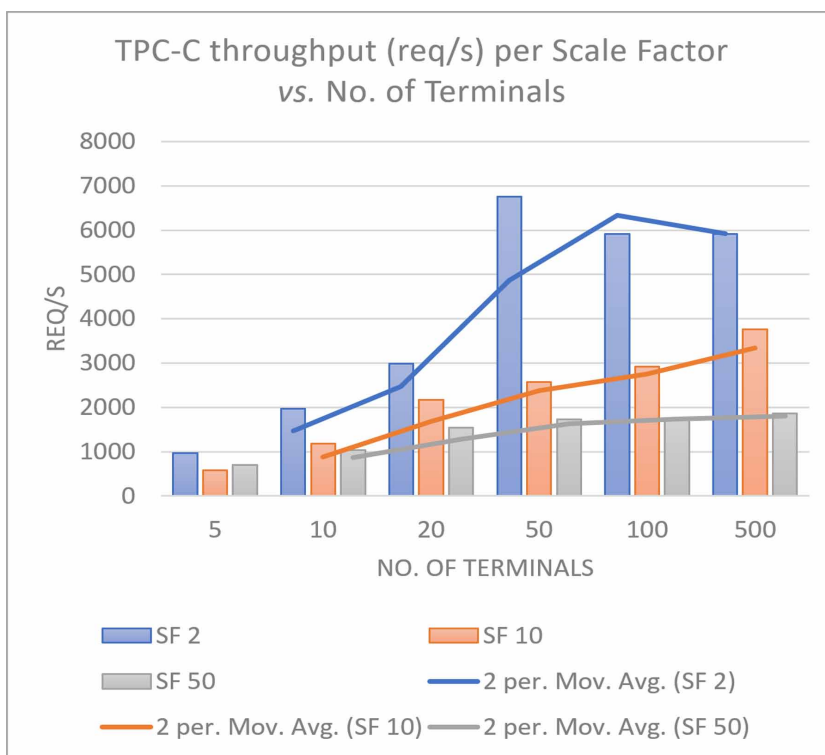
The first phase is the data loading phase, where TPC-H data for the corresponding scale factor was generated using the DBGen tool (Gray & Barkhatov, 1994; Phillips, 2011). OLTP-Bench was then used to create and load a database with the output of DBGen. In the case of PostgreSQL only, the indexes defined by the TPC-H specification were created on the tables after the data was loaded. The second phase is the query execute phase, where all the 44 TPC-H queries were executed in series.

The duration of each phase was recorded, and each benchmark was executed three times for each varying degree of scale factors 1, 2, 5, 10, 20 and 50. Results for data loading and query execution phases are shown in Figure 8 and Figure 9 respectively.

5.4 Observations

Although the experiments use TPC workloads, the results of these experiments are not presented as valid and official TPC results for the systems under test. On the other hand, given that the same experiments are executed on two different systems running on the same hardware, the results are

Figure 6. TPC-C Query Execution on PostgreSQL



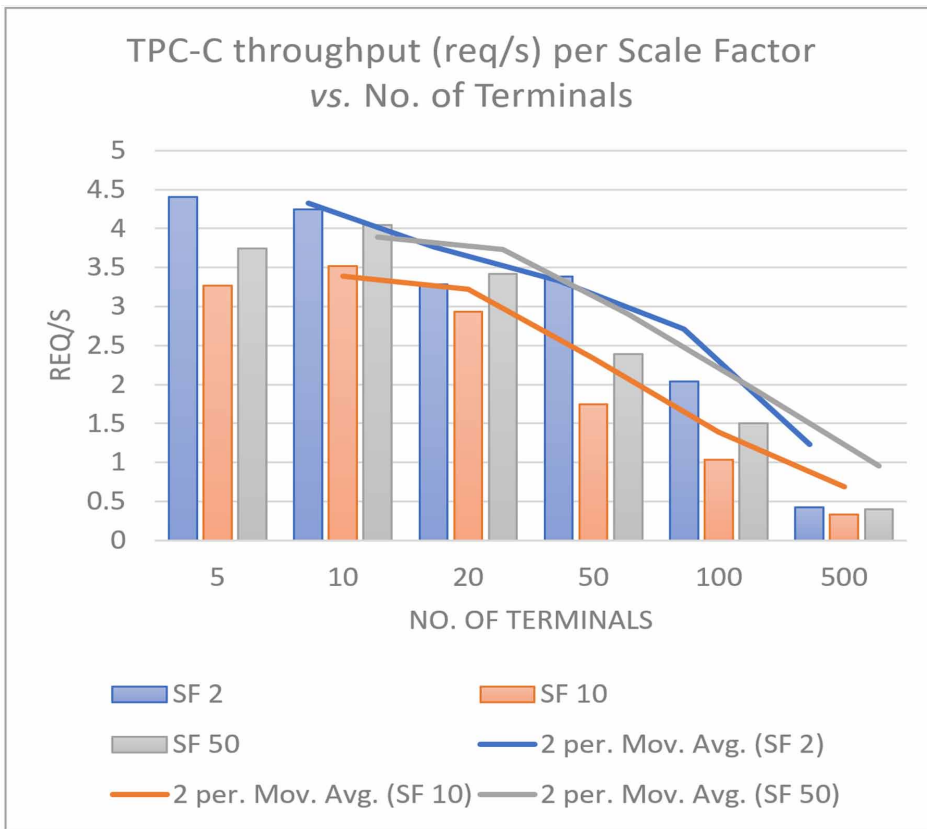
deemed valid comparative benchmark results as the throughput of PostgreSQL and Vertica for OLTP and OLAP workloads.

Figure 6 shows that the throughput of PostgreSQL increase as the number of terminals increase, up to a point where the resources are saturated (i.e. CPU and I/O usage is maxed out) that adding more terminals does not yield a larger throughput. These results also illustrate that this behaviour remains consistent across different values of the benchmark’s scale factor, and therefore, across different sizes of the database. A larger number of terminals also increases the difference in throughput across scale factors, with larger scale factors exhibiting smaller throughput. This is expected in a TPC-C workload.

Figure 7 on the other hand shows that the throughput of Vertica decreases as the number of terminals increase. Furthermore, when compared to PostgreSQL, the maximum throughput seen for a TPC-C workload is orders of magnitude lower than that achieved when running the same workload on PostgreSQL. These results are therefore consistent with the literature, such as the “house pattern” (Psaroudakis et al., 2014), where column-store DBMSs are not advocated for transactional workloads. In this case, the experiments go a step further in also quantifying the throughput of a column-store DBMS under the TPC-C workload.

Specifically, for Vertica, this behaviour is also consistent with the lock modes available in the DBMS (HPE Vertica, 2021). Vertica requires an *I* (Insert) object lock during an INSERT operation, and an *X* (Exclusive) object lock when performing DELETE and UPDATE operations. In Vertica, *object* locks apply to tables and projections. Particularly, *I* locks are compatible with each other, so INSERT operations on the same object can run concurrently. Conversely, *X* locks are not compatible with either *X* or *I* locks. Therefore, a DELETE or UPDATE operation coming from an equivalent OLTP operation locks a table exclusively, resulting in a greater probability of contention.

Figure 7. TPC-C Query Execution on Vertica



Subsequently, the behaviour observed for the TPC-H workload is opposite that of the TPC-C workload, as shown in Figure 8 and Figure 9. Specifically, loading a dataset within PostgreSQL becomes orders of magnitude slower than Vertica as the size of the dataset (defined by the scale factor sf) being imported increases, as shown in Figure 8. This is considered a significant metric, since DBs serving OLAP workloads are typically populated via batch ETL processes, during which large amounts of data need to be ingested. Therefore, the results show that such ETL processes would run much faster against Vertica than PostgreSQL, and this is consistent across data sets of different sizes.

Furthermore, the duration of the TPC-H query workload is also better when running against Vertica, and the performance gap grows as the size of the dataset increases, as illustrated in Figure 9. These results show that, as well as quantify by how much, OLAP query workloads execute faster in Vertica than PostgreSQL. Interestingly, in these benchmarks, the duration of the OLAP query workload against a scale factor of 50 was shorter when compared to the same test against a scale factor of 20. This behaviour requires further analysis, at least to ascertain whether it is also consistent for larger datasets.

6. HTAP WITH STREAMING ETL: AN IMPLEMENTATION

In Section 5, it was ascertained, and quantified, that the PostgreSQL row-store RDBMS and the Vertica column-store RDBMS are suited for OLTP and OLAP workloads respectively. As discussed previously, the time taken for a change in the OLTP database to be available to support decisions in

Figure 8. TPC-H Data load times

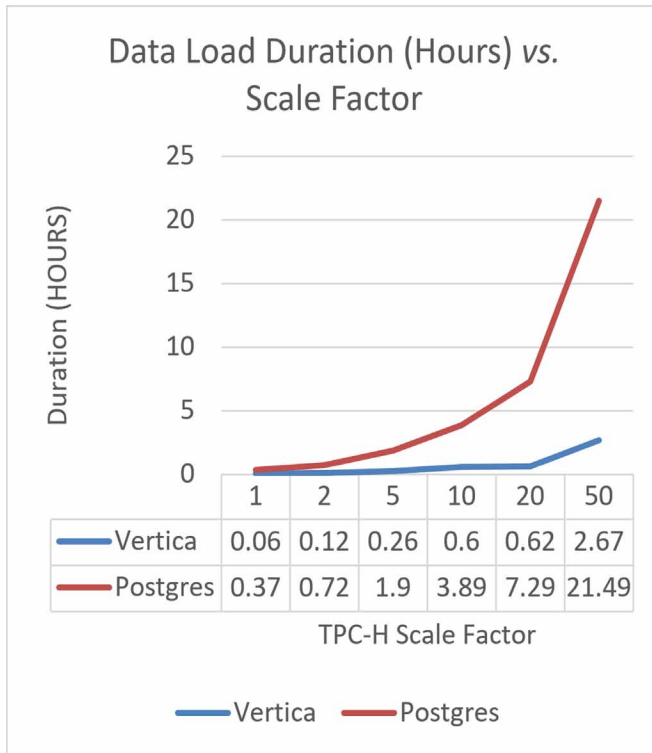
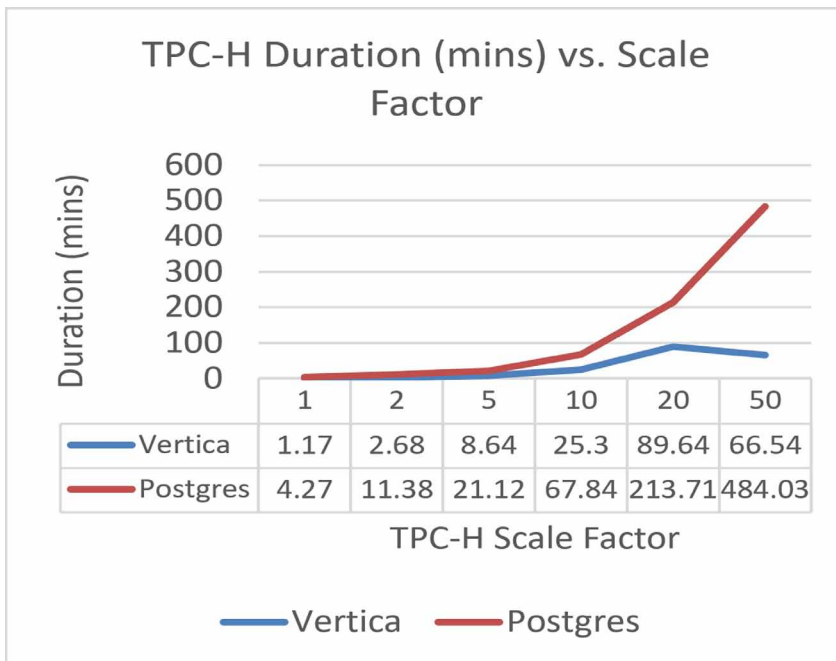


Figure 9. TPC-H Query execution times

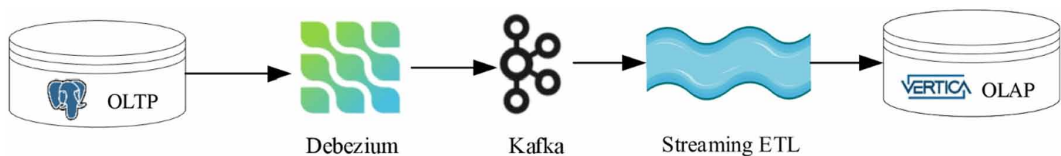


the OLAP database is crucial. Thus, an approach to measure this latency is designed, implemented and benchmarked.

6.1 Methodology

A streaming ETL pipeline as illustrated in Figure 10 was designed. The primary objective of this streaming pipeline is to capture changes happening in the OLTP database and construct and execute the equivalent SQL statements necessary to reflect the same change in the OLAP database.

Figure 10. Streaming ETL pipeline



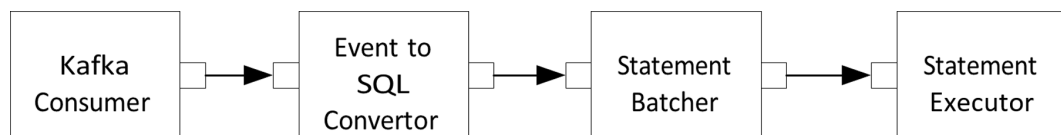
The implementation therefore focuses on problem domains where business decisions can be made from results of queries that operate upon the same data schema as the transactional one. These would therefore be domains where the “transform” step in the ETL process is trivial. However, the approach can also be generalised to more complex data transformation steps.

Debezium (Hauch, 2020) is plugged on to the DBMS handling the OLTP, such that write operations from the OLTP workload are turned into an event stream of CDC events.

A number of studies (Molina et al., 2018; Shen et al., 2019) employ Apache Kafka (Goodhope et al., 2012; Kafka, 2014) as a storage layer for events, and the same approach is taken to use Kafka as a fast, append-only log for the events generated by Debezium. For this purpose, Kafka is configured with a single topic in the OLTP database, in turn having one partition. Through Kafka’s message ordering guarantees, this configuration ensures that events are streamed in the same order as they happen within the OLTP database.

Finally, a custom streaming ETL module was built. The responsibility of this module is to consume events from Kafka, translate them to relevant SQL statements and execute the resultant operation to the OLAP database. The streaming ETL module consists of around 400 lines of Scala code, and is built upon reactive principles (Bonér et al., 2014) using Akka streams (Lightbend, 2020). Figure 11 illustrates the reactive stream implemented in the Streaming ETL module, where:

Figure 11. Reactive Stream stages of the Streaming ETL component



1. The *Kafka Consumer* source step pulls events from Kafka;
2. The *Event to SQL Convertor* flow step generates the relevant INSERT, UPDATE or DELETE statement;
3. The *Statement Batcher* flow step concatenates up to 1000 statements within a maximum of 100 milliseconds;

4. The *Statement Executor* sink step sends a batch of events to Vertica.

This implementation, through the Akka streams module, follows the Reactive Streams specification and specifically addresses the issues of:

1. Concurrency between the three main steps of the ETL pipeline: Kafka event consumption, SQL translation and Vertica query execution.
2. Flow control, by enforcing a back-pressure mechanism to protect against the “fast producer, slow consumer” problem.

6.2 Evaluation

The streaming ETL pipeline was deployed in a 3-node Kubernetes cluster in GCP, each of type *n1-standard-4* (4 vCPUs, 15 GB memory). OLTP and OLAP databases were deployed as PostgreSQL and Vertica, in the same configuration detailed previously. Two metrics were analysed in this evaluation, namely:

1. The CDC delay i.e. the duration between the time a record is committed to the log of the OLTP database, and the time it is stored in Kafka and made available for the first stage of the ETL stream;
2. The OLAP loading time, which is the time taken for the Statement Executor to commit a batch of statements to Vertica.

Figure 12 illustrates the mean duration and the relative probability density plot superimposed on the cumulative probability density plot, for the second metric, as measured against a TPC-C workload, executed for varying values of TPC-C terminals. Figure 13 shows the corresponding statistics for the first metric. The TPC-C workload was used for this benchmark, using a scale factor of 1. The workload executes for a duration of 60 seconds in each run.

6.3 Observations

The results presented show that CDC delay in the streaming ETL process is both minimal and largely invariant. Specifically, in these experiments a maximum of 5.15ms average delay was experienced between a transaction being committed in the OLTP Postgres database, and the data being available for consumption in Kafka. The box plot in Figure 13 also shows that latency at the 95th percentile varies by a maximum of 8.1 milliseconds from the mean.

The process of loading the data inside Vertica takes longer, and the durations are distributed across a wider range. From Figure 12, the average duration of this step of the ETL process varies between 11.8 seconds and 17.7 seconds. This analysis shows that although the mean duration does not vary significantly with the number of TPC-C terminals, or with the amount of CDC events being replicated to Vertica, the distribution of values is impacted more significantly:

1. The delay at the 90th percentile varies in the range of 25.6 and 40.8 seconds;
2. At the 99th percentile the range of variation is between 33.3 and 61.3
3. The maximum delay observed in the experiments is 93.2 seconds.

The experiments also show that both the number of concurrent terminals, as well as the number of CDC events captured during the experiment, have a bearing on the OLAP loading times.

Firstly, the amount of CDC events being replicated has a bearing on the duration of the OLAP loading times: the target OLAP DBMS (Vertica) generates exclusive table locks for UPDATE

Figure 12. OLAP loading times and corresponding CDC delay

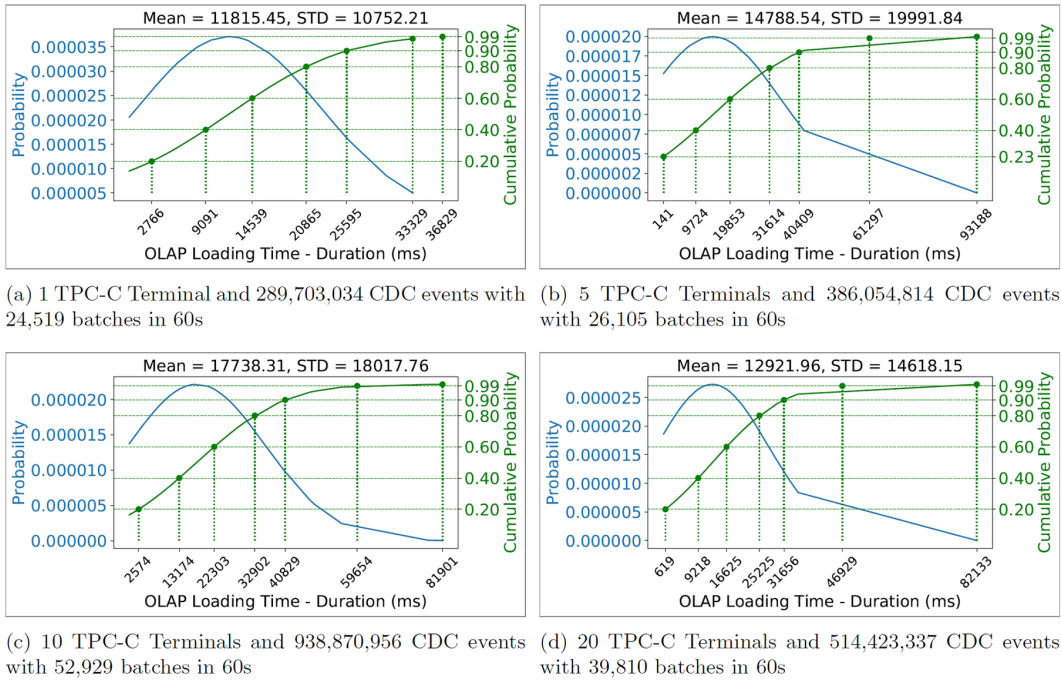
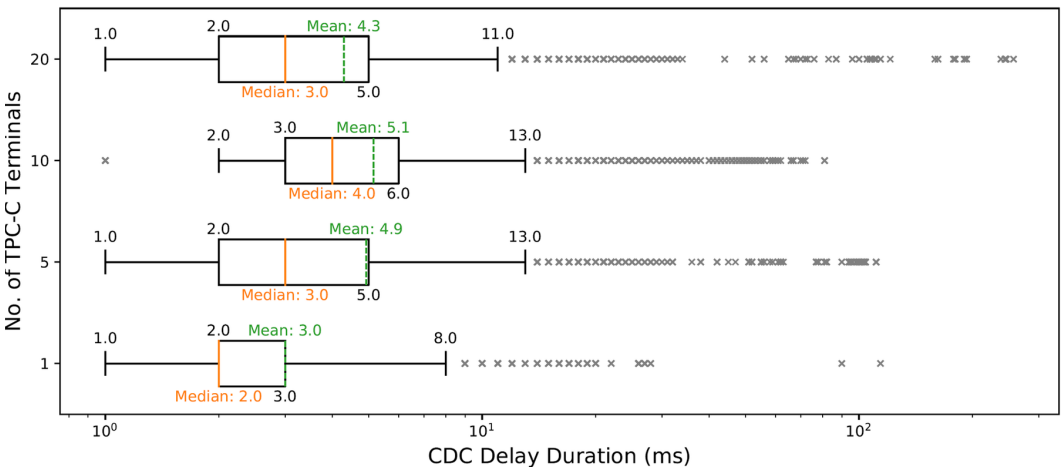


Figure 13. CDC delay within the 5% and 95% range



and DELETE operations and therefore, a higher number of CDC events increase the likelihood of contention.

Secondly the experiment with 10 TPC-C terminals generated more CDC events than the experiment with 20 TPC-C terminals. This is because the OLTP database became saturated with 20 TPC-C terminals and therefore less transactions were accepted during the benchmark. This situation also impacts the CDC delay: the benchmark with 20 TPC-C terminals had a much higher maximum

CDC delay (257 seconds) compared to the benchmark with 10 TPC-C terminals (81 seconds), caused by the fact that the OLTP database was saturated and hence the replication speed also suffered. The experiment with 10 TPC-C terminals exhibits a higher latency at the 90th and 99th percentile than the experiment with 20 TPC-C terminals, however the latter exhibits a slightly higher maximum latency at the tail-end.

This observation indicates that an OLTP workload that is not uniformly distributed across time (i.e. the amount of CDC events that need to be replicated to an OLAP database spike for a short period of time) can experience higher latencies at the tail end, compared to OLTP workloads which generate more CDC events but are more uniformly distributed across an equivalent period of time.

7. CONCLUSION

Through this work, two contributions have been put forward. Firstly, the strengths and weaknesses for row-stores and column-stores for OLTP and OLAP workloads were empirically shown using standard TPC workloads running on a public cloud infrastructure.

Secondly, a generic reactive data streaming pipeline was built to connect a row store DBMS with a column store DBMS, and the behaviour of the data capture and data transfer stages of this pipeline were illustrated. This work has also contributed a small extension to the popular open-source utility, OLTP Bench, to allow connection to the Vertica DBMS.

It was observed that it is possible to have a setup of loosely coupled, shared-nothing OLTP and OLAP DBMSs, but still achieve data freshness supporting actionable and reactive decisions. This contrasts with traditional batch ETL processes, where typically only historical decision support is possible.

Solutions that adopt the *Single System for OLTP and OLAP* approach for HTAP can support real-time decisions and can be considered simpler solutions where a single DBMS needs to be maintained. The state of the art systems that also adopt the *Separate System for OLTP and OLAP* based on bespoke DBMSs, such as the ones described in Section 4.3, report results showing that they can also support quasi-real-time and actionable decisions.

Conversely, the contributions of this article constitute a novel approach in tackling the problems identified in Section 2, and achieve the primary objective set out in Section 3.

Firstly, the solution can be deployed using robust and commercially supported DBMSs that implement the well-known relational data model, and which can be hosted on commodity hardware, in contrast to studies around bespoke DBMSs referred to in Section 4.3. The benchmarks were executed on a public cloud, showing that this approach does not require specialised hardware like other in-memory DBMSs.

This approach also provides maximum flexibility in picking the right tool for the right job. The reactive streaming pipeline approach can generalise to connect any DBMSs of equivalent functionality, also applying to scenarios where decisions need to be taken based on data coming from multiple OLTP databases. The approach therefore fits well in a wide variety of IS ecosystems, without requiring specialised DBA expertise. This is thus in contrast to the state of the art systems reviewed in Section 4.3, which are based on very specific DBMSs, or built around specific cloud services.

Finally, a secondary objective is also achieved in achieving implicit resiliency from two separate systems. For example, the OLAP DBMS can be taken offline for maintenance without impacting OLTP workloads, which is harder to achieve with a single-DBMS HTAP setup handling both workloads. The presented reactive streaming pipeline supports such scenarios, as CDC events can be queued until the target DBMS is ready to accept requests.

The results of the empirical benchmarks presented therefore show that dedicated, shared-nothing OLTP and OLAP database installations, based on commercially available DBMSs and running on commodity hardware, can support actionable and reactive decisions when connected via an efficient ETL pipeline built on the principles of the Reactive Manifesto. Deploying the streaming ETL solution

and running concurrent OLTP and OLAP workloads using CH-benCHmark for repeatable and comparable benchmarks on different DBMSs is left as an area of future research.

ACKNOWLEDGMENT

This work is partly funded by the ENDEAVOUR Scholarship Scheme (Malta), part-financed by the European Union – European Social Fund (ESF) under Operational Programme II – Cohesion Policy 2014-2020.

REFERENCES

- Abadi, D. J., Madden, S. R., & Hachem, N. (2008). Column-stores vs. row-stores: how different are they really? *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 967–980. doi:10.1145/1376616.1376712
- Andurkar, A. D. (2012). Implementation of column-oriented database in PostgreSQL for optimization of read-only queries. *Computer Science and Information Technology*, 2(3), 437–452.
- Appuswamy, R., Karpathiotakis, M., Porobic, D., & Ailamaki, A. (2017). The case for heterogeneous HTAP. *8th Biennial Conference on Innovative Data Systems Research, CONF*.
- Arulraj, J., Perron, M., & Pavlo, A. (2016). Write-behind logging. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 10(4), 337–348. doi:10.14778/3025111.3025116
- Babeanu, R., & Ciobanu, M. (2015). In-memory databases and innovations in Business Intelligence. *Database Systems Journal*, 6(1), 59–67.
- Bach, M., & Werner, A. (2016). Hybrid column/row-oriented DBMS. In *Man—Machine Interactions 4* (pp. 697–707). Springer. doi:10.1007/978-3-319-23437-3_60
- Bonér, J., Farley, D., Kuhn, R., & Thompson, M. (2014). *The reactive manifesto*. Dosegljivo: [Http://Www.Reactivemanifesto.Org/](http://www.Reactivemanifesto.Org/)
- Butterstein, D., Martin, D., Stolze, K., Beier, F., Zhong, J., & Wang, L. (2020). Replication at the speed of change: A fast, scalable replication solution for near real-time HTAP processing. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 13(12), 3245–3257. doi:10.14778/3415478.3415548
- Cole, R., Funke, F., Giakoumakis, L., Guy, W., Kemper, A., Krompass, S., Kuno, H., Nambiar, R., Neumann, T., & Poess, M. et al. (2011). The mixed workload CH-benCHmark. *Proceedings of the Fourth International Workshop on Testing Database Systems*, 1–6.
- Council, T. P. (1999). *TPC Benchmark H, Revision 2.18*. <http://www.tpc.org/tpch/>
- Council, T. P. (2010). *TPC Benchmark C, Revision 5.11*. <http://www.tpc.org/tpcc/>
- Dhindsa, P. B. S. K. (2012). A comparative study of database systems. *International Journal of Engineering and Innovative Technology*, 1(6).
- Elnaffar, S., Martin, P., & Horman, R. (2002). Automatically classifying database workloads. *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, 622–624. doi:10.1145/584792.584898
- Goodhope, K., Koshy, J., Kreps, J., Narkhede, N., Park, R., Rao, J., & Ye, V. Y. (2012). Building LinkedIn's Real-time Activity Data Pipeline. *IEEE Data Eng. Bull.*, 35(2), 33–45.
- Gray, J., & Barkhatov, A. (1994). DBGen synthetic data generator for SQL tables and text files on Windows platforms. *Proc of SIGMOD*, 243–252.
- Halfpap, S., & Schlosser, R. (2020). Exploration of Dynamic Query-Based Load Balancing for Partially Replicated Database Systems with Node Failures. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 3409–3412.
- Harizopoulos, S., Abadi, D. J., Madden, S., & Stonebraker, M. (2018). OLTP through the looking glass, and what we found there. In *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker* (pp. 409–439). doi:10.1145/3226595.3226635
- HauchR. (2020). *Debezium*. <https://debezium.io/>
- Huang, D., Liu, Q., Cui, Q., Fang, Z., Ma, X., Xu, F., Shen, L., Tang, L., Zhou, Y., Huang, M., Wei, W., Liu, C., Zhang, J., Li, J., Wu, X., Song, L., Sun, R., Yu, S., Zhao, L., & Tang, X. et al. (2020). TiDB: A Raft-based HTAP database. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 13(12), 3072–3084. doi:10.14778/3415478.3415535
- Kafka, A. (2014). *A high-throughput distributed messaging system*. Kafka. Apache. Org

- Lamb, A., Fuller, M., Varadarajan, R., Tran, N., Vandiver, B., Doshi, L., & Bear, C. (2012). The vertica analytic database: C-store 7 years later. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 5(12), 1790–1801. doi:10.14778/2367502.2367518
- Lee, E. A. (2018). What Is Real Time Computing? A Personal View. *IEEE Design & Test*, 35(2), 64–72. doi:10.1109/MDAT.2017.2766560
- Lee, J., Han, W.-S., Na, H. J., Park, C. G., Kim, K. H., Kim, D. H., Lee, J. Y., Cha, S. K., & Moon, S. (2018). Parallel replication across formats for scaling out mixed OLTP/OLAP workloads in main-memory databases. *The VLDB Journal*, 27(3), 421–444. doi:10.1007/s00778-018-0503-z
- Li, L., Wu, G., Wang, G., & Yuan, Y. (2019). Accelerating Hybrid Transactional/Analytical Processing Using Consistent Dual-Snapshot. *International Conference on Database Systems for Advanced Applications*, 52–69. doi:10.1007/978-3-030-18576-3_4
- Lightbend. (2020). *An introduction to Reactive Streams, Akka Streams and Akka HTTP for Enterprise Architects*. <https://info.lightbend.com/rs/558-NCX-702/images/COLL-white-paper-akka-reactive-streams.pdf>
- Macyna, W., & Kukowski, M. (2020). Flash-Aware Storage of the Column Oriented Databases. *Fundamenta Informaticae*, 173(1), 47–72. doi:10.3233/FI-2020-1915
- Makreshanski, D., Giceva, J., Barthels, C., & Alonso, G. (2017). BatchDB: Efficient isolated execution of hybrid OLTP+ OLAP workloads for interactive applications. *Proceedings of the 2017 ACM International Conference on Management of Data*, 37–50. doi:10.1145/3035918.3035959
- Molina, J. M., Garcia, J. F., & Jiménez, C. K. (2018). Archer: An Event-Driven Architecture for Cyber-Physical Systems. *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 335–340. doi:10.1109/UCC-Companion.2018.00077
- Neumann, T., & Freitag, M. J. (2020). *Umbr: A Disk-Based System with In-Memory Performance*. CIDR.
- Ordonez, C., & Bellatreche, L. (2018). A Survey on Parallel Database Systems from a Storage Perspective: Rows Versus Columns. *International Conference on Database and Expert Systems Applications*, 5–20. doi:10.1007/978-3-319-99133-7_1
- Pavlo, A., Angulo, G., Arulraj, J., Lin, H., Lin, J., Ma, L., Menon, P., Mowry, T. C., Perron, M., & Quah, I. et al. (2017). Self-Driving Database Management Systems. *CIDR*, 4, 1.
- Pavlo, A., & Aslett, M. (2016). What's really new with NewSQL? *SIGMOD Record*, 45(2), 45–55. doi:10.1145/3003665.3003674
- Phillips, D. (2011). TPC-H dbgen. In *GitHub repository*. GitHub.
- Poggi, N., Carrera, D., Gavaldà, R., Ayguadé, E., & Torres, J. (2014). A methodology for the evaluation of high response time on E-commerce users and sales. *Information Systems Frontiers*, 16(5), 867–885. doi:10.1007/s10796-012-9387-4
- Prasaad, G., Cheung, A., & Suci, D. (2020). Handling Highly Contended OLTP Workloads Using Fast Dynamic Partitioning. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 527–542. doi:10.1145/3318464.3389764
- Psaroudakis, I., Wolf, F., May, N., Neumann, T., Böhm, A., Ailamaki, A., & Sattler, K.-U. (2014). Scaling up mixed workloads: a battle of data freshness, flexibility, and scheduling. *Technology Conference on Performance Evaluation and Benchmarking*, 97–112.
- Raab, D. (2007). How to Judge a Columnar Database. *Information & Management*, 17(12), 33.
- Raab, F. (1993). *TPC-C-The Standard Benchmark for Online transaction Processing*. In *The Benchmark Handbook* (2nd ed.). Morgan Kaufmann Publishers Inc.
- Raza, A., Chrysogelos, P., Anadiotis, A. C., & Ailamaki, A. (2020). Adaptive HTAP through elastic resource scheduling. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2043–2054. doi:10.1145/3318464.3389783

- Rompf, T., & Amin, N. (2019). A SQL to C compiler in 500 lines of code. *Journal of Functional Programming*, 29, 29. doi:10.1017/S0956796819000054
- Shen, L., Lou, Y., Chen, Y., Lu, M., & Ye, F. (2019). Meteorological Sensor Data Storage Mechanism Based on TimescaleDB and Kafka. *International Conference of Pioneering Computer Scientists, Engineers and Educators*, 137–147. doi:10.1007/978-981-15-0118-0_11
- Sridhar, K. T. (2017). Modern column stores for big data processing. *International Conference on Big Data Analytics*, 113–125. doi:10.1007/978-3-319-72413-3_8
- Stonebraker, M., & Rowe, L. A. (1986). *The design of Postgres* (Vol. 15, Issue 2). ACM.
- Stonebraker, M., & Cetintemel, U. (2005). “One size fits all”: an idea whose time has come and gone. *21st International Conference on Data Engineering (ICDE'05)*, 2–11. doi:10.1109/ICDE.2005.1
- Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., & O’Neil, E. (2005). C-store: a column-oriented DBMS. *Proceedings of the 31st International Conference on Very Large Data Bases*, 553–564.
- Stonebraker, M., Madden, S., Abadi, D. J., Harizopoulos, S., Hachem, N., & Helland, P. (2007). The end of an architectural era:(it’s time for a complete rewrite). *Proceedings of the 33rd International Conference on Very Large Data Bases*, 1150–1160.
- Strauch, C. (2011). *NoSQL databases. Lecture selected topics on software-technology ultra-large scale sites*. Stuttgart Media University.
- Vertica, H. P. E. (2021). *Vertica Database. Vertica Administration Guide*. <https://www.vertica.com>
- Yang, J., Rae, I., Xu, J., Shute, J., Yuan, Z., Lau, K., Zeng, Q., Zhao, X., Ma, J., Chen, Z., Gao, Y., Dong, Q., Zhou, J., Wood, J., Graefe, G., Naughton, J., & Cieslewicz, J. (2020). F1 Lightning: HTAP as a Service. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 13(12), 3313–3325. doi:10.14778/3415478.3415553