HyPer-sonic Combined Transaction AND Query Processing

Florian Funke[0], Alfons Kemper[1], Thomas Neumann[2]
Fakultät für Informatik
Technische Universität München
Boltzmannstraße 3, D-85748 Garching
[0]florian.funke@in.tum.de | [1]kemper@in.tum.de | [2]neumann@in.tum.de

ABSTRACT

In this demo we will prove that it is – against common belief – indeed possible to build a main-memory database system that achieves world-record transaction processing throughput and best-of-breed OLAP query response times in **one** system **in parallel** on the **same** database state. The two workloads of online transaction processing (OLTP) and online analytical processing (OLAP) present different challenges for database architectures. Currently, users with high rates of mission-critical transactions have split their data into two separate systems, one database for OLTP and one so-called *data warehouse* for OLAP. While allowing for decent transaction rates, this separation has many disadvantages including data freshness issues due to the delay caused by only periodically initiating the *Extract Transform Load*-data staging and excessive resource consumption due to maintaining two separate information systems. We present an efficient hybrid system, called *HyPer*, that can handle both OLTP and OLAP simultaneously by using hardware-assisted replication mechanisms to maintain consistent snapshots of the transactional data. HyPer is a main-memory database system that guarantees the full ACID properties for OLTP transactions and executes OLAP query sessions (multiple queries) on arbitrarily current and consistent snapshots. The utilization of the processor-inherent support for virtual memory management (address translation, caching, copy-on-write) yields both at the same time: unprecedentedly high transaction rates as high as 100,000+ transactions per second and very fast OLAP query response times on a single system executing both workloads in parallel. The performance analysis is based on a combined TPC-C and TPC-H benchmark.

1. INTRODUCTION

There have been strong arguments by industry leaders, e.g., Hasso Plattner of SAP [6], that our current support for *real-time* Business Intelligence is inappropriate. The currently exercised separation of transaction processing on the OLTP database and BI query processing on the *data warehouse* that is only periodically refreshed violates this goal. We propose to enhance the transactional database with highly effective query processing capabilities. Real-time/operational business intelligence demands to execute OLAP queries on the current, up-to-date state of the transactional OLTP data. Therefore, mixed workloads of OLTP transaction processing and OLAP query processing on the same data (or the same replicated data state) have to be supported. This is somewhat counter to the recent trend of building dedicated systems for different applications. In this demo we will prove that it is – against common belief – indeed possible to build a main-memory database system that achieves world-record transaction processing throughput and best-of-breed OLAP query response times in **one** system **in parallel** on the **same** database state. This unprecedentedly high combined performance is achieved by exploiting the hardware-assisted virtual memory management facilities of modern operating systems.

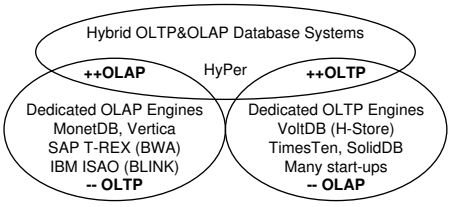

Figure 1: Best of Both Worlds: OLAP and OLTP.

The quest for a hybrid mixed workload system that reconciles the best of both worlds, high-throughput OLTP engines and high-performance OLAP query processors, is depicted in Figure 1. In the OLAP "world" there are highly efficient OLAP query processors based on column store technologies, as pioneered by MonetDB [1]. In the OLTP "world" main-memory database systems such as the recently developed VoltDB [3] (or the time-proven TimesTen) excel in transaction throughput. The goal we set out to pursue was the design of a database system that achieves the same excellent OLTP **and** OLAP performance in one system in parallel on the same data to enable the "*information at your fingertips*" performance requirements of an operational store.

Different architectures were proposed for achieving the real-time BI goal: versioning of the data and thereby separating the query from the transactions workload, continuous DW refreshing, heterogeneous workload management, update staging by periodically merging the update delta into the queryable main database, batch processing of updates and queries, and our newly developed virtual memory snapshot mechanism based on hardware-supported shadowed pages. The latter approach constitutes a main-memory database system – an architecture that receives renewed and increasing interest due to recent hardware developments. Currently, hardware vendors offer cost-effective servers with a TB of RAM for only ca. $50,000. This makes it possible to maintain the transactional data of even the largest applications in main memory of one (or a few) server(s). The RAMcloud development at Stanford is based on a similar observation by estimating, for example, Ama-

Proceedings of the VLDB Endowment, Vol. 4, No. 12

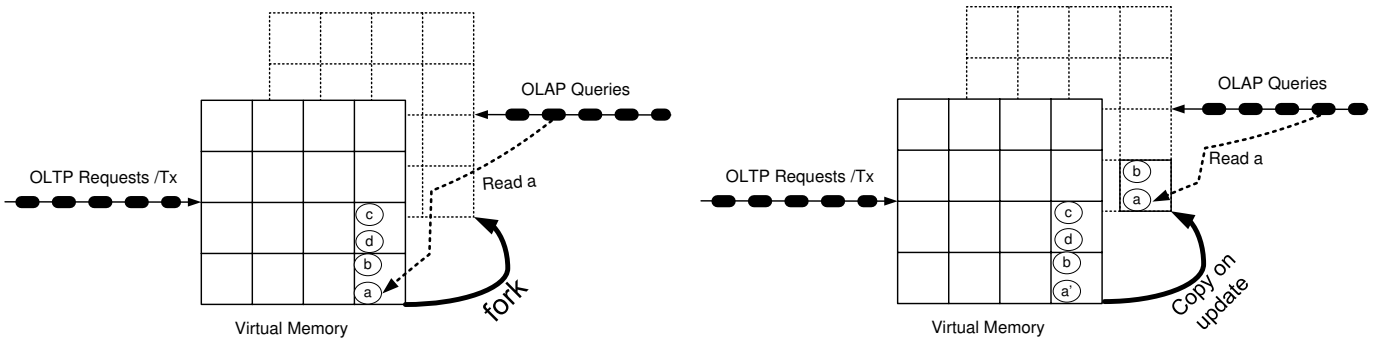

Figure 2: Forking a new Snapshot (left) and copy-on-write/update (right).

zon's transactional data as follows: At 15 billion revenue per year and an average item price of $15 the yearly data volume of 1 billion order-lines can be stored in about 64 GB (assuming that an order-line is as compact as it is in the TPC-C benchmark). We do not propose to abandon data warehouses as there is a need to collect, clean, and transform vast amounts of transactional and other, non-transactional data for in depth data analysis, such as data mining. Our goal is to enhance in-memory transactional database systems with highly effective BI query processing capabilities. Hardware progress favors in-memory technologies, as is also indicated by the many start-up companies developing OLTP main-memory databases, such as VoltDB, Clustrix, Akiban, DBshards, NimbusDB, ScaleDB, Lightwolf, and ElectronDB.

To demonstrate the performance of a hybrid database workload consisting of OLTP and BI/OLAP processing, we developed a new OLTP&OLAP benchmark that combines the transaction processing functionality of the TPC-C benchmark with the query suite of the TPC-H benchmark in one mixed workload. Based on this benchmark we substantiate the claim that it is indeed possible to architect a hybrid system that can achieve the transactional throughput rates of dedicated in-memory OLTP systems and, in parallel, execute a BI workload on the same data at the same performance as dedicated OLAP systems, such as in-memory column stores.

2. TECHNICAL REALIZATION

We have developed the novel hybrid OLTP&OLAP database system HyPer that is based on snapshotting transactional data via the virtual memory management of the operating system [4]. In this architecture the OLTP process "owns" the database and periodically (e.g., in the order of seconds or minutes) forks an OLAP process. This OLAP process constitutes a fresh transaction consistent snapshot of the database. Thereby, we exploit operating systems functionality to create virtual memory snapshots for new, cloned processes. In Unix, for example, this is done by creating a child process of the OLTP process via the `fork` system call. One possibility to guarantee transactional consistency is to `fork` only after quiescing the transaction processing. Actually, this constraint can be relaxed by utilizing the undo log (that is anyway maintained in-memory for all running transactions) to convert an action consistent snapshot (created in the middle of transactions) into a transaction consistent one.

The forked child process obtains an exact copy of the parent processes address space, as exemplified in Figure 2 by the overlayed page frame panel. This virtual memory snapshot that is created by the `fork`-operation will be used for executing a session of OLAP queries – as indicated on the right hand side of Figure 2. These queries can be executed in parallel threads or serially, depending on the system resources or client requirements.

The snapshot stays in precisely the state that existed at the time the `fork` took place. Fortunately, state-of-the-art operating systems do not physically copy the memory segments right away. Rather, they employ a lazy *copy-on-update* strategy – as sketched out in Figure 2. Initially, parent process (OLTP) and child process (OLAP) share the same physical memory segments by translating either virtual addresses (e.g., to object a) to the same physical main memory location. The sharing of the memory segments is highlighted in the graphics by the dotted frames. A dotted frame represents a virtual memory page that was not (yet) replicated. Only when an object, like data item a, is updated, the OS- and hardware-supported copy-on-update mechanism initiate the replication of the virtual memory page on which a resides. Thereafter, the new state denoted a' is accessible by the OLTP-process that executes the transactions and the old state denoted a is accessible by the OLAP query session.

In essence, the virtual memory snapshot mechanism constitutes a OS/hardware supported shadow paging mechanism as proposed decades ago for disk-based database systems. However, the original proposal incurred severe costs as it had to be software-controlled and it destroyed the clustering on disk. Neither of these drawbacks occurs in the virtual memory snapshotting as clustering across RAM pages is not an issue. Furthermore, the sharing of pages and the necessary copy-on-update/write is managed by the operating system with effective hardware support of the MMU (memory management unit) via the page table that translates VM addresses to physical pages and traps necessary replication (copy-on-write) actions. Therefore, the page replication is extremely efficiently done in 2 μs as we measured in a micro-benchmark.

HyPer adheres to the ACID paradigm for the transactional processing. The *atomicity* is guaranteed by maintaining the above mentioned undo-log in main-memory. The *durability* is achieved by efficiently writing logical redo-log records via a high-bandwidth network to a storage server and relying on group commit. The VM snapshot mechanism enables us to periodically write transaction consistent snapshots (mimicked as OLAP session) to a storage server (cf. Figure 3. As far as *isolation* is concerned we follow the approach pioneered in H-Store/VoltDB [3] of lockless synchronization. Transactions, which constitute stored procedures written in a SQL-like scripting language, are executed serially on their corresponding partition. If more than one partition is needed we resort to a serial (i.e., exclusive) operation mode.

HyPer's configuration as a single primary node system – that we propose to demonstrate with two interconnected laptop servers –with a secondary server for OLAP load balancing and stand-by is shown in Figure 3. The secondary server is "fed" by the logical log

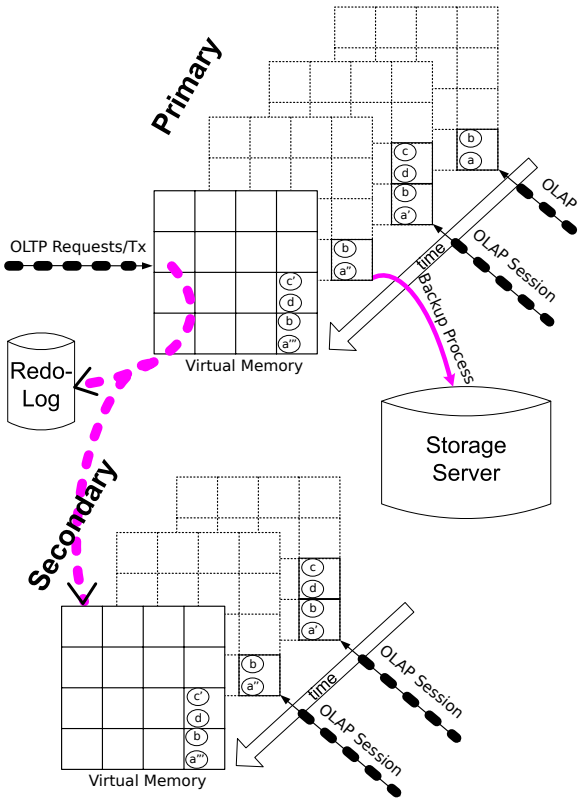

Figure 3: Secondary Server: Stand-By for OLTP and Active for OLAP.

records that are transmitted to the storage server for durability and to the secondary for availability and OLAP load balancing.

The primary as well as the secondary server are capable to create multiple snapshots that overlap in time. This can simply be achieved by periodically (or on demand) `fork`-ing a new snapshot and thus starting a new OLAP query session process. This is exemplified on the top (primary server) of Figure 3. Here we sketch the one and only OLTP process'es current database state (the front panel) and three active query session processes' snapshots – the oldest being the one in the background. Such a snapshot can also be used to back-up a Tx-consistent database archive. The successive state changes are highlighted by the four different states of data item a (the oldest state), a', a'', and a''' (the youngest transaction consistent state). Obviously, most data items do not change in between different snapshots as we expect to create snapshots for most up-to-date querying at intervals of a few seconds – rather than minutes or hours as is the case in current separated data warehouse solutions with ETL data staging. The number of active snapshots is, in principle, not limited, as each "lives" in its own process. By adjusting the priority we can make sure that the mission critical OLTP process is always allocated a core – even if the OLAP processes are numerous and/or utilize multi-threading and thus exceed the number of cores. A snapshot is deleted after the last query of a session is finished.

3. DEMO

Our demo of the HyPer system is based on a new benchmark we call CH-Benchmark [2] to denote that it is a "merge" of the two standard TPC benchmarks (`www.tpc.org`): The TPC-C benchmark was designed to evaluate OLTP database system performance and the TPC-H benchmark for analyzing OLAP query performance. Both benchmarks "simulate" a sales order processing (order entry, payment, delivery) system of a merchandising company. The

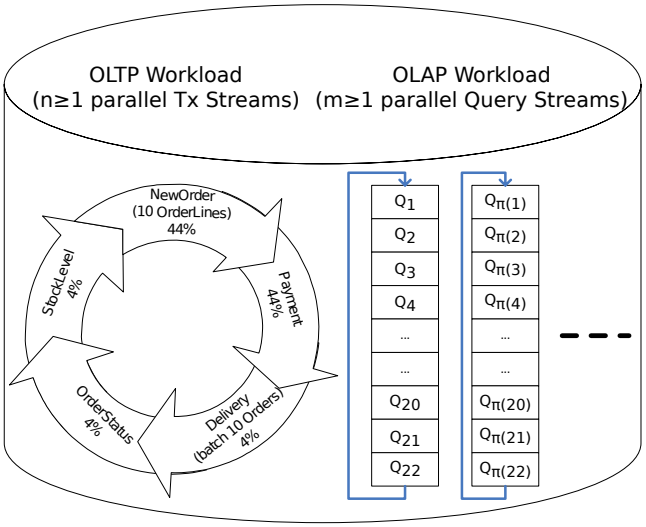

Figure 4: Mixed OLTP&OLAP Workload Benchmark.

benchmark constitutes the core functionality of such a commercial merchandiser like Amazon.

The database schema of the CH-Benchmark is based based on augmenting the TPC-C schema. The original TPC-C schema, that we kept entirely unchanged, consists of the 9 relations: Warehouse (W), District (10∗W), Customer (W∗30k), Order (W∗30k), Order-Line (W∗300k), Stock (W∗100k), Item (100k), and History (W∗30k). In addition, we included three relations from the TPC-H benchmark in order to be able to formulate all 22 queries of this benchmark in a meaningful way: There are 10,000 *Suppliers* that are referenced via a foreign key of the *Stock* relation. Thus, there is a fixed, randomly selected Supplier per Item/Warehouse combination. The relations *Nation* and *Region* model the geographic location of Suppliers and Customers.

The TPC-C OLTP transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses – cf. left-hand side of Figure 4. All these transaction, including the read-only transactions *Order-Status* and *Stock-Level* have to be executed in serializable semantics – in HyPer's case via the OLTP workload queue. For the comprehensive OLTP&OLAP Benchmark we adapted the 22 queries of the TPC-H benchmark for the CH schema. In the reformulation we made sure that the queries retained their semantics (from a business point of view) and their syntactical structure. The OLAP queries do not benefit from indexes or database partitioning as they all require scanning and aggregating large portions of the database.

In the demo (Figure 5) we will demonstrate that HyPer achieves unprecedented performance on this mixed workload benchmark. The OLTP transaction throughput of 200,000 Tx per second is better than for any other database system even though, at the same time, the OLAP queries are processed in parallel. Also, the OLAP query response times of, e.g., subseconds on a database with 3.6 mio Order-Lines, are as good as for any dedicated "query-only" OLAP engine – even while the OLTP processing takes place at "full speed" at the same time on the HyPer system. These performance results, as measured on a live system installed on a commodity (10,000 Euro) server, are shown in the performance monitor of Figure 5 – which will be visualized as a real-time "dash board" during the demo.

We will continuously monitor the performance aggregated over time windows. Thereby, we can show the effects of forking new snapshots on the memory utilization as sketched on the top left-hand side of the screen shot: The continuously increasing memory con-

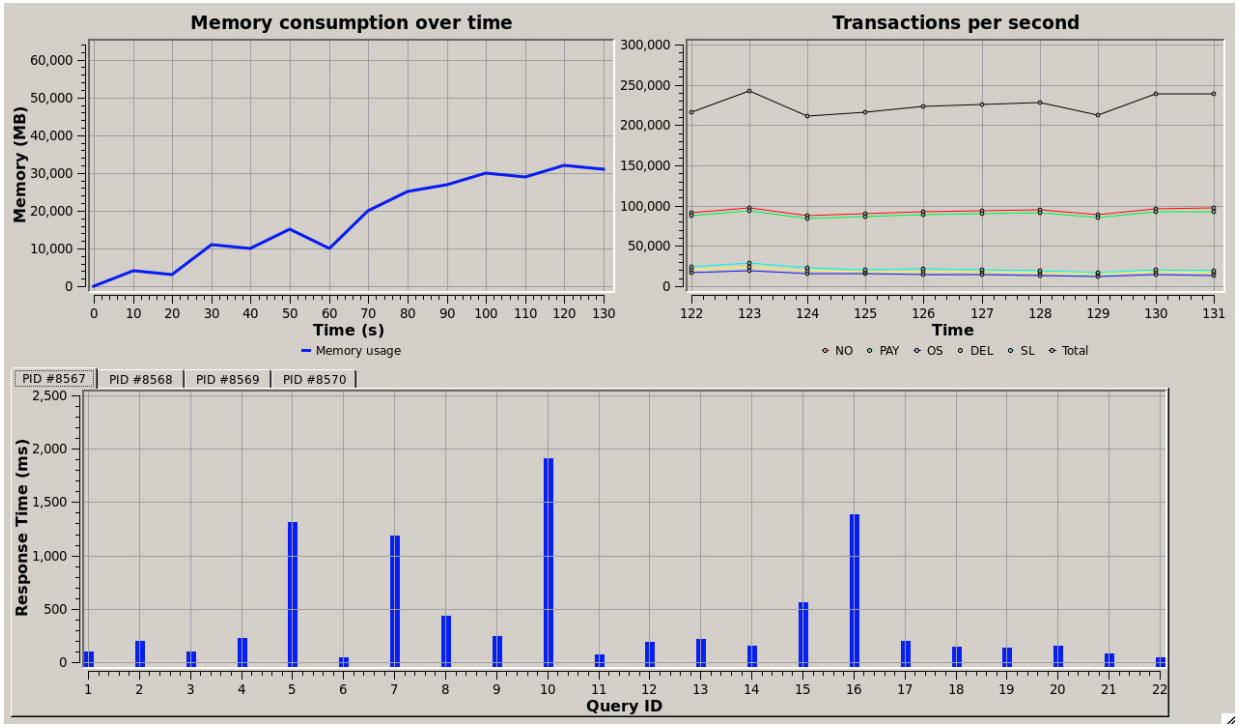

Figure 5: The Performance Monitor in Action: 4 Query Sessions in parallel to Transaction Processing.

sumption is due to the heavy insert rates of the TPC-C benchmark. The downward step in the curves is caused by terminating a snapshot (i.e., terminating a forked OLAP process and freeing its replicated memory pages). The upward steps are caused by creating a new snapshot that accumulated replicated pages due to the copy-on-write mechanism of the virtual memory management.

During the demo, the transaction and the query load can be dynamically varied. Also, the creation (forking) of a new snapshot can be manually triggered. Furthermore, it is possible to interactively add new queries to the OLAP workload. The real-time performance monitor allows to visualize the effects of these variations.

To demonstrate HyPer's SQL-based transaction scripting language, we can also interactively code the (simple) Voter benchmark that was promoted by VoltDB as a Web-scale benchmark to mimic the American Idol phone based voting.

4. SUMMARY

In this demo we want to convince the community that it is – against common belief – possible to architect and build a database system that accommodates both, transaction processing and OLAP/BI query processing in parallel at phenomenal performance. We will demonstrate that the combined OLTP&OLAP-performance is indeed superior to the performance of the best-performing dedicated database systems. HyPer's OLTP throughput is better than VoltDB's published TPC-C performance and HyPer's OLAP query response times are superior to MonetDB's query response times. It should be emphasized that HyPer can match (or beat) these two best-of-breed transaction (VoltDB) and query (MonetDB) processing engines **at the same time** by performing both workloads in parallel on the same database state. HyPer's performance is due to the following design criteria:

- HyPer relies on in-memory data management without the ballast of traditional database systems caused by DBMS-controlled page structures and buffer management. The SQL table definitions are transformed into simple vector-based virtual memory representations – which constitutes a column-oriented physical storage scheme.
- The OLAP processing is separated from the mission-critical OLTP transaction processing by **fork**-ing virtual memory snapshots. Thus, no concurrency control mechanisms are needed – other than the hardware-assisted VM management – to separate the two workload classes.
- Transactions and queries are specified in SQL and are efficiently compiled into LLVM assembly code [5].
- As in VoltDB, the parallel transactions are separated via lock-free admission control that allows only non-conflicting transactions at the same time.
- HyPer relies on logical logging where, in essence, the invocation parameters of the stored (transaction) procedures are logged via a high-speed network.

5. REFERENCES

[1] P. A. Boncz, S. Manegold, and M. L. Kersten, "Database architecture evolution: Mammals flourished long before dinosaurs became extinct," *PVLDB*, vol. 2, no. 2, pp. 1648–1653, 2009.

[2] R. Cole, F. Funke, L. Giakoumakis, A. Kemper, S. Krompaß, H. Kuno, R. Nambiar, T. Neumann, M. Poess, K.-U. Sattler, M. Seibold, E. Simon, and F. Waas, "The mixed workload CH-benCHmark," in *DBTest*, 2011, p. 8.

[3] S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker, "OLTP through the looking glass, and what we found there," in *SIGMOD*, 2008, pp. 981–992.

[4] A. Kemper and T. Neumann, "HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots," in *ICDE*, 2011, pp. 195–206.

[5] T. Neumann, "Efficiently compiling efficient query plans," in *VLDB*, 2011, pp. 539–550.

[6] H. Plattner, "A common database approach for OLTP and OLAP using an in-memory column database," in *SIGMOD*, 2009, pp. 1–2.