Hybrid Transactional and Analytical Processing Databases: A Systematic Literature Review

Daniel Hieber
Dept. of Computer Science
Aalen University
Aalen, Germany
Email: daniel.hieber@hs-aalen.de

Gregor Grambow
Dept. of Computer Science
Aalen University
Aalen, Germany
Email: gregor.grambow@hs-aalen.de

***Abstract*—Hybrid Transactional and Analytical Processing databases (HTAP, OLxP) are an emerging sector of databases combining Online Transactional Processing and Online Analytical Processing in the same system. Such databases yield many advantages like the reduction of the total cost of ownership or the elimination of redundant data sets for analytical and live data. Therefore, both Gartner and Forrester Research see disruptive potential in HTAP databases. While following a common goal, the database architectures of HTAP databases are quite diverse. The solutions range from scaled-up single server systems, using Multi Version Concurrency Control to keep their data consistent while at the same time executing thousands of queries, to scaled-out clusters of many servers using last writer wins approaches to allow even faster transactional processing. The development of HTAP databases resulted in various advances in the database sector like the creation of new index and data structures or improvements of existing concurrency control implementations. This paper provides a comprehensive summary of these implementations, giving an overview of the last decade of research on the emerging sector of HTAP Processing databases and discussing fundamental involved technologies.**

***Keywords*—Hybrid Transactional Analytical Processing; HTAP; Database; Literature Study; OLxP.**

I. Introduction

The need to analyse data in realtime and not to rely on copies of old databases combined with the growing wish of companies to gather all data in one database lead to the rise of Hybrid Transactional Analytical Processing, a term coined by Gartner [1] in 2014. But, even before that there has already been active research in the area. This systematic literature review summarizes the research on HTAP over the past 10 years.

Solving the problems of keeping data in two separated databases and at the same time reducing the total cost of ownership by introducing one unified system instead, HTAP efficiently combines Online Transactional Processing and Online Analytical Processing capabilities in one system. Therefore, both Gartner [2] and Forrester Research [3] see disruptive potential in HTAP.

In this review, the basics of the different fundamental architectures for HTAP database systems like HyPer [4] and SAP HANA [5] are explained. Further different approaches regarding the concrete implementations and optimization approaches are introduced. The aim of this systematic literature review is to reflect the current state of research in an ordered way, as well as to highlight important decisions leading to todays implementations.

This remainder of this paper is organized as follows: Section 2 provides background on database processing paradigms covered in this paper. Section 3 describes the underlying literature review process in detail. Section 4 discusses the findings and provides a comprehensive overview of the current development and research state of HTAP. Finally, Section 5 summarizes the provided work, supplying all required information in a short form.

II. Background

This section provides some background information regarding the database processing paradigms covered in this paper.

A. Online Transaction Processing

Online Transaction Processing (OLTP) describes a category of data processing that is focused on transaction-oriented tasks. The workload is heavily write oriented, consisting of insert, update and delete operations. The size of data involved is usually relatively small, while the amount of transactions can be massive.

Features like normalization and ACID (Atomicity, Consistency, Isolation, Durability) are required by OLTP to function efficiently. Besides fast processing and highest availability, data consistency is also one of the most important features of OLTP databases.

B. Online Analytical Processing

Online Analytical Processing (OLAP) is focused on complex queries for dataset analysis. The workload is read heavy and can include enormous datasets. In order to efficiently analyse such big amounts of data, intelligent indexing and fast read times are necessary. OLAP workloads are resource heavy and require high performance systems.

C. Hybrid Transactional Analytical Processing

Hybrid Transactional Analytical Processing (HTAP) combines both OLTP and OLAP in one database. Therefore, writing and analyzing data is efficiently handled in the same database, removing the need to run two separate systems and thereby reducing implementation efforts, maintenance and cost. However, the resource intensive workload of OLAP

 ISBN: 978-1-61208-816-7

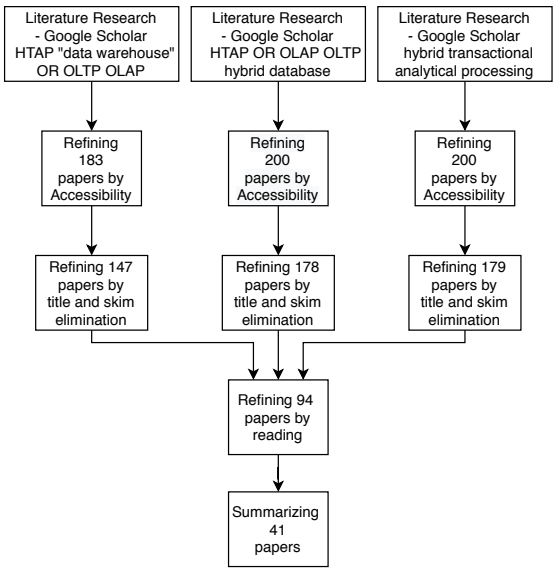

Figure 1. Literature Review Process.

queries and the required high availability of OLTP compete with each other and require new solutions to work on the same system.

III. Literature Review Methodology

Despite the existence of the term HTAP since 2014, many researchers still did not adopt it. To ensure a comprehensive and high-quality literature base for the review, several searches were carried out.

In the study, Kitchenham's systematic review procedure [6] was employed. The following steps were pursued:

1) Determining the topic of the research
2) Extraction of the studies from literature considering exclusion and inclusion criteria
3) Evaluation of the quality of the studies
4) Analysis of the data
5) Report of the results

The reviewing process (Figure 1) was conducted via Google Scholar as this search engine provided far more results than other search engines, which also included most results of the other engines. Searches with other engines did not return the desired quantity of material with the used queries, preventing a sophisticated review of the topic.

Using Google Scholar, the volume was appropriate but the quality was still lacking, mainly because the term HTAP is still not used by all research conducted on this topic. To counter this, three different searches were conducted all using different search terms to gain a sufficient literature base regarding quantity and quality. Quality in this context refers to the overall consistency of the provided content and the adherence to scientific standards.

The search was carried out using (1) "htap" "data warehouse" OR "OLTP" "OLAP" (returning 183 entries), (2) HTAP OR OLAP OLTP hybrid database (returning 200 entries) and (3) hybrid transactional analytical processing (returning 200 entries). Only publications from 2010 or later were considered. The latter queries returned more papers, but were reduced to the 200 most recommended papers, since quality and relevance were continuously decreasing.

Only papers accessible without additional fees and written in German or English were taken into account. Further, theses were not considered in this paper. These criteria left 147 (1), 178 (2) and 179 (3) papers to refine further. With title and abstract based elimination, the papers lacking a combination of required key words or only mentioning HTAP as a side note were excluded. After that step, 55 (1), 44 (2) and 56 (3) papers were left for further analysis. This leaves a total of 94 papers (deducted duplicated paper from the different queries) for a final review.

Of these 94 papers, 15 were found to be of insufficient quality and 19 did not focus on the topic of HTAP databases or on fundamental technologies for those. The 60 papers, which were found scientifically significant and fulfilling the quality requirements were finally reduced to 41, deducting papers providing only outdated non-fundamental information.

IV. Findings and Discussion

The methods to create HTAP databases, their functionality and their optimizations take many different approaches. The contents of the papers were organized into the following sections according to the kind of information provided.

A. Fundamental Architecture

HTAP databases build up a new database sector and there are many databases which were newly developed for this workload, e.g., [4][7][8]. However, some existing databases also have been upgraded to handle HTAP workloads like SAP HANA [5], initially an OLAP database, and PostgreSQL [9], initially an OLTP database, proving that existing databases can be extended to handle HTAP.

Comparing the reviewed database architectures, two main storage paradigmas can be clearly identified with the reviewed solutions: (1) heavily main memory focused databases, keeping all of their (hot) data in memory like HANA [10], HyPer [4], BatchDB [8] and Hyrise [11], as well as (2) cloud/shared disk data stores, keeping some data in memory but relying on a persistent out of memory data store accessible by all instances, e.g., Wildfire [12] and Janus [13].

Further, a Non-Uniform Memory Access (NUMA) architecture is a base requirement for most main memory HTAP databases like SAP HANA [10], AIM [14], BatchDB [8], Hyrise [11] and HyPer [4] enabling multiple cores to access each others memory.

1) Scaling out and up: Another big difference in HTAP databases is their scaling approach. Systems like HyPer [4] (commercialized by Tableau) or Hyrise [11] are deployed on single servers utilizing NUMA to scale-up onto multiple cores, thus creating multiple nodes. This approach can reduce processing time as no data transfer between different servers

 ISBN: 978-1-61208-816-7

is required and all data can be accessed in memory. As a downside however, large systems require a strong server with a large main memory. Both HyPer and Hyrise also provide scale-out approaches, normally keeping their OLTP processing on the main server, e.g., ScyPer [15].

Like Hyrise and HyPer - SAP HANA [16] keeps the OLTP workload on one machine, utilizing NUMA to use as many cores as required and available, but implements scaling the OLTP workload out to other servers as a base feature. Using HANA Asynchronous Parallel Table Replication (ATR) the database distributes its data amongst multiple replicas enabling a more efficient OLAP approach.

BatchDB [8] also handles the OLTP workload on the main server. The OLAP workload can be either executed on a different node of the same machine, or an entirely different server.

Contrarily, Wildfire [17] (commercialized as IBM BD2 Event Store) utilizes a fully distributed approach. Heavily relying on Apache Spark, all requests pass Sparks API and get distributed across multiple Spark executors. These executors delegate the transactional and analytical requests to the Wildfire engine daemons. All daemons use their main memory as well as solid-state drives (SSDs) and are connected to one shared data storage, e.g., a cloud data store. With this approach more throughput can be achieved, but ACID on the other hand is no longer possible. The latest research on the Wildfire system, Wildfire-Serializable (WiSer) [18] also offers high availability besides HTAP. It is furthermore optimized for Internet of Things workloads.

Like Wildfire, SnappyData [19] also uses Spark as a core component to scale out the system to a database cluster. Therefore, the system enables more information to be kept in memory without the need for one expensive server.

Janus [13] also uses a distributed setup but implements the query distribution on its own with execution servers. These delegate the query to a corresponding row partitioned server for OLTP workloads or a column partitioned server for OLAP workloads.

2) Data/Table Structure: When dealing with OLTP and OLAP workloads, finding the right table format can be difficult. HTAP databases therefore employ different table and data structures. Wildfire [17] exclusively uses column oriented tables since they are the most efficient solution for OLAP workload.

SAP HANA [10] implements a row-store query engine and a column-storage engine to combine the advantages of both technologies. Thus, it is possible to save data in row or column tables. The column layout is the default, more optimized, option.

HyPer [20] and Hyrise [11] both use columnar stores with self implemented data models. Hyrise further presented a hybrid column layout in an older version [21], combining simple one-attribute-columns with rows. This is planned to be implemented again in the new version, but has low priority and is work in progress.

Opposed to this, PostgreSQL [9] continues to use its row data storage for OLTP, but has a column store extension for OLAP workloads, merging the delta from the row store continuously in the column store.

SnappyData [19] follows a hybrid approach, where the fresh data is stored in an in-memory row-store and is moved in an on-disk column-store after aging.

The Cloud data store Janus [13] is fully hybrid, utilizing row partitions for OLTP and column partition for OLAP. Via a redo-log inspired batching approach and a graph-based dependency management, the delta from the row replicas can be merged into the column replicas.

The Casper prototype [22] uses a tailored column layout to support mixed read/write workloads more efficiently. With this approach, runtime column adaptations are possible.

Flexible Storage Model (FSM) [23] presented a tile based architecture to allow a transition from OLTP optimized tables to OLAP optimized tables depending on the hotness of data. The data is saved in a row oriented manner at the beginning and, depending on the hotness, is tile-wise transitioned to an OLAP column oriented tile structure.

3) Saving and Partitioning Data: For scale-up focused databases, removing data from main memory to larger, more cost efficient stores (e.g., hard drives), or efficiently compressing its size, is crucial. HyPer uses horizontal partitioning and saves its hot data uncompressed on the main memory. The cold data can also be kept in memory. Instead of evicting data to a disk, the data is compressed into self implemented Data Blocks [20] and kept in main memory. However, it is possible to evict them to secondary storage solutions if preferred (e.g., Non-volatile random-access memory) and use them as persistent backups. The compression technique is chosen based on the data actually saved in the Data Block.

Utilizing Small Materialized Aggregates (SMAs) including meta data like min and max values, irrelevant compressed data can easily be skipped in searches. If data cannot be skipped on SMA basis, Positional SMAs (PSMAs), another lightweight indexing structure developed by the HyPer team, can be used. These help to determine the range of positions in the Data Block were the relevant values are located.

Hyrise [21] solves this problem using horizontal partitioning and by saving data in 2 kinds of columns: Memory-Resident Columns for hot data in memory, allowing fast access, and uncompressed row-oriented Secondary Storage Column Groups for cold data on hard drives. As the cold data is saved uncompressed, the cost of accessing it is reduced in comparison to classical compressed approaches.

Furthermore, the data is organized in so-called chunks [11] similar to Data Blocks. Chunks can be mutable as long as they are not full. As soon as they reach their capacity they transition to an immutable append-only container. They also have indexes and filters on a per chunk basis like Data Blocks, allowing faster search and access operations.

Smart Larger Than Memory [24] stores cold data in files on the hard drive decoupled from the database. Modifications to the data are no longer possible. The entries can only be

 ISBN: 978-1-61208-816-7

deleted. This happens via removing the reference entry in memory without accessing the cold data and thereby saving time. Updating cold data is possible, but the update is a hidden delete of the cold data index and an insert of new hot data. To fully take advantage of SmartLTM the read operations always check the main memory entries first. If the data cannot be found, cuckoo filters or SMAs are used to locate the data in the files on the hard drive.

Finally, partitioning workloads in an intelligent manner without extra statistical data structures is possible, too. As presented by Boissier and Kurzynski [25], physical horizontal data partitioning as well as the adapted aggressive data skipping approach can skip up to 90% of data on OLAP queries.

B. Concurrency

Handling multiple versions of data is a crucial part of all HTAP databases. Current OLTP and OLAP operations require a solution to parallelize data access.

The most common approach is Multi Version Concurrency Control (MVCC). It is utilized in combination with a delta by PostgreSQL [9], SAP HANA [10] and in new versions of HyPer [26].

Hyrise [27] is also using MVCC, but is following a look free commit approach, replacing the delta.

SnappyData and BatchDB [8] also use MVCC oriented approaches. SnappyData [7] relies on GemFire to handle concurrent access and snapshots, while BatchDB [8] uses MVCC on its OLTP replica, while updating the isolated OLAP replica batch-wise.

Although HyPer is now using MVCC with delta, it initially used the fork systemcall to create multiple isolated in-memory snapshots [28]. Utilizing a copy on write approach to reduce memory consumption OLAP queries could be executed on snapshots while the OLTP operations updated the main memory entries.

In addition to the MVCC on its main OLTP replica, SAP HANA [16] further uses ATR with a replication log system to synchronize its multiple server architecture. This synchronizes data with sub-second visibility delay between the replicas.

Wildfire [17] chooses speed over concurrency as already mentioned. Therefore, a simple last writer wins approach is used by the Wildfire engine.

While most systems nowadays use some implementation of MVCC, research on more efficient snapshotting techniques is still being carried out, e.g., [29]. Inspired by earlier HyPer implementations, another research project on snapshotting, AnKer [30], uses a customized Linux kernel with an updated fork system call. This updated fork, called vm_snapshot, enables high frequency snapshotting. Through vm_snapshot the researchers are able to snapshot only the used columns. This significantly outperforms the default fork used initially by HyPers implementation, providing a possible alternative to MVCC systems.

Wait free HTAP (WHTAP) [31] utilizes snapshotting for concurrency as well. In this dual snapshot engine approach data for OLAP and OLTP are stored in different replicas, using a five state process and two deltas. In this process, the deltas form the OLAP and OLTP replicas are switched and the old OLTP delta is merged into the OLAP replica which takes effect without slowing the analytical queries down.

C. Garbage Collection

MVCC implementations require performant garbage collection to prevent large amounts of versions to slow down the transactions on the database. SAP HANA [10] uses timestamps and visibility bits to track versions of their data. Data gets created/edited with a timestamp. When all active transactions can see this version the timestamp is replaced with a bit indicating the visibility. If the row is no longer visible to any snapshot, it can be deleted with the next delta merge.

HyPers garbage collector Steam [26] follows a similar approach. The main difference is that the garbage collector is called with every new transaction instead of being a background task like with SAP HANA. This approach called eager pruning removes all versions not required by any transaction. This happens by checking every time the chain is extended whether all versions included in the version chain are used by a transaction. With eager pruning the version chain can only be as long as the amount of different queries.

Due to its heavily distributed architecture with many data sets saved in main memory and SSDs on the different servers, Wildfire follows a different solution and implements a lazy garbage collection approach [17]. When performing lazy garbage collection, data is only deleted if there is no possibility that a query could require it. However, the concrete implementation is not explained.

D. Query Handling

The ways to access the concurrent data differ significantly from database implementation to implementation.

1) Query Handling in Scale-up Systems: The systems HyPer [32] and Hyrise [11], primarily engineered for scale-up solutions working on one dataset, implemented the query operators as C++ code in their database. The missing variables are inserted via just in time compilation. After the insertion, the code is compiled to LLVM assembler code, allowing fast query execution. As mentioned before, the two databases also have prototype scale-out options, but focus on the scale-up approach.

Another approach is to dynamically schedule memory and computing resources actively [33]. With this approach, the cores are assigned to the OLAP or OLTP workload as required, always trying to maximize productivity and the database throughput.

2) Query Handling in Scaled-out Systems: Scale-out systems are separated in two groups: On the one hand, systems keeping the OLTP workload on one server, scaling only the OLAP workload to other servers, as e.g., SAP HANA [16] or BatchDB [8]. On the other hand, systems distributing OLTP and OLAP workloads over multiple servers, e.g., Wildfire [12][17] and SnappyData [7][19].

 ISBN: 978-1-61208-816-7

BatchDB [8] and HANA [16] both handle their OLTP workload on a single server, scaling-up via NUMA as described earlier. For OLAP workloads they are able to scale out onto multiple servers working on replicas of the main data.

Wildfire [12] and SnappyData [19] contrarily scale out via Apache Spark, allowing OLTP and OLAP transactions to be executed on a cluster of nodes dealing with big data and streaming workloads. Wildfire [12] executes OLTP queries on the fresh data on Wildfire daemons. OLAP workloads can be executed via Spark Executor as requests to the daemons or directly accessing the shared data of the Wildfire database cluster. With this approach, old data can be consumed from the shared file system without slowing down OLTP throughput while the latest data can still be received if required.

3) Query Language: While the databases offer many new functionalities to access and modify data, Structured Query Language (SQL) is still commonly supported. The database systems SAP HANA [5], Wildfire [17], Hyrise [11], HyPer [4], SnappyData [7] and AIM [14] all enable basic SQL queries to interact with the database. However, many of them further provide new optimized ways to interact with the data.

Wildfire [17] and SnappyData [7] provide data access via an extended version of the SparkSQL API. SnappyData also further extends the Spark Streaming API.

SAP HANA [5] provides more specific access through SQL Script and Multidimensional Expressions (MDX). The database is also is natively optimized for the ABAP language and runtime. This allows to bypass the SQL connectivity stack by directly accessing special internal data representations via Fast Data Access (FDA). The Native For All Entries (NFAE) technique further modifies the ABAP runtime to allow even more performance improvements.

Hyrise [11] provides a command-line interface which allows SQL queries but also provides additional visualization and management functions. Furthermore, the wire protocol of PostgreSQL allows access through common PostgreSQL drivers and clients.

HyPer [32] uses HyPerScript as its query language. HyPerScript is a SQL-based query language and therefore allows base SQL statements as well. The features consist of passing whole tables as query parameters and providing the possibility to use query results in a later part of the query, removing the need to query the same value multiple times.

E. Indices

To allow efficient data access and querying on multiple servers and/or different versions of data, the right index structure is of special importance in HTAP databases. Wildfire's multi-version multi-zone index Umzi [34] employs a LSM-like structure with multiple runs. It divides index runs in multiple zones and implements efficient evolve operations to handle zone switches of data. Further Umzi uses a multi-tier storage using SSDs and memory caching with self-updating functionality for fast execution while persisting the indexes on Wildfires shared data.

HyPer developed the Adaptive Radix Tree (ART) [32] based on the radix tree. ART uses four different node types that can handle 4, 16, 48 and 256 entries. The maximum height for the tree is k for k-byte trees. To further reduce the tree height and required space, the tree is build lazily, saving single leaf branches higher in the tree. Additionally, path compression is used to remove common paths and to insert them as a prefix of the inner node thereby removing cache inefficient one-way node chains.

SAP HANA [10] and Hyrise [11] both use B-Trees. Hyrise further supports the ART index from HyPer [32] and a group-key-index, implemented by the Hyrise project.

BatchDB utilizes a simplified version of the look-free Bw-Tree [8]. The version relies on atomic multi-word compare-and-swap updates.

In 2019, a predictive indexing approach [35] was introduced to cope with the dynamic demands of a HTAP database. Predictive indexing increases the throughput by up to 5%. In this approach, a machine learning system calculates the optimal index structure for the data according to the workload.

The Multi-Version Partitioned B-Tree (MV-MBT) [36] is another recent research in the indexing sector for HTAP databases from 2019. This extension of partitioned B-Tree creates a version aware index, able to maintain multiple partitions within a single tree structure, sorted in alphanumeric order.

Likewise proposed in 2019, the Parallel Binary Tree (P-Tree) [37] is an extension of a balanced binary tree relying on copy-on write mechanisms to create tree copies on updates. With this approach, the indices become the version history without requiring other data structures.

F. Big Data on HTAP Databases

Wildfire was created with big data as its primary use case [17]. Through the distributed design of its big data platform, Wildfire is able to concurrently handle high-volumes of transactions as well as execute analytics on latest data. At the same time, the system is able to scale onto many machines because of its close integration of Apache Spark. The usage of an open data format further enables compatibility with the big data ecosystem. Nowadays, the commercial version IBM Db2 Event Store is capable of handling more than 250 billion events per day [38]. SnappyData [19] is an analogically capable big data platform with an architecture similar to Wildfire.

The SAP HANA database can be used as part of the SAP HANA data platform to handle big data workloads [39]. Using a combination of different SAP products, namely SAP Synbase ESP and SAP Synbase IQ, as well as smart data access frameworks as Hadoop, Teradata or Apache Spark, the SAP HANA data platform is a fully functional big data system with SAP HANA in its core.

HyPerInsight [40] provides big data capabilities in the area of data exploration on the HyPer database. The goal is to minimize the required user expertise with the dataset while simultaneously supporting the user with the formulation of queries. The support for lambda functions in SQL queries

 ISBN: 978-1-61208-816-7

allows user defined code to be executed within the queries. In combination with the HTAP HyPer system as the database, data mining on real-time data is possible.

G. Recovery and Logging

As many HTAP databases rely on volatile main memory as primary storage and the other systems utilize distributed data sets, recovery in case of failure is of special importance. Data loss has to be prevented and downtime must be minimized.

SAP HANA instances log data persistently on the local drive for recovery on failure or restart purposes [5]. The logging approach is inspired by SAP MaxDB.

As already explained, HANA works with ATR in its distributed architecture [16]. Following the store-and-forward approach, the data is replicated to multiple servers. An algorithm then compares the record version IDs of the incoming data and stored data, requesting the resend of lost log entries if deviations occur.

Recovery for the latest version of Hyrise is still work in progress [11], but recovery for older versions of Hyrise was explained [27]. The database dumps the main partition of the table as a binary dump on the disk and records the delta to a log via group commits to hide the latency. At checkpoints, the delta partitions are also saved as a binary dump on the drive. If recovery is required, the main dump and delta dump from a checkpoint are restored and an eventually existing delta log is replayed on the table, restoring the old state.

BatchDB logs successful transactions on its OLTP replica in batches via command logging on durable storage [8]. In case of a failure, the database can recover from these logs. The OLAP replica itself has no durable logging and has to recover from the main OLTP replica on failure.

SnappyData [7] uses Apache Sparks logging and recovery mechanisms, logging transformations used to build Sparks Resilient Distributed Datasets (RDDs). Saving RDDs to storage is also possible. In SnappyData the combination with GemFire however allows Spark to save the RDDs in GemFires storage instead of the persistent storage of the server. Small recoveries can be handled directly by GemFires eager replication, leaving batched and streaming recovery to Spark, in combination with the GemFire storage.

Further, a peer-to-peer (p2p) approach is used in SnappyData clusters. Any in-memory data can be synchronously replicated from the cluster. Additional to the replication via the p2p approach, data is always replicated to at least one other node in the cluster.

H. Benchmarking

The combination of OLTP and OLAP workloads on one database also created the need for new benchmarks covering this sector. In 2011, CH-benCHmark [41] was introduced. The CH-benCHmark is based on the TPC-C and TPC-H benchmarks. It executes a transactional and analytical workload in parallel on a shared set of tables on the same database. The benchmark can also be used for single workload databases.

In 2017, HTAPBench [42] was published. This benchmark is able to compare OLTP, OLAP and hybrid workloads on the database. Its main difference to CH-benCHmark lies in its Client Balancer, controlling the coexisting OLAP and OLTP workloads.

Hyrise [11] implements a special benchmark runner to easily execute benchmarks.

I. Stream Processing

Streaming as a special case of OLTP is an emerging use case for HTAP database systems. In 2016 scientists from ETH Zürich in cooperation with Huawei presented AIM [14], which is a high performance event-processing and real time analytics HTAP database. The three-tiered multi node system processes events at one tier, stores the data at a central tier and finally analyses the processed data in real time on the third tier. AIM however, is optimized for a special streaming use case from the telecommunications industry.

In early 2019, the research team around HyPer compared modified versions of HyPer with AIM and Apache Flink [43] in order to determine the current state of streaming capabilities of main memory database systems (MMDB). While MMDBs are still inferior to dedicated streaming frameworks like Flink, the HyPer team was confident, that HTAP databases could catch up with some adjustments, even implementing some of those on HyPer. The main areas requiring improvement are network optimization, parallel transaction processing, skew handling and a strong distributed architecture.

SnappyData aims to solve OLTP, OLAP and streaming all in one product [19] with their tight integration of Apache Spark and GemFire. In an evaluation, SnappyData was able to outperform both Spark on TPC-H queries as well as MemSQL on all kinds of throughput. The focus with SnappyDatas stream processing lies on complex analytical queries on streams, which are not possible with default stream processor solutions [7].

SAPs approach for big data, SAP Big Data [39], supports streaming as well. Since it uses other SAP products to achieve it and is not a part of the base SAP HANA database infrastructure, but rather is built on top of it, it is not further discussed in this paper.

J. Future

HTAP databases are a new sector which has evolved over the past 10 years. On an annual basis, companies and researchers contribute new ideas to lift their database above the competition. Currently, the newest trends tend to lead into three directions:

1. Heterogenous HTAP (HHTAP/H2TAP): with new hardware supporting heterogenous parallelism, HHTAP becomes a possibility. In this approach, CPUs and GPGPUs can access shared memory and divide the workload between both. Complex OLAP queries are solved on the GPGPUs, leaving the OLTP workload for the CPUs. The Caldera [44] prototype proved the feasibility for HHTAP. Early 2020, the data store GridTables [45] was published and proved the concept again.

 ISBN: 978-1-61208-816-7

However, in their summary, the authors of GridTables pointed out that there are still many research issues left to be solved. Further, early in 2020, a paper was published about GPU accelerated data management [46] explaining how to fully exploit hardware isolation between CPUs and GPUs and presenting a SemiLazy access method to reduce the required data transfer.

2. Streaming workloads: as described in detail in the previous section, stream processing is a possible use case for HTAP databases. Because of the optimization for high OLTP throughput and the ability to analyse these data streams in the same system, HTAP databases are an emerging alternative to current stream processing solutions. While still inferior to dedicated stream processors, the research on such solutions saw an increase in interest over the last years, e.g., by the HyPer team [43] and dedicated streaming HTAP databases like SnappyData [19] and AIM [14].

3. Optimization: while the bigger part of the 2010s was spent on researching for new systems [4][5][27], the last quarter focused on their optimization. Few new database systems were proposed and research started optimizing existing systems even further [26][30][37].

Further solutions using machine learning are slowly emerging. These allow databases to adapt on their own according to current workload and requirements. However, there is still not enough research to speak of an own trend and it can rather be viewed as another kind of optimization research. An example for such research is the presented predictive indexing approach [35].

K. Open Source and Free Versions

Some of the database systems summarized in this paper provide open source and/or free solutions. SnappyData [47] is available with a getting started guide covering the basic usage. The source code can be found on GitLab. The project is licensed with the Apache License, Version 2.0.

MemSQL [48], is available, well documented and can be used for smaller projects up to 4 nodes for free. Many extensions are available at the official GitHub account.

Hyrise [49] is available under the MIT license. However, as it is a research database, breaking changes may occur more frequently.

The free MemSQL version and a basic SnappyData setup can already be used on low end systems, naming 8GB main memory as their minimal requirement to operate efficiently.

To try a HTAP database without a setup process, HyPer can be used. The HyPer research version is provided as a simple web tool for exploration and testing [50]. This version, however, is running on a low end system and cannot be used in production.

V. Conclusion

In this paper, we have shown that HTAP databases are nowadays serious alternatives to traditional database solutions. The existence of a market for commercial products like SAP HANA, IBM Db2 Event Store and Tableau further reinforces our findings. Moreover, we have highlighted differences of existing approaches regarding key properties like the fundamental architecture, concurrency, or big data capabilities. Thus, this study can aid both researchers and practitioners in the process of selecting a matching HTAP solution. Finally, by providing a comprehensive overview of current research approaches as well as productive solutions, this study helps to identify trends and point out directions for future research. The following paragraphs provide a brief summary of our findings.

Open source and free HTAP products place HTAP databases on the same level as current database systems, allowing the integration in other products and exploring this new technology without financial risks.

The combination of OLTP and OLAP queries on one database efficiently reduces the total cost of ownership and allows a narrower tech stack for companies. The possibility to analyse data in real time further validates HTAP databases as a productive solution with a great added value compared to conventional databases.

Many different implementations, providing different advantages, are available and can be used as required by the costumer. Solutions using main memory as a primary/sole storage as well as solutions relying on shared data storages exist and are both valid options. Powerful single server database systems allow a slim tech stack while still being faster than most traditional OLTP and OLAP optimized databases. Distributed multi server clusters allow more fail-safe and easier to scale solutions, while at the same time requiring less performant machines. SnappyData and MemSQL, for example, can already be executed on machines with 8GB of memory, scaling up from there.

Over the last years, new indices, filters, data structures and replication techniques were developed, optimizing performant HTAP systems even further. The future seems to be heading in three main directions: HHTAP - utilizing new heterogenous hardware to include the GPUs in HTAP databases and allow even more efficient architectures.

Streaming - HTAP databases optimized for streaming are making a combination with external stream processors unnecessary, further reducing the total cost of ownership and reducing the size of the required tech stack.

Optimization - while the bigger part of the 2010's was spent on developing the base technologies and databases themselves, the last quarter was primarily spent on optimization, still leaving much room for improvement.

Machine learning for self adapting databases also could be an emerging sector in the future, but currently there is not enough research in this direction to call it a trend.

References

[1] R. Nigel, F. Donald, P. Massimo, and E. Roxane, "Hybrid transaction/analytical processing will foster opportunities for dramatic business innovation," 2014.

[2] U. Joseph *et al.*, "Predicts 2016: In-memory computing-enabled hybrid transaction/analytical processing supports dramatic digital business innovation," 2015.

[3] Y. Noel and G. Mike, "Emerging technology: Translytical databases deliver analytics at the speed of transactions," 2015.

 ISBN: 978-1-61208-816-7

[4] A. Kemper and T. Neumann, "Hyper: A hybrid oltp olap main memory database system based on virtual memory snapshots," in *2011 IEEE 27th International Conference on Data Engineering*, 2011, pp. 195–206.

[5] F. Färber *et al.*, "The sap hana database - an architecture overview," *Bulletin of the Technical Committee on Data Engineering / IEEE Computer Society*, vol. 35, no. 1, pp. 28–33, 2012.

[6] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, 08 2004.

[7] B. Mozafari, "Snappydata," in *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya, Eds. Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-319-63962-8_258-1

[8] D. Makreshanski, J. Giceva, C. Barthels, and G. Alonso, "Batchdb: Efficient isolated execution of hybrid oltp+olap workloads for interactive applications," *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017.

[9] M. Nakamura *et al.*, "Extending postgresql to handle olxp workloads," in *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 2015, pp. 40–44.

[10] N. May, A. Böhm, and W. Lehner, "Sap hana – the evolution of an in-memory dbms from pure olap processing towards mixed workloads," in *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, B. Mitschang, D. Nicklas, F. Leymann, H. Schöning, M. Herschel, J. Teubner, T. Härder, O. Kopp, and M. Wieland, Eds. Gesellschaft für Informatik, Bonn, 2017, pp. 545–546.

[11] M. Dreseler *et al.*, "Hyrise re-engineered: An extensible database system for research in relational in-memory data management," in *EDBT*, 2019.

[12] R. Barber *et al.*, "Evolving databases for new-gen big data applications," in *CIDR*, 2017.

[13] V. Arora, F. Nawab, D. Agrawal, and A. E. Abbadi, "Janus: A hybrid scalable multi-representation cloud datastore," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 689–702, 2018.

[14] L. Braun *et al.*, "Analytics in motion: High performance event-processing and real-time analytics in the same database," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 251–264. [Online]. Available: https://doi.org/10.1145/2723372.2742783

[15] T. Mühlbauer, W. Rödiger, A. Reiser, A. Kemper, and T. Neumann, "Scyper: elastic olap throughput on transactional data," in *DanaC '13*, 2013.

[16] J. Lee *et al.*, "Parallel replication across formats in sap hana for scaling out mixed oltp/olap workloads," *Proc. VLDB Endow.*, vol. 10, no. 12, p. 1598–1609, Aug. 2017. [Online]. Available: https://doi.org/10.14778/3137765.3137767

[17] R. Barber, V. Raman, R. Sidle, Y. Tian, and P. Tözün, *Wildfire: HTAP for Big Data*. Germany: Springer, 2019.

[18] R. Barber *et al.*, "Wiser: A highly available HTAP DBMS for iot applications," in *2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, December 9-12, 2019*. IEEE, 2019, pp. 268–277. [Online]. Available: https://doi.org/10.1109/BigData47090.2019.9006519

[19] R. Jags *et al.*, "Snappydata: Streaming, transactions, and interactive analytics in a unified engine," 2016.

[20] H. Lang *et al.*, "Data blocks: Hybrid oltp and olap on compressed storage using both vectorization and compilation," in *SIGMOD '16*, 2016.

[21] M. Boissier, R. Schlosser, and M. Uflacker, "Hybrid data layouts for tiered htap databases with pareto-optimal data placements," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 2018, pp. 209–220.

[22] M. Athanassoulis, K. S. Bøgh, and S. Idreos, "Optimal column layout for hybrid workloads," *Proc. VLDB Endow.*, vol. 12, no. 13, p. 2393–2407, Sep. 2019. [Online]. Available: https://doi.org/10.14778/3358701.3358707

[23] J. Arulraj, A. Pavlo, and P. Menon, "Bridging the archipelago between row-stores and column-stores for hybrid workloads," in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 583–598. [Online]. Available: https://doi.org/10.1145/2882903.2915231

[24] P. R. P. Amora, E. M. Teixeira, F. D. B. S. Praciano, and J. C. Machado, "Smartltm: Smart larger-than-memory storage for hybrid database systems," in *SBBD*, 2018.

[25] M. Boissier and D. Kurzynski, "Workload-driven horizontal partitioning and pruning for large htap systems," in *2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW)*, 2018, pp. 116–121.

[26] J. Böttcher, V. Leis, T. Neumann, and A. Kemper, "Scalable garbage collection for in-memory mvcc systems," *Proc. VLDB Endow.*, vol. 13, no. 2, p. 128–141, Oct. 2019. [Online]. Available: https://doi.org/10.14778/3364324.3364328

[27] D. Schwalb, M. Faust, J. Wust, M. Grund, and H. Plattner, "Efficient transaction processing for hyrise in mixed workload environments," in *IMDM@VLDB*, 2014.

[28] F. Funke, A. Kemper, T. Mühlbauer, T. Neumann, and V. Leis, "Hyper beyond software: Exploiting modern hardware for main-memory database systems," *Datenbank-Spektrum*, vol. 14, no. 3, pp. 173–181, 2014.

[29] L. Li *et al.*, "A comparative study of consistent snapshot algorithms for main-memory database systems," *ArXiv*, vol. abs/1810.04915, 2018.

[30] A. Sharma, F. M. Schuhknecht, and J. Dittrich, "Accelerating analytical processing in mvcc using fine-granular high-frequency virtual snapshotting," in *Proceedings of the 2018 International Conference on Management of Data*, ser. SIGMOD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 245–258. [Online]. Available: https://doi.org/10.1145/3183713.3196904

[31] L. Li, G. Wu, G. Wang, and Y. Yuan, "Accelerating hybrid transactional/analytical processing using consistent dual-snapshot," in *Database Systems for Advanced Applications*. Cham: Springer International Publishing, 2019, pp. 52–69.

[32] K. Alfons *et al.*, "Transaction processing in the hybrid oltp&olap main-memory database system hyper," *IEEE Computer Society Data Engineering Bulletin*, vol. Special Issue on "Main Memory Databases", 2013.

[33] A. Raza, P. Chrysogelos, A. G. Anadiotis, and A. Ailamaki, "Adaptive HTAP through elastic resource scheduling," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2020, pp. 2043–2054. [Online]. Available: https://doi.org/10.1145/3318464.3389783

[34] C. Luo *et al.*, "Umzi: Unified multi-zone indexing for large-scale htap," in *EDBT*, 2019.

[35] A. Joy, X. Ran, M. Lin, and P. Andrew, "Predictive indexing," 2019.

[36] C. Riegger, T. Vinçon, R. Gottstein, and I. Petrov, "Mv-pbt: Multi-version index for large datasets and htap workloads," *ArXiv*, vol. abs/1910.08023, 2020.

[37] Y. Sun, G. Blelloch, W. S. Lim, and A. Pavlo, "On supporting efficient snapshot isolation for hybrid workloads with multi-versioned indexes," *Proc. VLDB Endow.*, vol. 13, pp. 211–225, 2019.

[38] IBM, "Ibm db2 event store," last visited: 12.10.2020. [Online]. Available: https://www.ibm.com/de-de/products/db2-event-store

[39] N. May *et al.*, "Sap hana - from relational olap database to big data infrastructure," in *EDBT*, 2015.

[40] N. Hubig *et al.*, "Hyperinsight: Data exploration deep inside hyper," *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017.

[41] R. L. Cole *et al.*, "The mixed workload ch-benchmark," in *DBTest '11*, 2011.

[42] F. Coelho, J. Paulo, R. Vilaça, J. Pereira, and R. Oliveira, "Htapbench: Hybrid transactional and analytical processing benchmark," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, ser. ICPE '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 293–304. [Online]. Available: https://doi.org/10.1145/3030207.3030228

[43] A. Kipf *et al.*, "Scalable analytics on fast data," *ACM Trans. Database Syst.*, vol. 44, no. 1, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3283811

[44] A. Raja, K. Manos, P. Danica, and A. Anastasia, "The case for heterogeneous htap," *8th Binnial conference on Innovative Data Systems Reseach (CIDR '17)*, 2017.

[45] M. Pinnecke, G. Campero Durand, D. Broneske, R. Zoun, and G. Saake, "Gridtables: A one-size-fits-most h2tap data store: Vision and concept," *Datenbank-Spektrum*, 01 2020.

[46] A. Raza *et al.*, "Gpu-accelerated data management under the test of time," in *CIDR*, 2020.

[47] SnappyData. Snappydata 1.2.0 - getting started in 5 minutes or less. Last visited: 12.10.2020. [Online]. Available: https://snappydatainc.github.io/snappydata/quickstart/

[48] MemSQL. Memsql documentation. Last visited: 12.10.2020. [Online]. Available: https://docs.memsql.com/v7.1/introduction/documentation-overview/
[49] Hyrise, “Hyrise github,” last visited: 12.10.2020. [Online]. Available: https://github.com/hyrise/hyrise
[50] HyPer, “Hyper online interface,” last visited: 12.10.2020. [Online]. Available: http://hyper-db.de/interface.html

 ISBN: 978-1-61208-816-7