HyPer: one DBMS for all
Tobias Mühlbauer, Florian Funke, Viktor Leis, Henrik Mühe, Wolf Rödiger, Alfons Kemper, Thomas Neumann
Technische Universität München
New England Database Summit 2014
http://www.hyper-db.com/
One DBMS for all? OLTP and OLAP

Traditionally, DBMSs are either optimized for OLTP or OLAP

**OLTP**
- high rate of mostly tiny transactions
- high data access locality

**OLAP**
- few, but long-running transactions
- scans large parts of the database
- must see a consistent database state during execution

**Conflict of interest**: traditional solutions like 2PL would block OLTP

However: **Main memory DBMSs enable new options!**
One DBMS for all? Wimpy and brawny

High-performance DBMSs are optimized for brawny servers

**Brawny servers**
- predominantly x86-64 arch
- multiple sockets, many cores

**Wimpy devices**
- predominantly ARM arch
- wide-spread use of SQLite
- energy efficiency very important

Question: How to enable high-performance OLTP/OLAP on different architectures (ARM, x86-64, ...)?
HyPer: one DBMS for all

Simultaneous (ACID) OLTP and (SQL-92+) OLAP:

- **Efficient snapshotting** (ICDE 2011)

Platform-independent high-performance on brawny and wimpy nodes:

- **Data-centric code generation** (VLDB 2011)

Recent research:

- **ARTful indexing**: the adaptive radix tree (ICDE 2013)
- **HTM** for concurrent transaction processing (ICDE 2014)
- **Compaction: Hot/Cold Clustering** of transactional data (VLDB 2012)
- **Instant Loading**: bulk loading at wire speed (VLDB 2014)
- **ScyPer**: replication and scalable OLAP throughput (DanaC 2013)
- **Locality-aware parallel DBMS operators** (ICDE 2014)
Efficient Snapshotting

HyPer isolates long-running transactions (e.g., OLAP) using virtual memory (VM) snapshots.

- OLTP “owns” database
- for OLAP only VM page table is initially copied
- MMU/OS keep track of changes
- snapshots remain consistent (copy on write)
- very little overhead

Extremely fast execution model.

Supports OLTP and OLAP simultaneously.
Main memory is so fast that CPU becomes the bottleneck

- classical iterator model fine for disks, but not so for main memory
- iterator model: many branches, bad code and data locality

HyPer’s data-centric code generation

- touches data as rarely as possible
- prefers tight work loops
  1. load data into CPU registers
  2. perform all pipeline operations
  3. materialize into next pipeline breaker
- efficient platform-independent machine code generation using LLVM
Resent Research
Tentative Execution


- HyPer offers outstanding performance for tiny transactions: > 100,000 TPC-C transactions per second per hardware thread
- Long-running transactions that write to the database cripple performance
- Tentative execution: process long-running transactions on snapshot and “merge” changes into main process
ARTful Indexing


Adaptive Radix Tree (Trie) indexing

- efficient general-purpose read/write/update-able index structure
- faster than highly-tuned, read-only search trees
- optimized for modern hardware
- highly space-efficient by adaptive choice of compact data structures
- performance comparable to HT
- BUT: data is sorted; enables, e.g., range scans and prefix lookups

Adaptive Radix Tree (Trie) indexing

- efficient general-purpose read/write/update-able index structure
- faster than highly-tuned, read-only search trees
- optimized for modern hardware
- highly space-efficient by adaptive choice of compact data structures
- performance comparable to HT
- BUT: data is sorted; enables, e.g., range scans and prefix lookups

Adaptive Radix Tree (Trie) indexing

- efficient general-purpose read/write/update-able index structure
- faster than highly-tuned, read-only search trees
- optimized for modern hardware
- highly space-efficient by adaptive choice of compact data structures
- performance comparable to HT
- BUT: data is sorted; enables, e.g., range scans and prefix lookups

Adaptive Radix Tree (Trie) indexing

- efficient general-purpose read/write/update-able index structure
- faster than highly-tuned, read-only search trees
- optimized for modern hardware
- highly space-efficient by adaptive choice of compact data structures
- performance comparable to HT
- BUT: data is sorted; enables, e.g., range scans and prefix lookups

Adaptive Radix Tree (Trie) indexing

- efficient general-purpose read/write/update-able index structure
- faster than highly-tuned, read-only search trees
- optimized for modern hardware
- highly space-efficient by adaptive choice of compact data structures
- performance comparable to HT
- BUT: data is sorted; enables, e.g., range scans and prefix lookups
Concurrent transaction processing in main memory DBMSs often relies on user-provided data partitioning (1 thread per partition)

**BUT:** what if no partitioning is provided/can be found?

**Hardware transactional memory (HTM)** allows for efficient light-weight optimistic concurrency control without explicit partitions.
Compaction: Hot/Cold Clustering

Funke, F.; Kemper, A.; Neumann, T., “Compacting Transactional Data in Hybrid OLTP&OLAP Databases”, VLDB 2012

Main memory is a scarce resource!

Ideal clustering

Reality

Hot/Cold Clustering

“Temperature” detection:

- Hardware-assisted (MMU)
- Almost no overhead!

Hot and Cold Purge hot items

Immutable

Compressed

Optimized for OLAP

Frozen
Instant Loading


- Bulk loading of structured text data is slow in current DBMSs
- Current loading does not saturate wire speed of data source and data sink
- Instant Loading allows scalable bulk loading at wire speed by efficient task- and data-parallelization
- 10x faster than Vectorwise/MonetDB, orders of magnitude faster than traditional DBMSs (with same conversions/consistency checks)
ScyPer: Replication and Scalable OLAP Throughput


- Secondary nodes needed for high availability—why not use them for OLAP?
- Primary HyPer node processes transactions and multicasts the redo log to secondaries (via Ethernet or InfiniBand)
- **We advocate for physical log multicasting after commit**
  - non-determinism in transaction logic and by unforeseeable faults
  - no need to re-execute expensive transaction logic
Data Shuffling

Locality-aware data shuffling:
• can exploit already small degrees of data locality
• does not degrade when data exhibits no co-location
• optimally assigns partitions
• avoids cross traffic through communication scheduling

Distributed operators like the join operator need data shuffling
HyPer – A Hybrid OLTP&OLAP High Performance DBMS

HyPer is a hybrid online transactional processing (OLTP) and online analytical processing (OLAP) high-performance main memory database system that is optimized for modern hardware. HyPer achieves highest performance—compared to state of the art main memory databases—for both, OLTP (> 100,000 single-threaded TPC-C TX/s on modern commodity hardware) and OLAP (best-of-breed response times), operating simultaneously on the same database.

http://www.hyper-db.com/
HyPer WebInterface

Note: This WebInterface queries a HyPer instance executing queries in a single thread on a low-end server (Intel® Core™ i3-2120 CPU, 16 GB RAM); mind that this interface is thus not intended for benchmarking purposes.

Enter a SQL query against a scale-factor 1 TPC-H or the Uni database and retrieve the result set or show the optimized query plan:

```
0
1 select
2     o_year,
3   sum(case
4     when nation = 'BRAZIL' then volume
5     else 0
6   end) / sum(volume) as nkt_share
7 from
8   {
9     select
10       extract(year from o_orderdate) as o_year,
11       l_extendedprice * (1 - l_discount) as volume,
12       n2.n_name as nation
13     from
14       part,
15       supplier,
16       lineitem,
17       orders,
18       customer,
19       nation n1,
20       nation n2,
21       region
22     where
23       p_partkey = l_partkey
24       and s_suppkey = l_suppkey
25       and l_orderkey = o_orderkey
26       and o_custkey = c_custkey
27       and c_nationkey = n1.n_nationkey
28       and n1.n_regionkey = r_regionkey
29       and r_n_name = 'AMERICA'
30       and s_nationkey = n2.n_nationkey
31       and o_orderdate between date '1995-01-01' and date '1996-12-31'
32       and p_type = 'ECONOMY ASSOCIATED STEEL'
33   } as all_nations
34 group by o_year
35 order by o_year
```

http://www.hyper-db.com/interface.html
TPC-H query 8

Query Result

Compilation time: 48.5447 ms
Execution time: 50.4597 ms
Result set size: 2

<table>
<thead>
<tr>
<th>o_year</th>
<th>mkt_share</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>0.0344</td>
</tr>
<tr>
<td>1996</td>
<td>0.0414</td>
</tr>
</tbody>
</table>

WebInterface on TPC-H scale factor 1, non-parallel query execution

Note: compilation time independent of scale factor

Soon: WebInterface on brawny machine, TPC-H scale factor 100, parallel query execution and instant loading interface for your own files!

http://www.hyper-db.com/interface.html
Inspired by “OLTP through the looking glass” and “The End of an Architectural Era”, the HyPer project has been started.

HyPer is one DBMS that …

• offers highest performance on both, brawny x86-64 platforms as well as wimpy ARM platforms
• performance comparable to/outperforming VoltDB in TPC-C
• performance comparable to/outperforming Vectorwise in TPC-H
• enables OLAP on the most recent OLTP state

2) Neumann, T., “Efficiently compiling efficient query plans for modern hardware”, VLDB, 2011


5) Leis, V.; Kemper, A.; Neumann, T., “Exploiting Hardware Transactional Memory in Main-Memory Databases”, ICDE, 2014


