ScyPer: A Hybrid OLTP&OLAP Distributed Main Memory Database System for Scalable Real-Time Analytics

Tobias Mühlbauer[*], Wolf Rödiger, Angelika Reiser, Alfons Kemper, Thomas Neumann

Technische Universität München · Boltzmannstraße 3 · D-85748 Garching
{muehlbau, roediger, reiser, kemper, neumann}@in.tum.de

Abstract: ScyPer is an abbreviation for *Sc*aled-out H*yPer*, a version of the HyPer main memory hybrid OLTP&OLAP database system that horizontally scales out on shared-nothing commodity hardware. Our demo shows that ScyPer a) achieves a near-linear scale-out of OLAP query throughput with the number of active nodes, b) sustains a constant OLTP throughput, c) is resilient to node failures, and d) offers real-time analytical capabilities through market-leading query response times and periodically forked TX-consistent virtual memory snapshots with sub-second lifetime durations.

1 Motivation

Database systems face two distinct workloads: online transactional processing (OLTP) and online analytical processing (OLAP). These two workloads are nowadays mostly processed in separate systems, a transactional one and a data warehouse for OLAP, which is periodically updated by a so-called extract-transform-load (ETL) phase. However, ETL interferes with mission-critical OLTP performance and is thus often carried out once every night which inevitably leads to a problem of *data staleness*. Industry leaders such as Hasso Plattner of SAP argue that this data staleness is inappropriate for *real-time business analytics* [PZ11]. New hybrid OLTP&OLAP main memory database systems such as SAP's HANA or HyPer [KN11] achieve best-of-breed transaction processing throughput and OLAP query response times in one system in parallel on the same database state.

Back-of-the-envelope calculations show that these main memory database systems can, on the one hand, handle the OLTP workload and transactional data volumes even for the largest commercial enterprises. E.g., for Amazon we estimate a yearly volume of 54GB for the order lines, the dominating repository in such a sales application [KN11]. Today, servers with 1TB of main memory are available for less than $40,000 and database systems such as HyPer can process more than 100,000 transactions per second on such machines — enough to process a human generated OLTP workload even during busy hours. Furthermore, limited main memory is not a restriction as data can be divided into hot and cold data where the latter can be compacted and swapped to disk [FKN12]. Thus, we conjecture that for the vast majority of use cases OLTP can be managed by a single database server.

Analytical queries on the other hand can be quite complex and computationally expensive. Thus, to maintain performance under high OLAP load, the database needs to be scaled

[*]Tobias Mühlbauer is a recipient of the Google Europe Fellowship in Structured Data Analysis, and this research is supported in part by this Google Fellowship.

out. A scale-out also addresses the need for high availability in the sense that the database can fail-over to an active replica on the fly. In this work we demonstrate ScyPer, a version of the HyPer database system that horizontally scales out on commodity hardware.

2 The HyPer Main Memory Database System

HyPer [KN11] is a hybrid OLTP&OLAP relational main memory database system in which OLAP queries are processed on arbitrarily recent virtual memory (VM) snapshots of the transactional database. This isolation prevents long-running OLAP queries from interfering with mission-critical OLTP processing for which the ACID properties are guaranteed. Unlike traditional database systems, HyPer achieves market-leading performance (compared to state-of-the-art main memory OLTP or OLAP database systems) for both, OLTP and OLAP workloads, operating simultaneously on the same database. HyPer's performance is, among others, due to the following key characteristics: (i) HyPer relies on in-memory data management which eliminates the ballast caused by DBMS-controlled page structures and buffer management. (ii) OLAP processing is separated from mission-critical OLTP processing by forking transaction (TX)-consistent VM snapshots using the POSIX system call `fork()`. The hardware- and operating system-supported "copy on update" mechanism then preserves the consistency of the snapshot by copying pages only if a page is modified. No concurrency control mechanism — other than a short synchronization between the fork and OLTP processing — is needed to separate the two workload classes. (iii) Transactions and queries are specified in SQL or in a scripting language and are efficiently compiled into LLVM code [Neu11]. (iv) As in VoltDB, parallel transactions are separated via lock-free admission control. Thus, non-conflicting transactions can be simultaneously processed in multiple threads (working on different partitions (Ptn)) without the need of concurrency control. (v) HyPer relies on logical logging where, in essence, the invocation parameters of transaction procedures are logged via a high-speed network.

3 ScyPer Architecture

A ScyPer cluster consists of one primary and several secondary nodes where each node runs a ScyPer instance. The architecture of the system is shown in Figure 1.

The primary node is the entry point of the system for transactions as well as analytical queries. The OLTP workload is processed in an OLTP process and the logical redo log is multicasted to all secondary nodes using pragmatic general multicast (PGM). The PGM protocol is scalable and provides the reliable delivery of packets in guaranteed ordering from a single sender to multiple receivers. The redo log, in essence, contains the invocation parameters of transaction procedures together with log sequence numbers. The primary node is not restricted to but usually uses a row-store data layout which is a suitable choice for OLTP processing and keeps indexes that support efficient transaction processing. It can, but does not necessarily have to have TX-consistent snapshots on which it can process OLAP queries or write TX-consistent backups out to a storage node. A coordinator process on the primary node receives incoming OLAP queries and load balances these queries among the secondary nodes.

Secondary nodes receive the multicasted logical redo log from the primary node and rerun each of the transactions. As a large portion of a usual OLTP workload is read-only (i.e.,

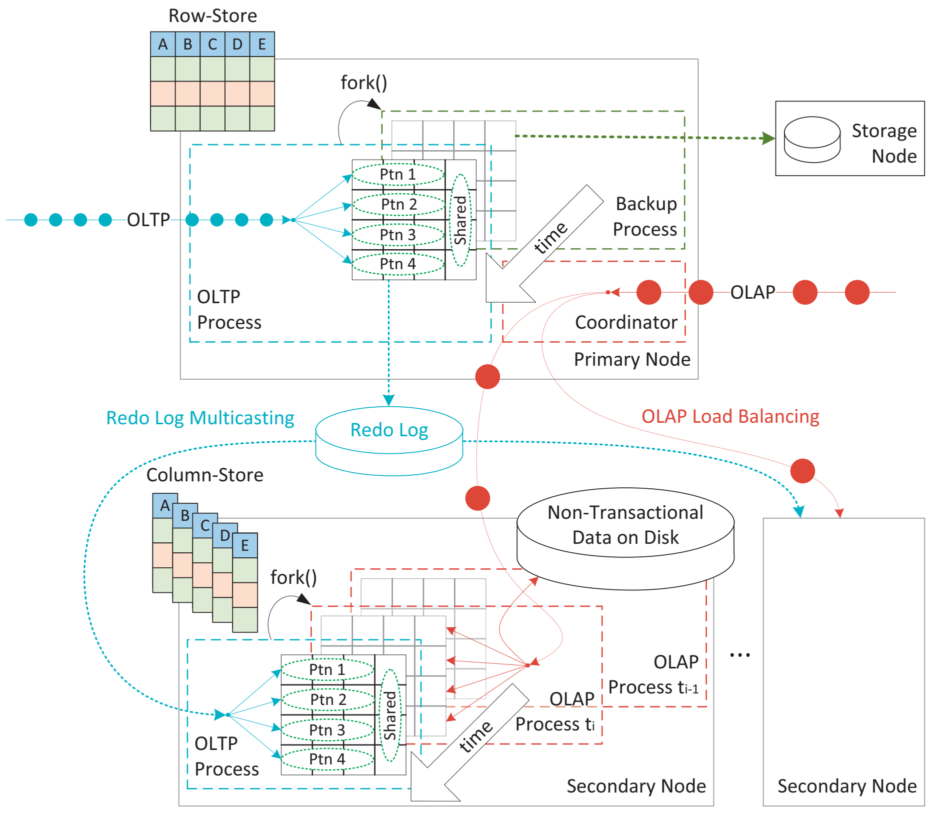
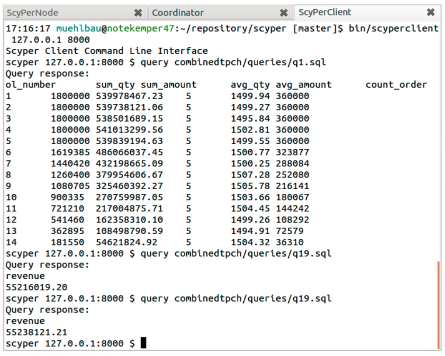


Figure 1: ScyPer architecture

no redo is necessary), secondary nodes usually face less OLTP work than primary nodes. These additional resources are used to process incoming OLAP queries or create backups on forked TX-consistent snapshots. Furthermore, indexes for efficient analytical query processing can be created. Secondary nodes can either store data in a row-, column-, or hybrid row- and column-store data format. Additionally, these nodes can have non-transactional data on disk which can be queried by OLAP queries.

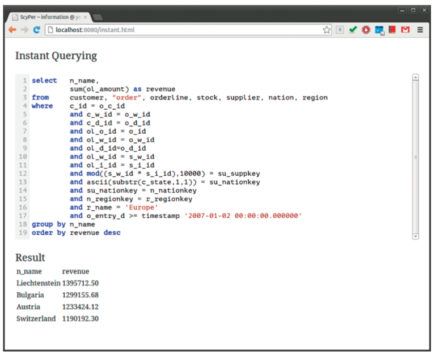
Clients are not restricted to stored transactions and may add new transaction definitions. Internally, such a request is converted to a system-internal transaction which treats the transaction definition as an input of a system-internal *compile* transaction. Similarly, cross-node consistent snapshots can be created where the snapshots have the common logical time of the log sequence number of the system-internal transaction. On such snapshots, queries can be parallelized over several nodes.

All nodes multicast heartbeats such that node failures can be detected. Secondary nodes can fail arbitrarily as we assume that clients re-send OLAP query requests after a timeout. Alternatively OLAP requests can be replicated in the system so processing can be resumed if a secondary node fails. In case of a primary node failure, the secondary nodes elect a new primary using a PAXOS-based protocol. The latest acknowledged log sequence number when failing over is determined by majority consensus.



```
17:16:17 muehlbau@notekemper47:~/repository/scyper [master]$ bin/scyperclient
 127.0.0.1 8000
Scyper Client Command Line Interface
scyper 127.0.0.1:8000 $ query combinedtpch/queries/q1.sql
Query response:
ol_number        sum_qty sum_amount      avg_qty avg_amount      count_order
1       1800000 539978467.23    5       1499.94 360000
2       1800000 539738121.06    5       1499.27 360000
3       1800000 538501689.15    5       1495.84 360000
4       1800000 541013299.56    5       1502.81 360000
5       1800000 539839194.63    5       1499.55 360000
6       1619385 486066037.45    5       1500.77 323877
7       1440420 432198665.09    5       1500.25 288084
8       1260400 379954606.67    5       1507.28 252080
9       1080705 325460392.27    5       1505.78 216141
10      900335  270759987.05    5       1503.66 180067
11      721210  217004875.71    5       1504.45 144242
12      541460  162358310.10    5       1499.26 108292
13      362895  108498790.59    5       1494.91 72579
14      181550  54621824.92     5       1504.32 36310
scyper 127.0.0.1:8000 $ query combinedtpch/queries/q19.sql
Query response:
revenue
55216019.20
scyper 127.0.0.1:8000 $ query combinedtpch/queries/q19.sql
Query response:
revenue
55238121.21
scyper 127.0.0.1:8000 $ █
```

Figure 2: ScyPer CLI Client



Instant Querying

```
select   n_name,
         sum(ol_amount) as revenue
from     customer, "order", orderline, stock, supplier, nation, region
where    c_id = o_c_id
         and c_w_id = o_w_id
         and c_d_id = o_d_id
         and ol_o_id = o_id
         and ol_w_id = o_w_id
         and ol_d_id=o_d_id
         and ol_w_id = s_w_id
         and ol_i_id = s_i_id
         and mod((s_w_id * s_i_id),10000) = su_suppkey
         and ascii(substr(c_state,1,1)) = su_nationkey
         and su_nationkey = n_nationkey
         and n_regionkey = r_regionkey
         and r_name = 'Europe'
         and o_entry_d >= timestamp '2007-01-02 00:00:00.000000'
group by n_name
order by revenue desc
```

Result

n_name	revenue
Liechtenstein	1395712.50
Bulgaria	1299155.68
Austria	1233424.12
Switzerland	1190192.30

Figure 3: ScyPer Web Client

4 Demonstration Setup

To demonstrate the capabilities of ScyPer, we run (i) a benchmark demo and (ii) an interactive demo where we simulate the sales order processing system (order entry, payment, delivery) of a merchandising company. The schema and a mixed OLTP and OLAP workload for this system are based on the CH-benCHmark [C$^+$11], a “merge” of the standard TPC-C (OLTP) and TPC-H (OLAP) benchmarks. This scenario resembles the core business of a commercial merchandiser like Amazon.

The demos run on a ScyPer cluster composed of a varying number of commodity machines (Intel Core i7-3770 CPU, 32 GB dual-channel DDR3-1600 DRAM, Linux 3.5 64bit) and located at TUM. The primary node of the cluster solely processes the OLTP workload; secondary nodes process the logical redo log and periodically fork TX-consistent snapshots with a varying lifetime for OLAP processing. During both demos we simulate node failures to demonstrate ScyPer’s high availability capabilities.

(i) In the *benchmark demo* we run the CH-benCHmark with a varying number of nodes. Performance metrics such as the linearly-scalable OLAP and the constant OLTP throughput as well as query response times are visualized on a dashboard.

(ii) In the *interactive demo* we only run the OLTP workload of the CH-benCHmark while participants can simultaneously enter analytical SQL-92 queries via a command line (see Figure 2) or a web (see Figure 3) interface that presents results “at the keystroke”.

References

[C$^+$11] R. Cole et al. The mixed workload CH-benCHmark. In *DBTest*, pages 8:1–8:6, 2011.

[FKN12] F. Funke, A. Kemper, and T. Neumann. Compacting Transactional Data in Hybrid OLTP&OLAP Databases. *PVLDB*, 5(11):1424–1435, 2012.

[KN11] A. Kemper and T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *ICDE*, pages 195–206, 2011.

[Neu11] T. Neumann. Efficiently compiling efficient query plans for modern hardware. *PVLDB*, 4(9):539–550, 2011.

[PZ11] H. Plattner and A. Zeier. *In-Memory Data Management: An Inflection Point for Enterprise Applications*. Springer, 2011.