HyPer: one DBMS for all

Tobias Mühlbauer, Florian Funke, Viktor Leis, Henrik Mühe, Wolf Rödiger, Alfons Kemper, Thomas Neumann

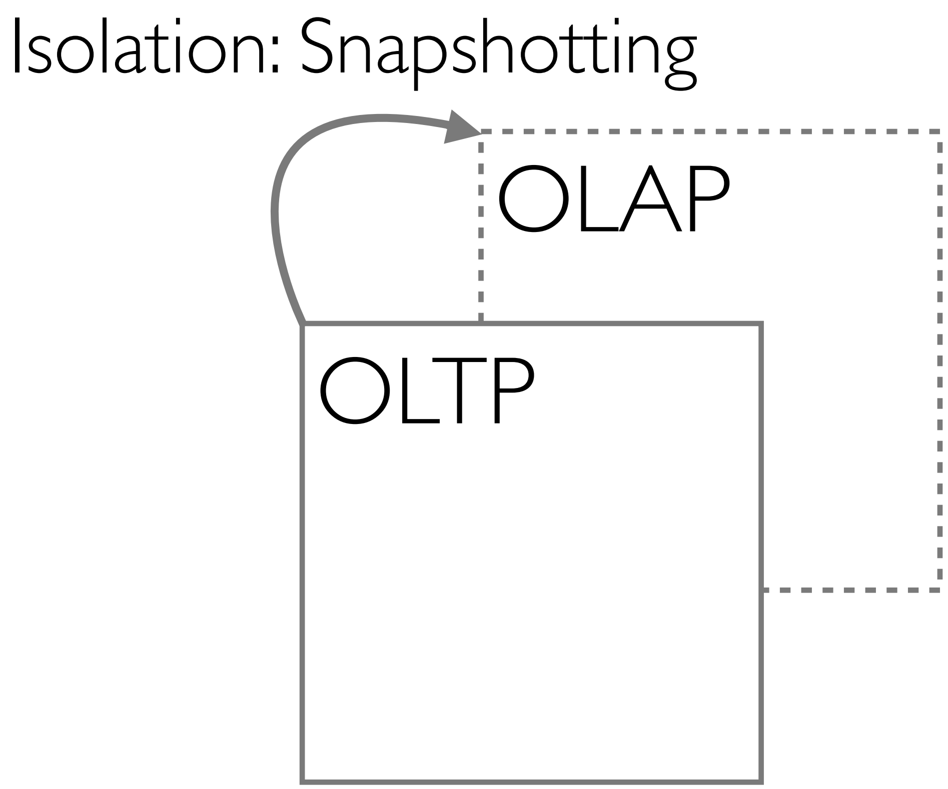
Technische Universität München

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

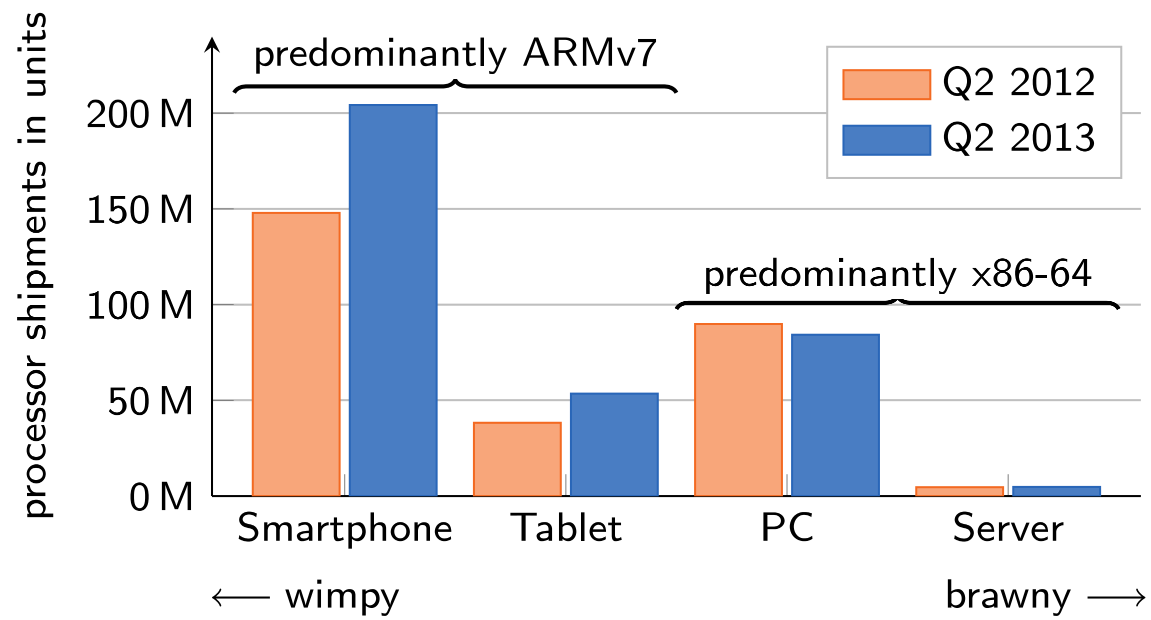
High-performance DBMSs are **optimized for brawny servers**

Brawny servers

- predominantly x86-64 arch
- multiple sockets, many cores

Wimpy devices

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



IHS. Processor Market Set for Strong Growth in 2013, Courtesy of Smartphones and Tablets.

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)

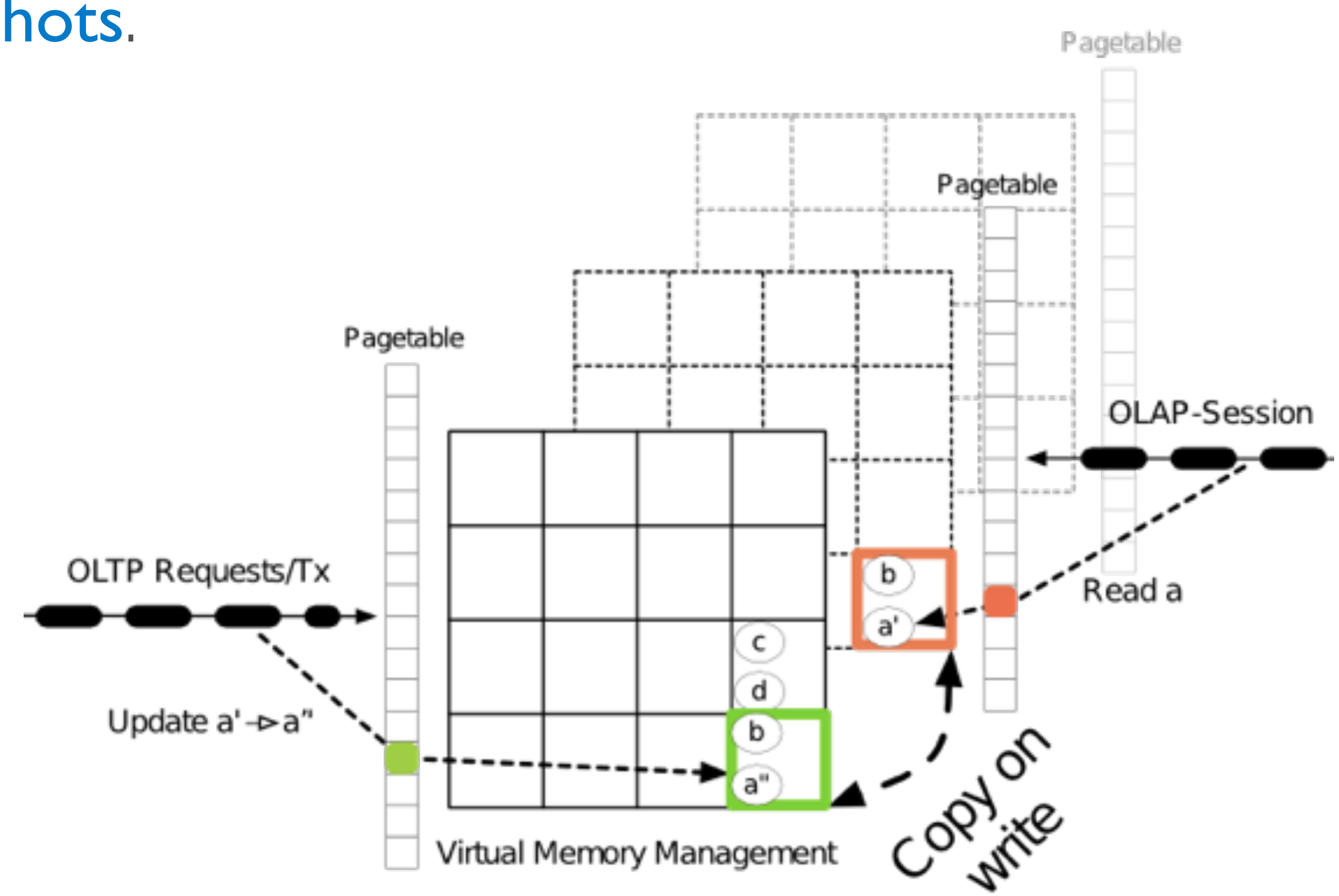
Recent research:

- 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



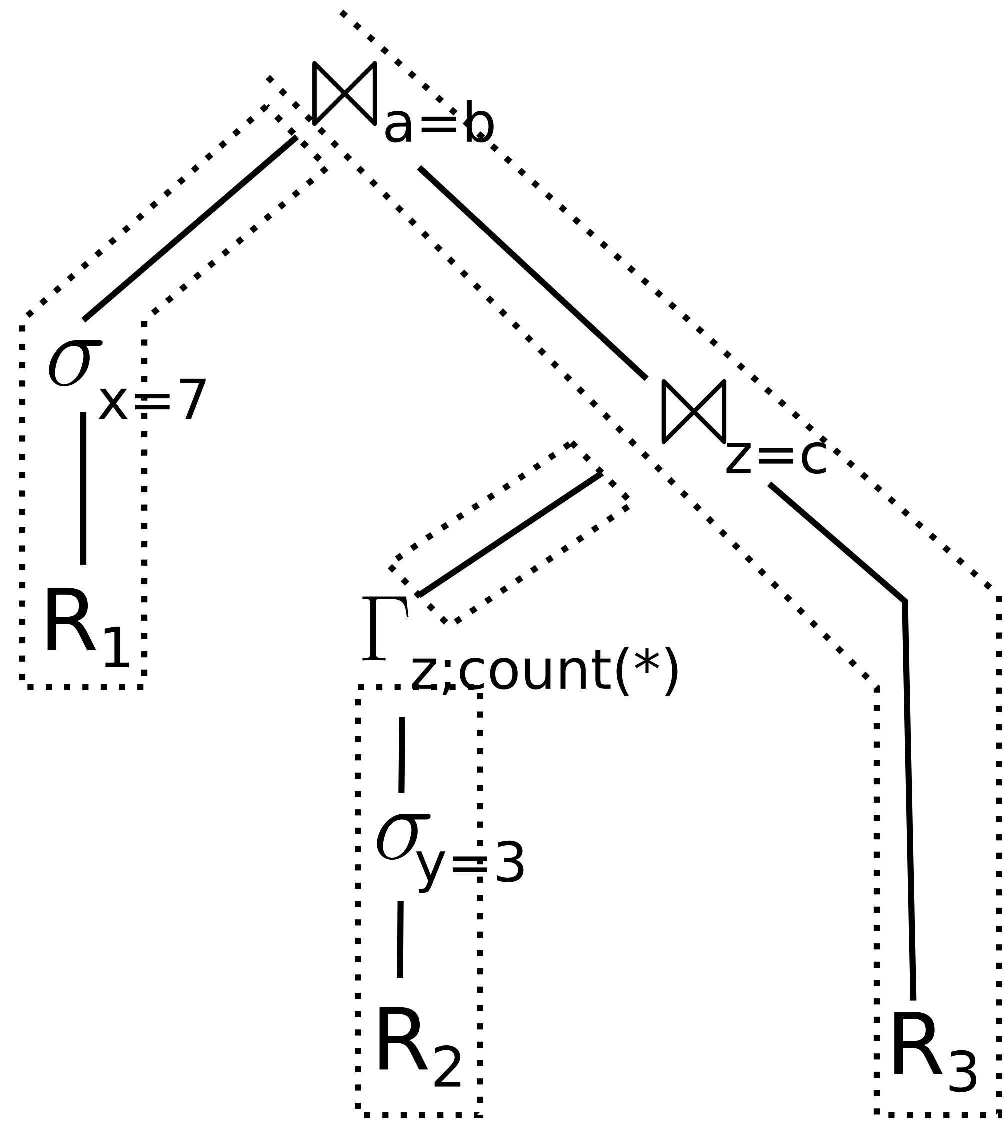
Extremely fast execution model.

Supports OLTP and OLAP simultaneously.

Data-centric Code Generation

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



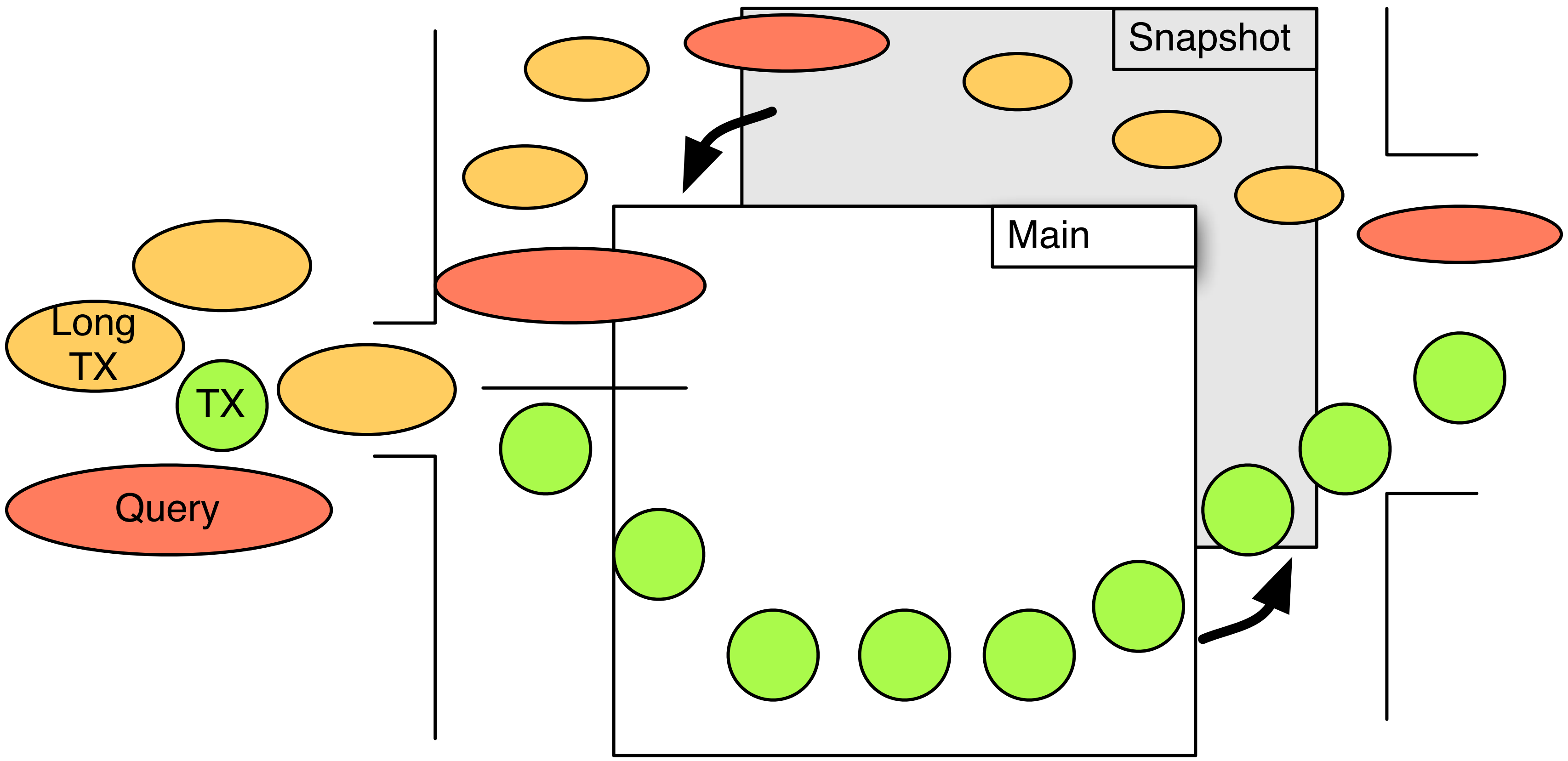
HyPer's **data-centric code generation**

- 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

Mühe, H; Kemper, A.; Neumann, T., "Executing Long-Running Transactions in Synchronization-Free Main Memory Database Systems", CIDR, 2013



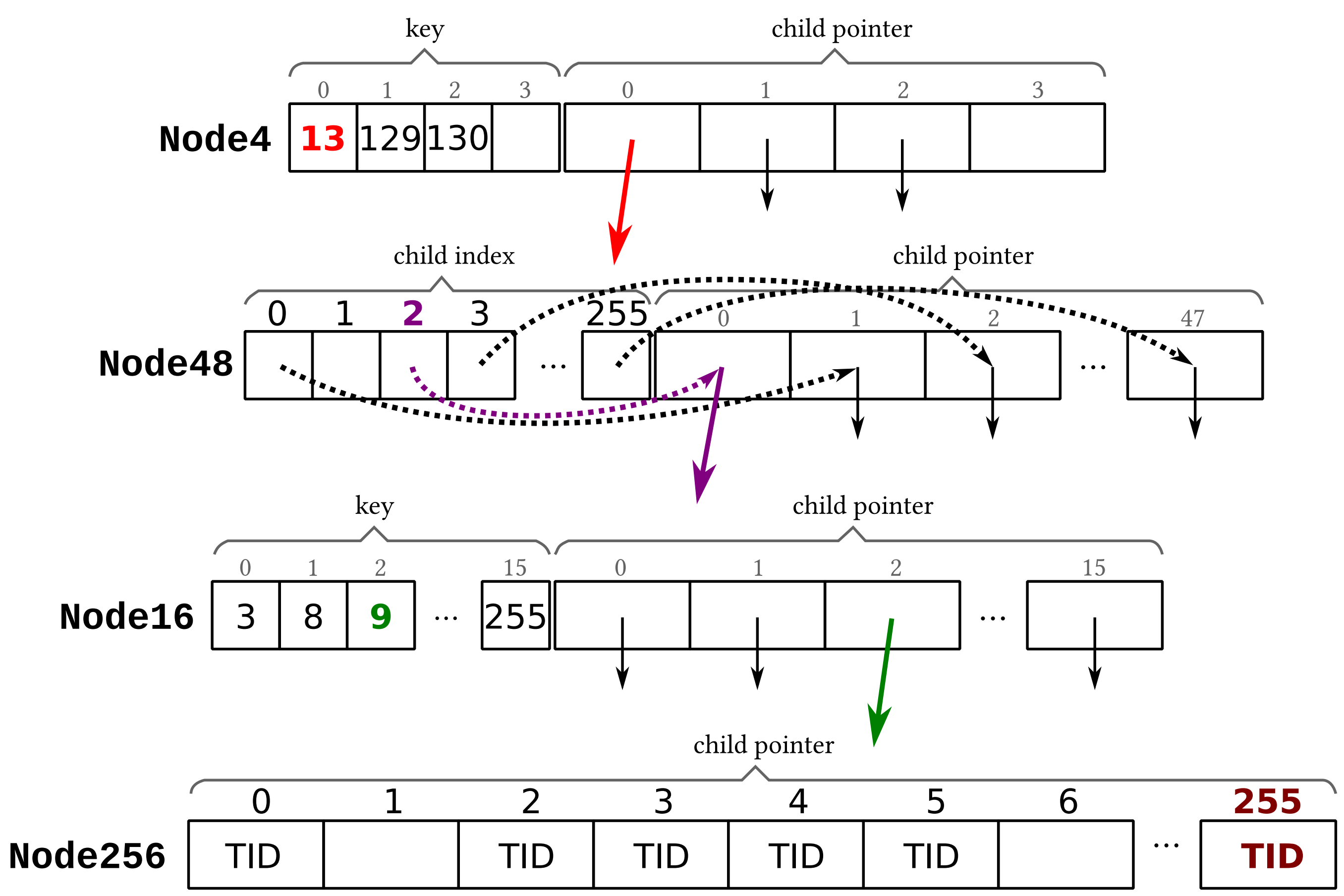
- HyPer offers outstanding performance for tiny transactions: > 100,000 TPC-C transactions per second per hardware thread
- Long-running transactions that write to the database cripple performance
- **Tentative execution**: process long-running transactions on snapshot and "merge" changes into main process

ARTful Indexing

Leis, V.; Kemper, A.; Neumann, T., "The adaptive radix tree: ARTful indexing for main-memory databases", ICDE, 2013

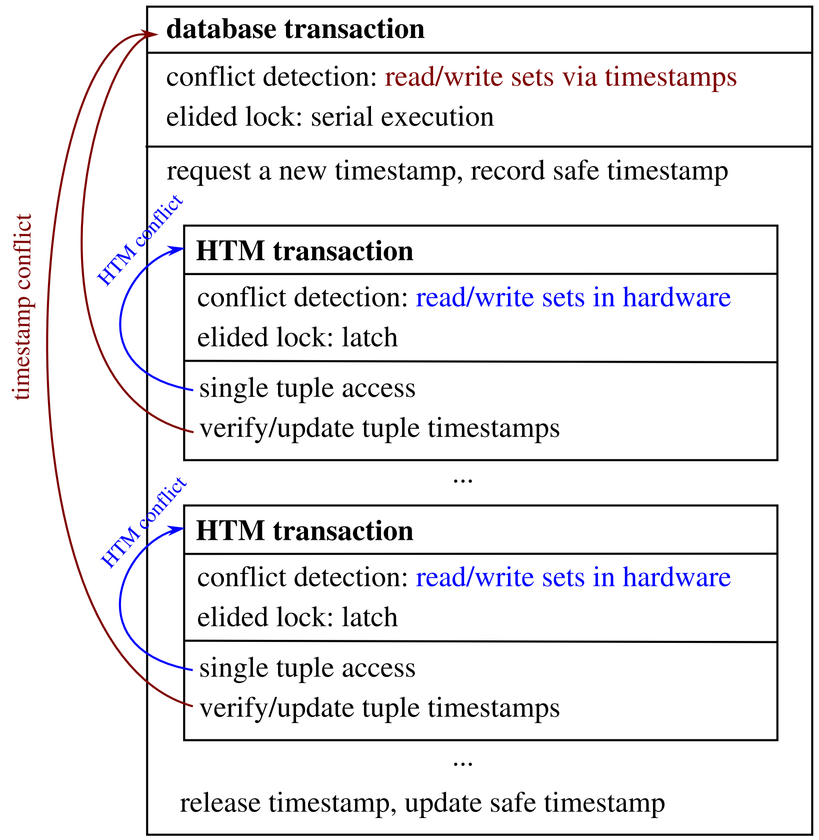
Adaptive Radix Tree (Trie) indexing

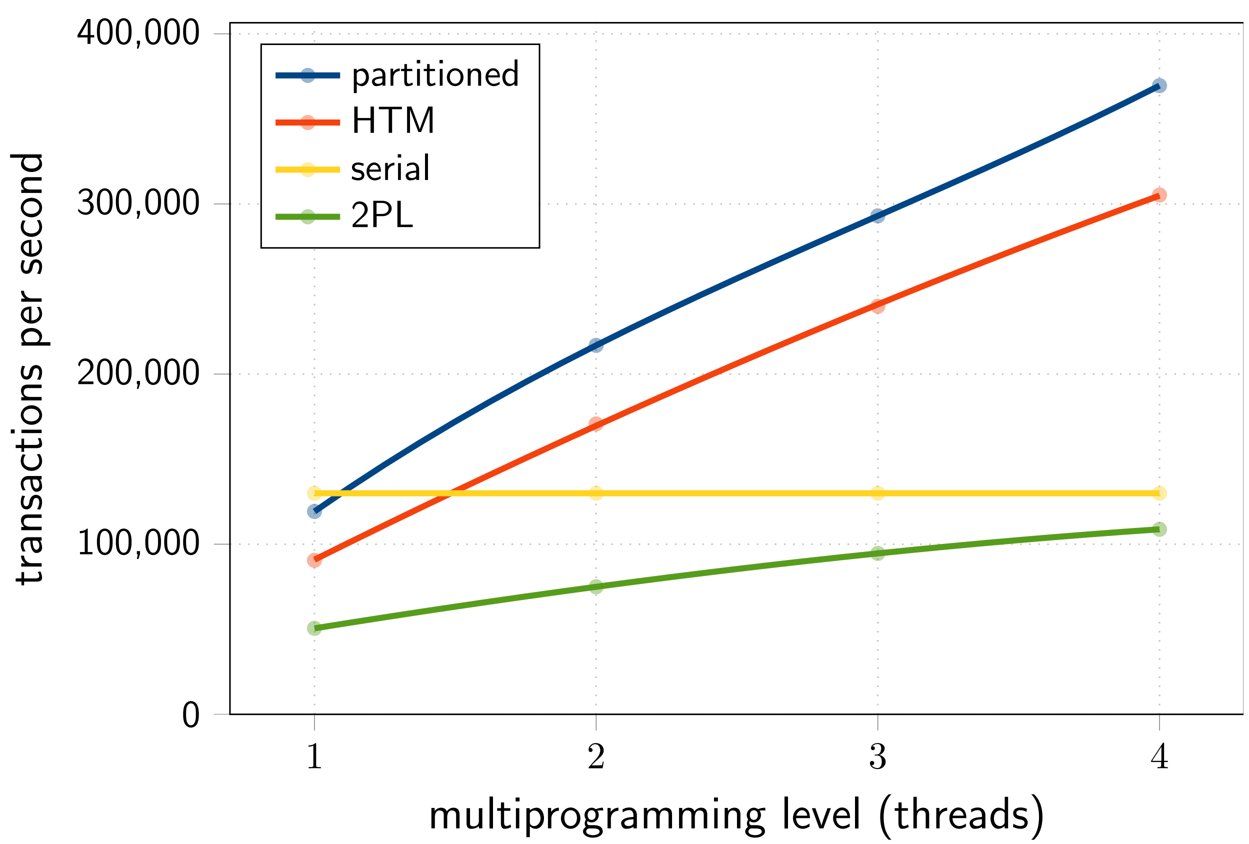
- efficient **general-purpose** read/write/update-able index structure
- faster than highly-tuned, read-only search trees
- optimized for modern hardware
- **highly space-efficient** by adaptive choice of compact data structures
- **performance comparable to HT**
- BUT: data is sorted; enables, e.g.,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



HTM for Concurrency Control

Leis, V.; Kemper, A.; Neumann, T., "Exploiting Hardware Transactional Memory in Main-Memory Databases", ICDE, 2014



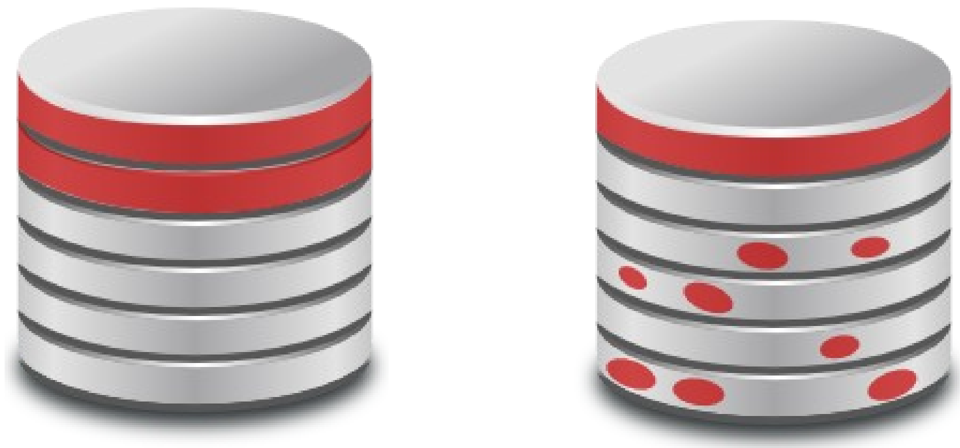


- Concurrent transaction processing in main memory DBMSs often relies on **user-provided data partitioning (1 thread per partition)**
- BUT: **what if no partitioning is provided/can be found?**
- **Hardware transactional memory (HTM)** allows for efficient light-weight optimistic concurrency control 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!

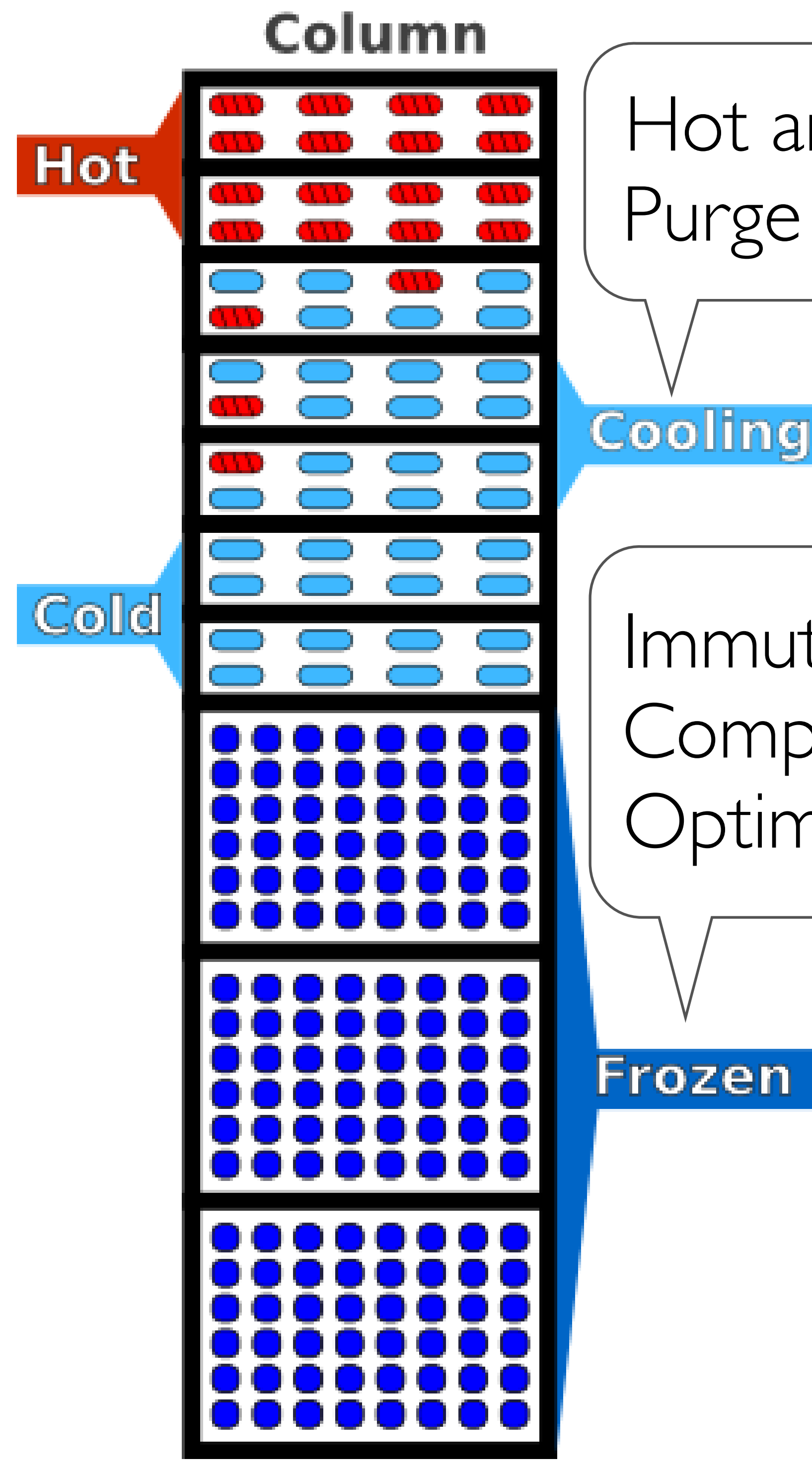


Ideal clustering Reality

Hot/Cold Clustering

"Temperature" detection:

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

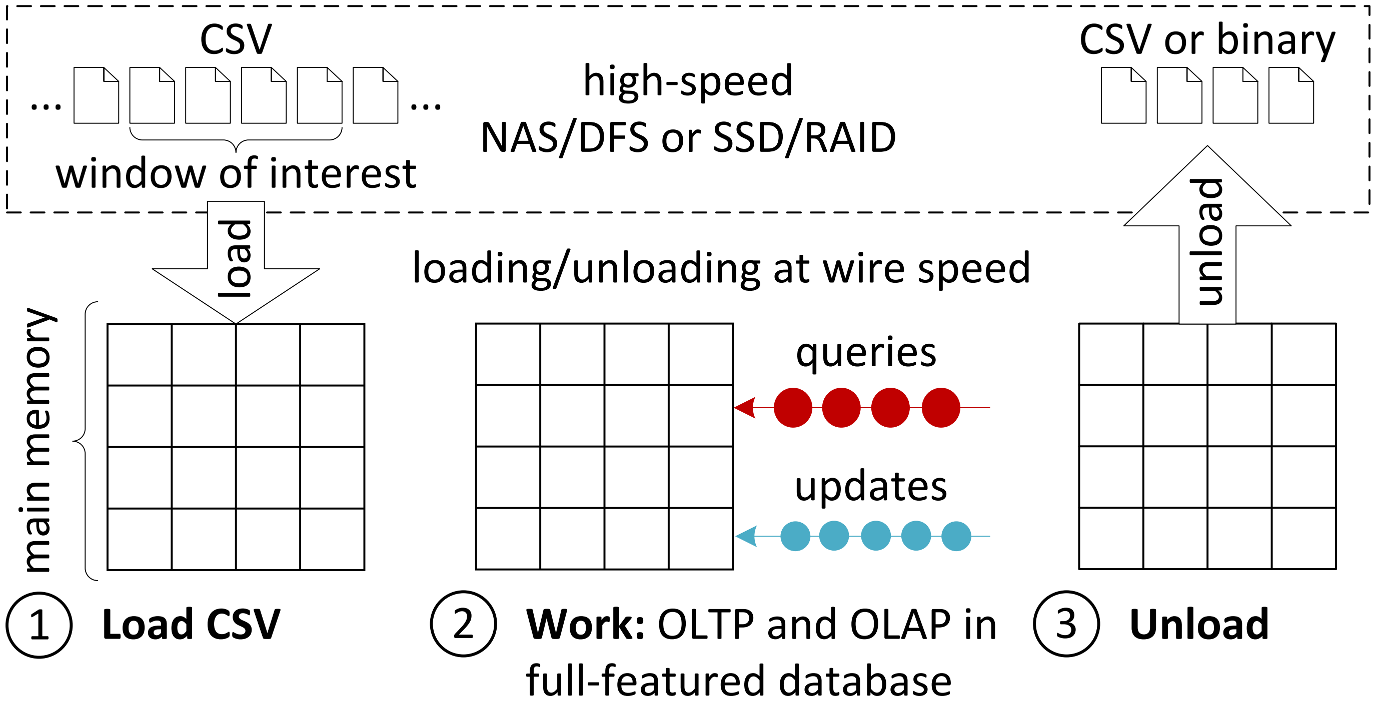


Hot and Cold
Purge hot items

Immutable
Compressed
Optimized for OLAP

Instant Loading

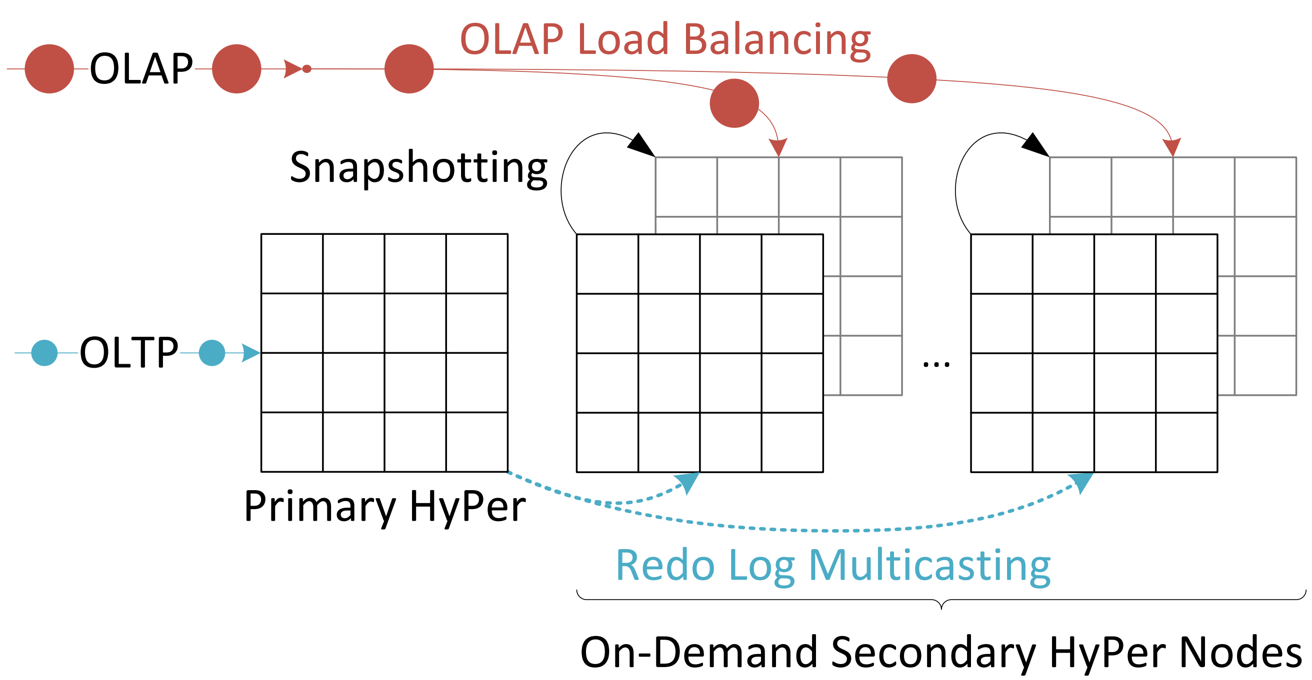
Mühlbauer, T.; Rödiger, W.; Seilbeck, R.; Reiser, A.; Kemper, A.; Neumann, T., "Instant Loading for Main Memory Databases", PVLDB, 2013, presented at VLDB 2014

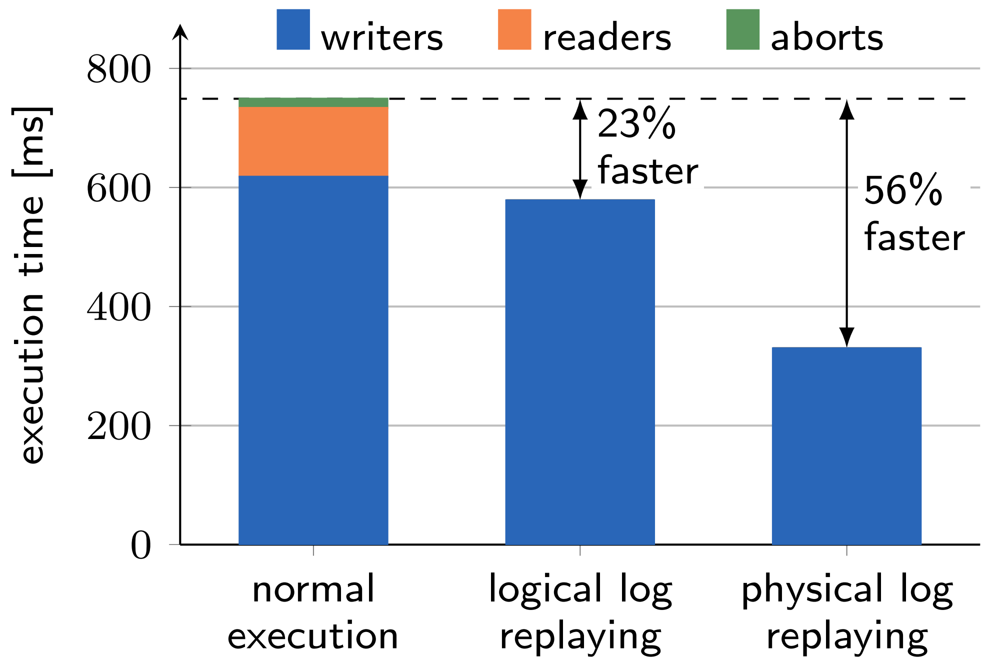


- **Bulk loading of structured text data is slow in current DBMSs**
- Current loading does not saturate wire speed of data source and data sink
- **Instant Loading allows scalable bulk loading at wire speed** by efficient task- and data-parallelization
- 10x faster than Vectorwise/MonetDB, orders of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

ScyPer Replication and Scalable OLAP Throughput

Mühlbauer, T.; Rödiger, W.; Reiser, A.; Kemper, A.; Neumann, T., "ScyPer: Elastic OLAP Throughput on Transactional Data", DanaC, 2013

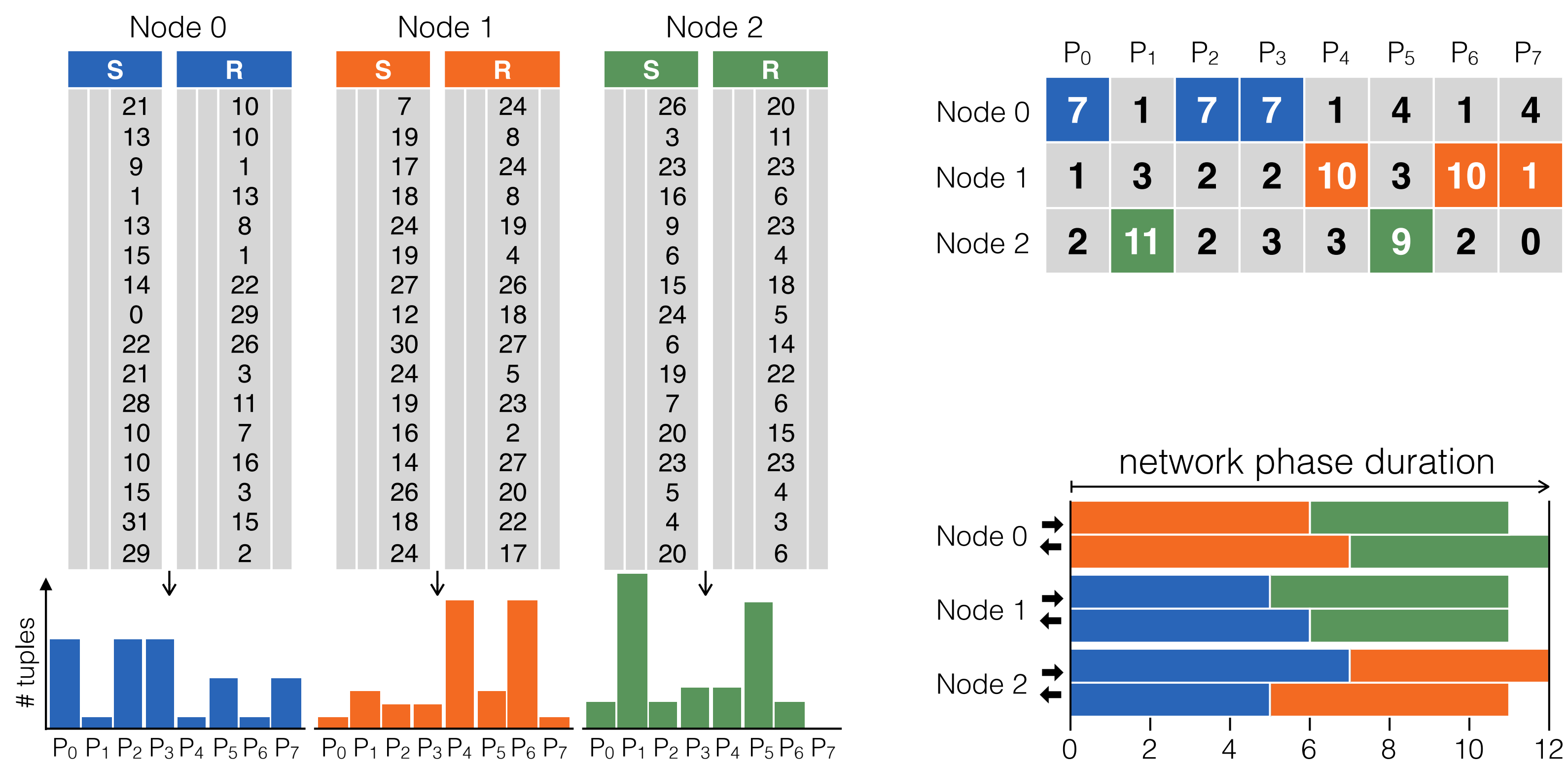




- Secondary nodes needed for **high availability**—why not use them for OLAP?
- Primary HyPer node processes transactions and **multicasts the redo log** to secondaries (via Ethernet or Infiniband)
- **We advocate for physical log multicasting after commit**
  - non-determinism in transaction logic and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

Data Shuffling Distributed Query Processing with HyPer

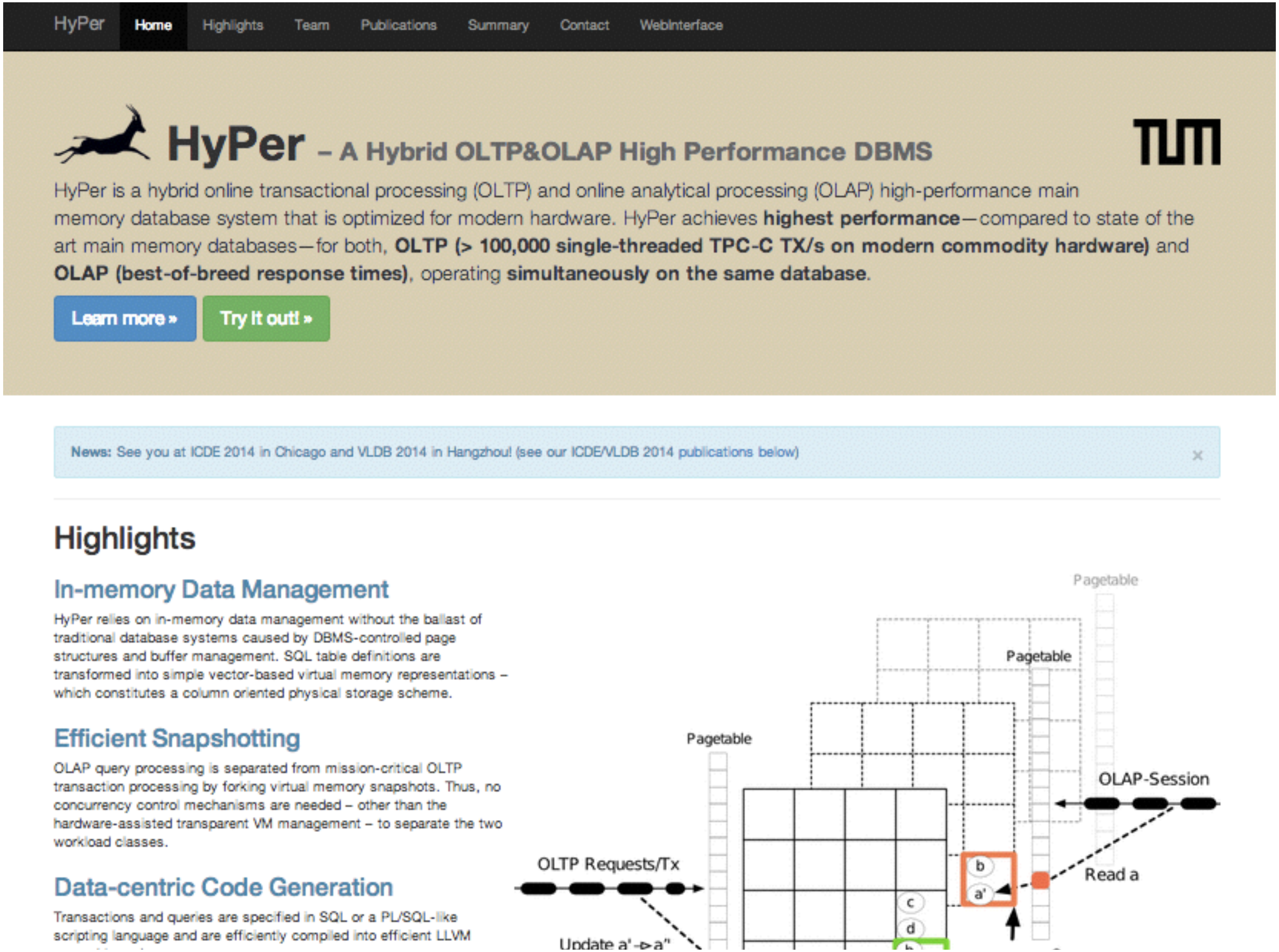
Rödiger, W.; Mühlbauer, T.; Unterbrunner, P.; Reiser, A.; Kemper, A.; Neumann, T., "Locality-Sensitive Operators for Parallel Main-Memory Database Clusters", ICDE, 2014



Distributed operators like the join operator need 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**



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

HyPer WebInterface

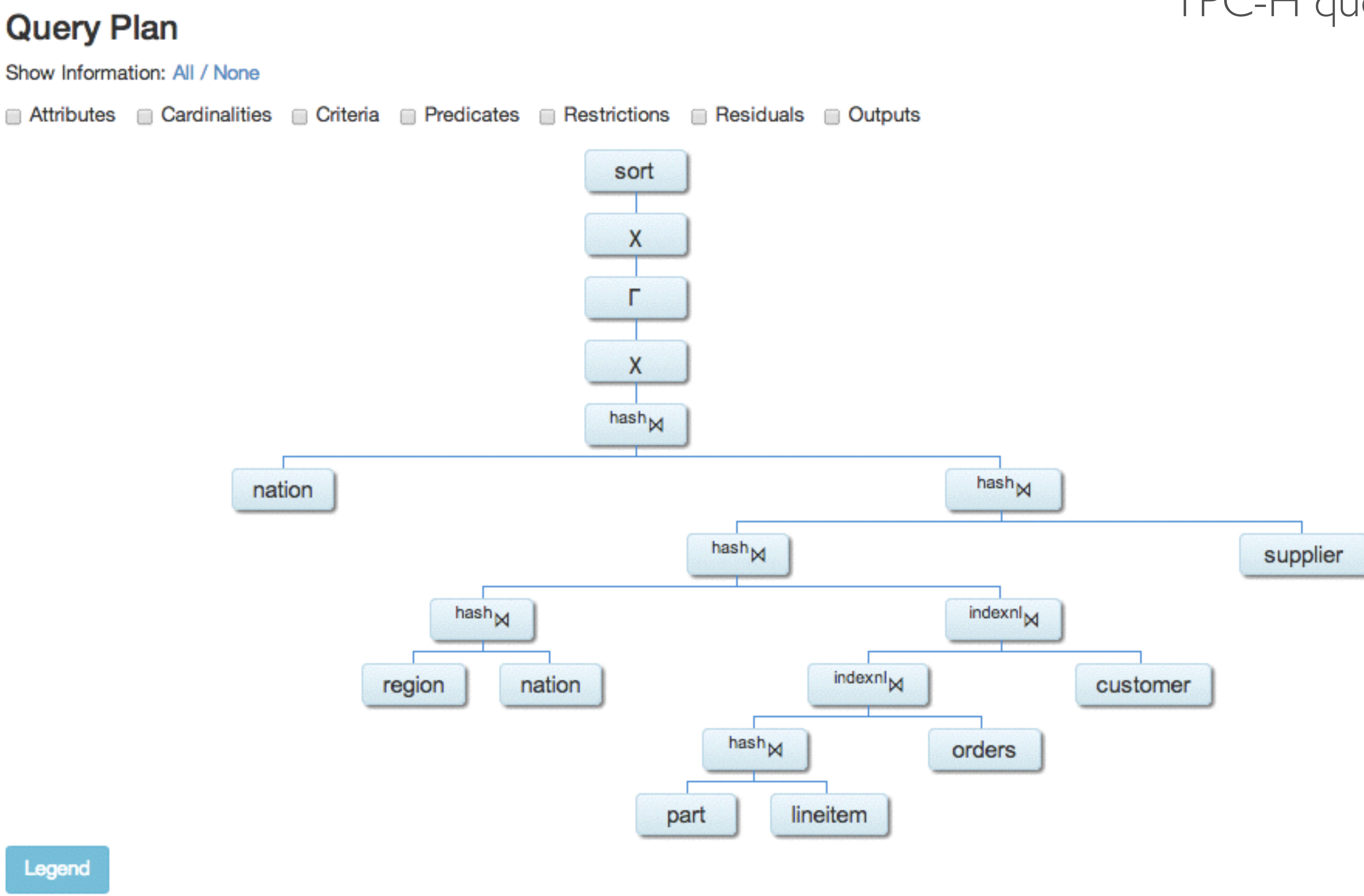
Note: This WebInterface queries a HyPer instance executing queries in a single thread on a low-end server (Intel® Core™ i3-2120 CPU, 16 GB RAM); mind that this interface is thus not intended for benchmarking purposes. ×

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:

```sql
select
        o_year,
        sum(case
                when nation = 'BRAZIL' then volume
                else 0
        end) / sum(volume) as mkt_share
from
        (
                select
                        extract(year from o_orderdate) as o_year,
                        l_extendedprice * (1 - l_discount) as volume,
                        n2.n_name as nation
                from
                        part,
                        supplier,
                        lineitem,
                        orders,
                        customer,
                        nation n1,
                        nation n2,
                        region
                where
                        p_partkey = l_partkey
                        and s_suppkey = l_suppkey
                        and l_orderkey = o_orderkey
                        and o_custkey = c_custkey
                        and c_nationkey = n1.n_nationkey
                        and n1.n_regionkey = r_regionkey
                        and r_name = 'AMERICA'
                        and s_nationkey = n2.n_nationkey
                        and o_orderdate between date '1995-01-01' and date '1996-12-31'
                        and p_type = 'ECONOMY ANODIZED STEEL'
        ) as all_nations
group by
        o_year
order by
        o_year
```

Query | Show Query Plan | TPC-H Schema | Uni Schema | Insert TPC-H Query

http://www.hyper-db.com/interface.html



http://www.hyper-db.com/interface.html

Query Result

Compilation time: 48.5447 ms

Execution time: 50.4597 ms

Result set size: 2

o_year	mkt_share
1995	0.0344
1996	0.0414

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

Conclusion

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

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

References

1) Kemper, A.; Neumann, T., "HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots", ICDE, 2011
2) Neumann, T., "Efficiently compiling efficient query plans for modern hardware", VLDB, 2011
3) Mühe, H; Kemper, A.; Neumann, T., "Executing Long-Running Transactions in Synchronization-Free Main Memory Database Systems", CIDR, 2013
4) Leis, V.; Kemper, A.; Neumann, T., "The adaptive radix tree: ARTful indexing for main-memory databases", ICDE, 2013
5) Leis, V.; Kemper, A.; Neumann, T., "Exploiting Hardware Transactional Memory in Main-Memory Databases", ICDE, 2014
6) Funke, F.; Kemper, A.; Neumann, T., "Compacting Transactional Data in Hybrid OLTP&OLAP Databases", VLDB 2012
7) Mühlbauer, T.; Rödiger, W.; Seilbeck, R.; Reiser, A.; Kemper, A.; Neumann, T., "Instant Loading for Main Memory Databases", PVLDB 2013, VLDB 2014
8) Mühlbauer, T.; Rödiger, W.; Reiser, A.; Kemper, A.; Neumann, T., "ScyPer: Elastic OLAP Throughput on Transactional Data", DanaC, 2013
9) Rödiger, W.; Mühlbauer, T.; Unterbrunner, P.; Reiser, A.; Kemper, A.; Neumann, T., "Locality-Sensitive Operators for Parallel Main-Memory Database Clusters", ICDE, 2014