In-Memory Columnar Databases HyPer

Arto Kärki
University of Helsinki
30.11.2012

In-Memory Columnar DBs HyPer

- Introduction
- Columnar Databases
- Design Choices
- Data Clustering and Compression
- Conclusion

In-Memory Columnar DBs HyPer Introduction

The relational database systems are today mostly separated in two different technical solutions because of the increased number of rows and for performance issues.

OLTP (OnLine Transaction Processing) is designed for fast row inserts, updates and selections.

OLAP (OnLine Analytic Processing) is designed for long lasting queries.

In-Memory Columnar DBs HyPer Introduction

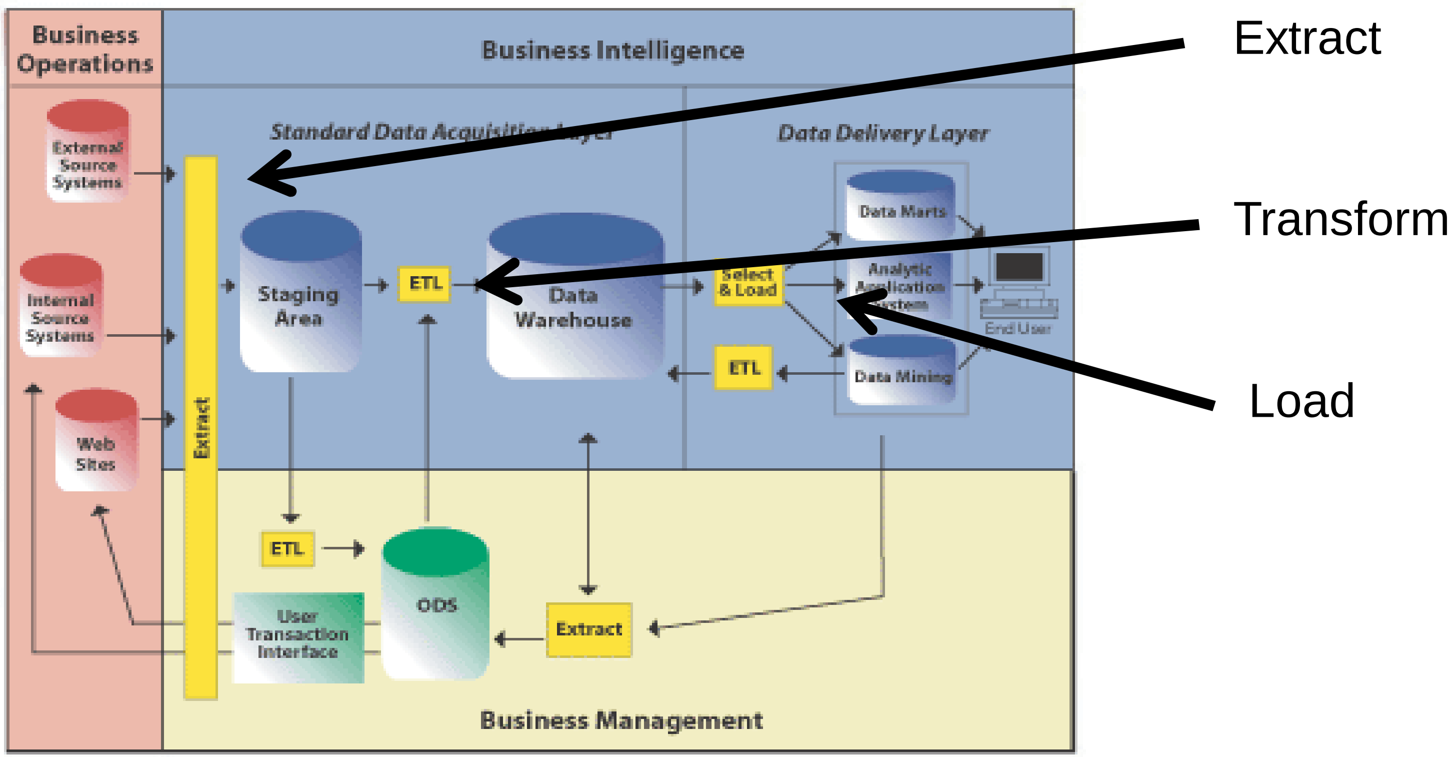
Problematic ETL-mechanism

An OLAP database is populated with rows from OLTP database. This means complicated data extracting, transforming and loading mechanism. This mechanism is usually slow and can be made only once a day submitting batch jobs during night time.

Data is not fresh any more for business intelligence calculations.

In-Memory Columnar DBs HyPer Introduction

ETL example



In-Memory Columnar DBs HyPer Introduction

The HyPer's challenge, how one could:

1. avoid ETL overhead and merge OLTP and OLAP back in the same database system.
2. manage to preserve the fast OLTP transactions and at the same time achieve up-to-date data for OLAP queries.
3. satisfy business intelligence calculations.

In-Memory Columnar DBs HyPer Introduction

Next:

- Some aspects on columnar databases
- Hyper's design choices
- Hyper's data clustering and compression

And finally:

- Conclusion

In-Memory Columnar DBs HyPer

Introduction

$\boxed{\text{Columnar Databases}}$

Design Choices

Data Clustering and Compression

Conclusion

In-Memory Columnar DBs HyPer Columnar Databases

H.Plattner:

- Early tests at SAP and HPI with in-memory databases of the relational type based on row storage did not show significant advantages over leading RDBMSs with equivalent memory caching.
- The alternative idea to investigate the advantages of using column store databases for OLTP was born.
- Column storage was successfully used for many years in OLAP and really surged when main memory became abundant.

In-Memory Columnar DBs HyPer Columnar Databases

Abadi et al.:

- Store each column separately, with attribute values belonging to the same column stored contiguously as opposed to traditional db systems that store entire rows one after the other.

- Reading a subset of table's columns becomes faster when scanning multiple columns.

- Potential expense of exessive disk-head seeking from column to column for scattered reads or updates.

In-Memory Columnar DBs HyPer
Columnar Databases

Abadi et al.:

Two of the most-often cited disadvantages:

- write operations (inserted tuples have to be broken up into their component attributes and each attribute must be written separately).

- the dense-packed data layout makes moving tuples within page nearly impossible.

In-Memory Columnar DBs HyPer

Introduction

Columnar Databases

Design Choices

Data Clustering and Compression

Conclusion

In-Memory Columnar DBs HyPer Design Choices

What's HyPer?

It is a hybrid OLTP&OLAP main memory database system. And it is columnar in order *to achieve best possible query execution performance for OLAP applications.*

Next: Hyper's performance is due to the following design choices.

In-Memory Columnar DBs HyPer Design Choices 1

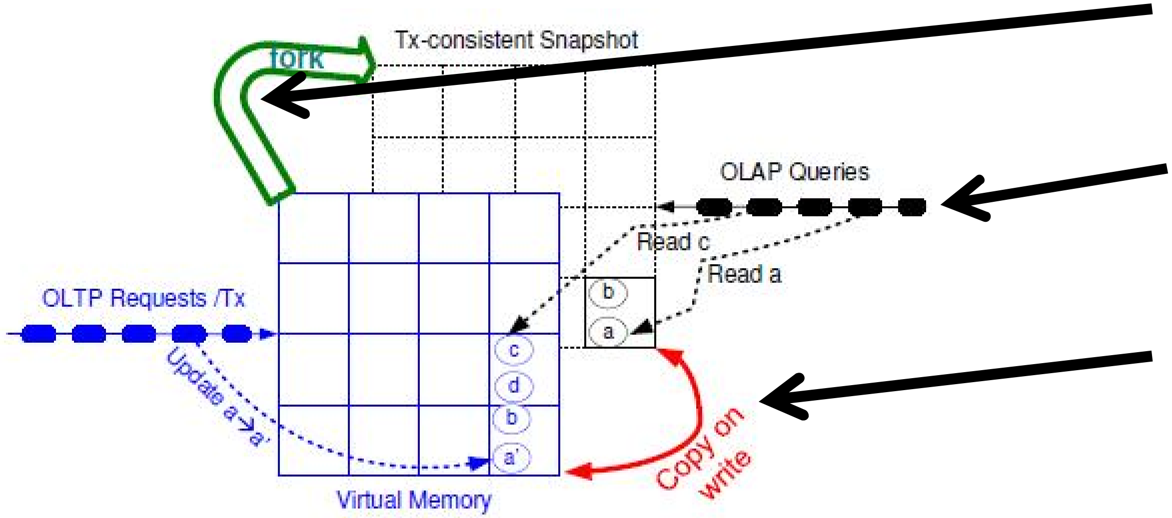
HyPer relies on in-memory data management without the ballast of traditional database systems caused by DBMS-controlled page structures and buffer management.

The SQL table definitions are transformed into simple vector-based virtual memory representations – which constitutes a column-oriented physical storage scheme.

In-Memory Columnar DBs HyPer Design Choices 2

The OLAP processing is separated from the mission-critical OLTP transaction processing by fork-ing virtual memory snapshots. Thus, no concurrency control mechanisms other than the hardware-assisted VM management are needed to separate the two workload classes.

In-Memory Columnar DBs HyPer Design Choices 3



Fork

OLAP queries

Hardware assisted copy

In-Memory Columnar DBs HyPer Design Choices 4

Transactions and queries are specified in SQL and are efficiently compiled into LLVM assembly code. The transactions are specified in an SQL scripting language and registered stored procedures.

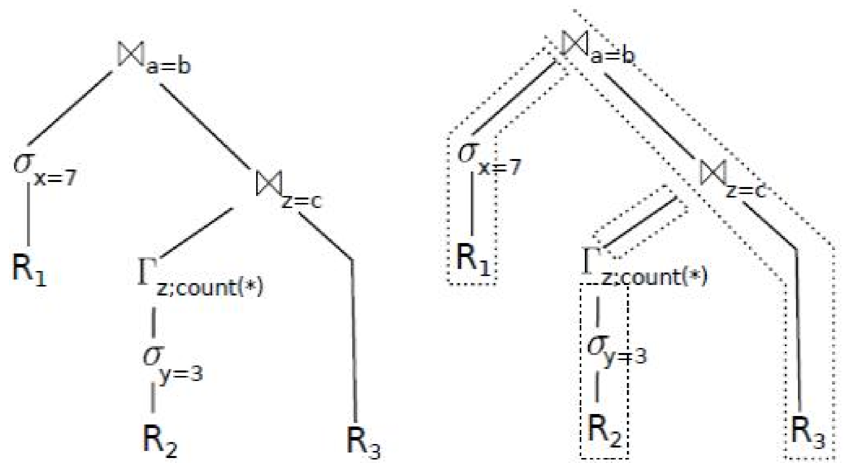
The query evaluation follows a data-centric paradigm by applying as many operations on a data object as possible in between pipeline breakers.

In-Memory Columnar DBs HyPer Design Choices 5

Example Query

```
select      *
from        R1,R3,
            (select     R2.z,count(*)
             from       R2
             where      R2.y=3
             group by   R2.z) R2
where       R1.x=7 and R1.a=R3.b and R2.z=R3.c
```

Example execution plan



original   with pipeline boundaries

Compiled query

initialize memory of $\bowtie_{a=b}$, $\bowtie_{c=z}$, and Γ_z

for each tuple t in R_1
    if $t.x = 7$
        materialize t in hash table of $\bowtie_{a=b}$

for each tuple t in R_2
    if $t.y = 3$
        aggregate t in hash table of Γ_z

for each tuple t in Γ_z
    materialize t in hash table of $\bowtie_{z=c}$

for each tuple t_3 in R_3
    for each match t_2 in $\bowtie_{z=c}[t_3.c]$
        for each match t_1 in $\bowtie_{a=b}[t_3.b]$
            output $t_1 \circ t_2 \circ t_3$

In-Memory Columnar DBs HyPer Design Choices 6

LLVM assembler

- The experiments have shown that data-centric query processing is a very efficient query execution model. DBMS can achieve a query processing efficiency that rivals hand-written C++ code.
- The data-centric compilation approach is promising for all new database projects. By relying on mainstream compilation frameworks the DBMS automatically benefits from future compiler and processor improvements without re-engineering the query engine.

In-Memory Columnar DBs HyPer Design Choices 7

As in VoltDB, the parallel transactions are separated via lock-free admission control that allows only nonconflicting transactions at the same time. Parallelism in this serial execution model is achieved by logically partitioning the database and admitting multiple partition-constrained transactions in parallel. However, for executing partition-crossing transactions the scheduler resorts to strict serial execution, rather than costly locking-based synchronization.

In-Memory Columnar DBs HyPer Design Choices 8

HyPer relies on logical logging where, in essence, the invocation parameters of the stored prosedures / transactions are logged via a high-speed network. The serial execution model in combination with partitioning and group committing achieves extreme scalability in terms of transaction throughput – without compromising the ”holy grail” of ACID.

In-Memory Columnar DBs HyPer Design Choices 9

While in-core OLAP query processing can be based on sequential scans, this is not possible for transaction processing as we require execution times of a few microseconds only. Therefore, HyPer has deleloped sophisticated main-memory indexing structures based on hashing, balanced search trees and radiax trees. Hash indexes are dispensable for exact match (e.g., primary key) accesses that are most common in transactional processing while the tree structured indexes are essential for smallrange queries, that are commonly encouterd in transactional scripts as well.

In-Memory Columnar DBs HyPer

Introduction

Columnar Databases

Design Choices

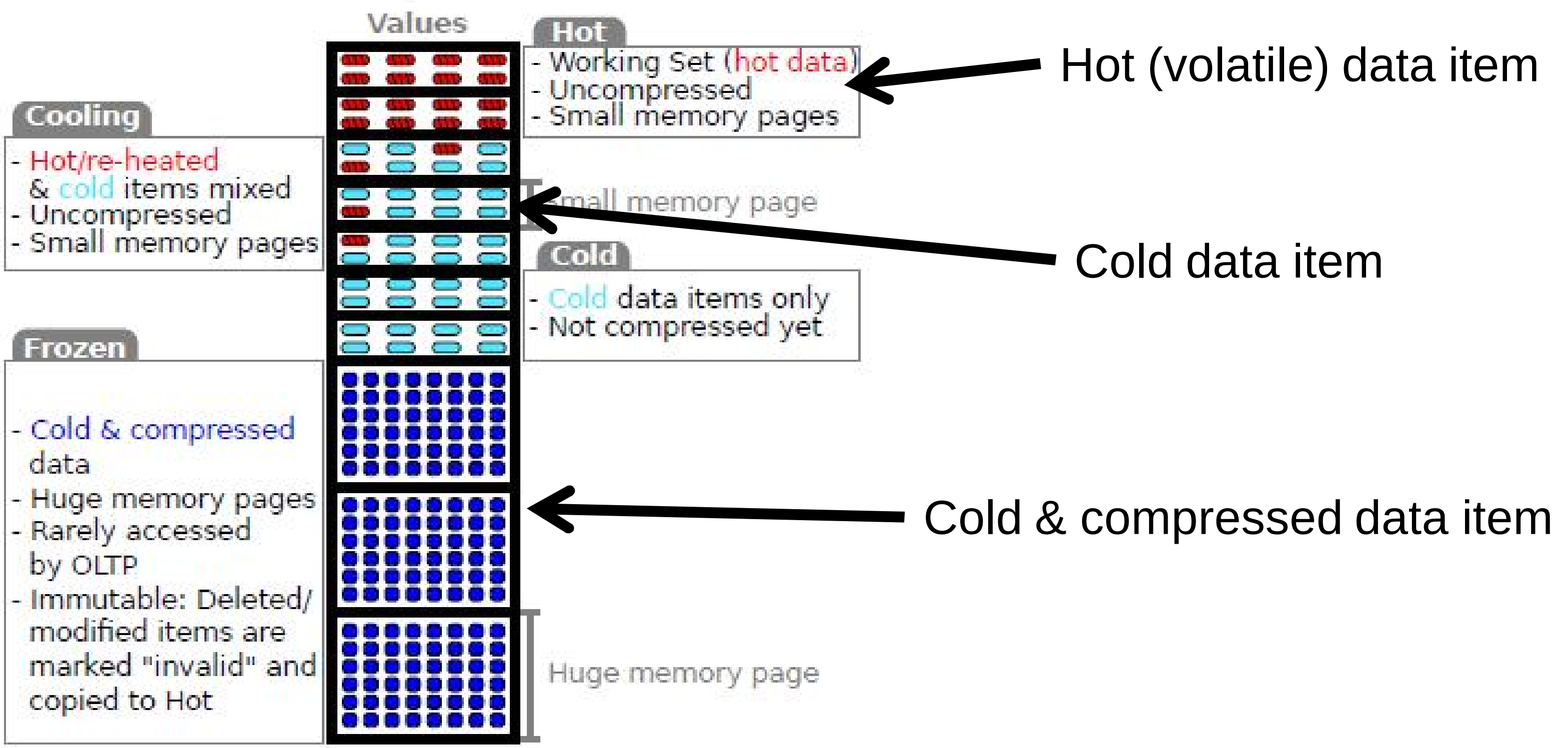
$\boxed{\text{Data Clustering and Compression}}$

Conclusion

In-Memory Columnar DBs HyPer Data Clustering and Compression

HyPer's approach to compression in hybrid OLTP & OLAP column stores is based on the observation that while OLTP workloads frequently modify the dataset, they often follow the working set assumption: only a small subset of the data is accessed and an even smaller subset of this working set is being modified.

In-Memory Columnar DBs HyPer Data Clustering and Compression



In-Memory Columnar DBs HyPer Data Clustering and Compression

Hot/cold clustering is an elegant solution to this problem, as the cold bulk of the data can be stored on huge memory pages while the hot, frequently modified working set remains on regular memory pages that can be replicated inexpensively.

The frozen, huge data pages are never modified; if a frozen data object is changed, after all, it is invalidated in the frozen partition and re-inserted into the hot working set.

In-Memory Columnar DBs HyPer

Introduction

Columnar Databases

Design Choices

Snapshotting

Data Compression

Conclusion

In-Memory Columnar DBs HyPer Conclusion

- A reunion of OLTP & OLAP systems via snapshotting.
- Queries compiled into machine code using the optimizing LLVM compiler => the DBMS can achieve a query processing efficiency that rivals hand-written C++ code.
- Hot/cold clustering to store frequently accessed tuples together on regular memory pages while cold, immutable tuples can reside on huge pages => advantageous combination of page table size and thus snapshot creation costs.

In-Memory Columnar DBs HyPer

Thank you

In-Memory Columnar DBs HyPer References

T. Cao, M. Salles, B. Sowell, Y. Yue, J. Gehrke, A.Demers, and W. White. Fast Checkpoint Recovery Algorithms for Frequently Consistent Applications. In SIGMOD, 2011.

Florian Funke, Alfons Kemper, Thomas Neumann: Compacting Transactional Data in Hybrid OLTP & OLAP Databases. PVLDB 5(11):1424-1435(2012).

S. H´eman, M. Zukowski, N. J. Nes, L. Sidirourgos,and P. A. Boncz. Positional Update Handling in Column Stores. In SIGMOD, pages 543–554, 2010.

Alfons Kemper, Thomas Neumann, Florian Funke, Viktor Leis, Henrik Mühe: HyPer:Adapting Columnar Main-Memory Data Management for Transactional AND Query Processing. IEEE Data Eng. Bull. (DEBU)35(1):46-51(2012).

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

Hasso Plattner: A common database approach for OLTP and OLAP using an in-memory column database. SIGMOD 2009:1-2.

http://www.information-management.com/issues/20031101/7607-1.html