SAP HANA - Main Memory Technology: A Challenge for Development of Business Applications

Jürgen Primsch, SAP AG
July 2011
Why In-Memory?
Information at the Speed of Thought

Imagine access to business data, supporting analytical queries on the transactional data, no need for data replication, and zero response time

How would this change the way you work with data?
How would this change your perception of information availability?
How would this new information influence your decisions?

In-memory databases are a technology with huge disruption potential, we need to assume a competitive leadership position.
Motivation: Make Use of Modern Hardware

Game Changing Hardware Trends
- Disk is Tape
- Main memory affordable
- CPU: more cores, no clock rate increase
- Software makers need to react

Past
- CPU clock rate growing
- Software runs faster without change

Today
- CPU clock rate growth is flat
- Number of cores increases
- Write software that scales with number of cores!
Massive amount of memory and parallel processing power for under $1M

1 server ...8 CPUs, each CPU with 8 cores for $100,000

- can execute 64 threads in parallel
- can hold memory modules adding up to ~2TB today

100GB/sec data throughput per server

8 Server blades ...for less than $1M

- can execute 512 threads in parallel
- can hold memory modules adding up to ~16TB
Simplified Memory Hierarchy (based on Intel Nehalem)

<table>
<thead>
<tr>
<th>Core</th>
<th>1st Level Cache</th>
<th>2nd Level Cache</th>
<th>Shared 3rd Level Cache</th>
</tr>
</thead>
</table>

**CPU**

<table>
<thead>
<tr>
<th>Core</th>
<th>1st Level Cache</th>
<th>2nd Level Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>64KB</td>
<td>2 ns</td>
</tr>
<tr>
<td></td>
<td>~4 cycles</td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>256KB</td>
<td>5 ns</td>
</tr>
<tr>
<td></td>
<td>~10 cycles</td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>8MB</td>
<td>20 ns</td>
</tr>
<tr>
<td></td>
<td>35-40+ cycles</td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>several GBs up</td>
<td>~200 cycles</td>
</tr>
<tr>
<td></td>
<td>to TBs</td>
<td>~100 ns</td>
</tr>
<tr>
<td>Core</td>
<td>several TB</td>
<td>several million</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cycles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>several ms</td>
</tr>
</tbody>
</table>

Main Memory

Disk
Main Memory Bottleneck

**Disk is Tape**

$10^5$ times slower compared to main memory access

No disk access during normal operation

Use disk as kind of “archive”

**Memory**

Price / MB RAM is going down

64 bit architectures make maximum limit of 4GB main memory obsolete

**Memory Access**

Access to main memory is not arbitrarily fast

Memory bandwidth increases but memory access latency remains the same

Cache misses limit performance

Traditional database: CPU spends 50% of time waiting after cache misses (1999)

Transfer memory-cache is block wise (cache lines), for example 64 bytes

Main memory access benefits from data locality, random access is slow
“In-Memory” Computing

Keep all required data in main memory

- Rarely accessed data can be moved to disk (e.g. after 1 year)
- Compress data in memory
- Disk I/O no longer an optimization target

Cache sensitive data layout

- High locality (data that is needed together is stored together)
- Compression (decompress in cache)

Parallelization

- Data structures allow splitting into pieces that can be processed in parallel
- Avoid locks

Unify OLAP and OLTP systems: combine column and row store

- Column store for reporting like access
- Row store for OLTP like access
Row Based and Column Based Data Storage

<table>
<thead>
<tr>
<th>Country</th>
<th>Product</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>Alpha</td>
<td>3.000</td>
</tr>
<tr>
<td>US</td>
<td>Beta</td>
<td>1.250</td>
</tr>
<tr>
<td>JP</td>
<td>Alpha</td>
<td>700</td>
</tr>
<tr>
<td>UK</td>
<td>Alpha</td>
<td>450</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Aggregate sales figures:

Sum(sales):
Sales: 4 byte, country: 2 byte, product: 10 byte
Cache line: 64 Byte → 48 byte waste = 75%

Row-Based

US | Alpha | 3.000 |
US | Beta  | 1.250 |
JP | Alpha | 700   |
UK | Alpha | 450   |

Columnar

US | US | JP | UK | ... | Alpha | Beta | Alpha | Alpha | ... | 3.000 | 1.250 | 700 | 450 |

Data stored in contiguous memory
High locality

Random access!
Row Based and Column Based Data Storage

**Column Store**
- Many calculations on single or few columns only
- Searches based on values of a few columns
- Big number of columns
- Big number of rows and columnar operations are required
  - aggregate, scan, etc.
- High compression rates possible
  - Most columns contain only few distinct values

**Row Store**
- Application often needs to process single records at one time
  - many selects and/or updates of single records
- Application typically needs to access the complete record
- Columns contain mainly distinct values
- Aggregations and fast searching not required
- Small number of rows (e.g. configuration tables)
SAP HANA database
Overall Architecture of the engine
Row Store Architecture

Row Store
Data records managed in pages
Temporary versions in separate memory
Index only in memory
Optimistic latch free index access
Loads all tables at startup
Incremental transaction logs
- Parallel write and restart
Data persisted during savepoint operation
Column Store Architecture

Column Store

Own optimizer and execution control
OLAP engine: Special optimizer and operators

For each column
- Compressed main storage
- Delta storage with only basic compression
- Write operations go into delta only
- Merge operation re-encodes main area
- Optional history storage (also main + delta)

Delta log written per table

Main area persisted during merge operation

Write and read possible during merge

Loads tables on demand (preload configurable)
SAP HANA database - Assets

High Performance and Scalability
- Compression, cache-aware storage
- Executing application logic inside database
- Distribution

Reducing Complexity and Cost
- Hybrid: column based, row based (future: object store and disk based) in one system
- Analytics and OLTP in one system

Support for OLAP
- MDX, calculation engine, OLAP views

Compatibility and Standard DBMS Features
- SQL, ACID
- run SAP applications that use Open SQL without changes.

Multi Tenant Support

Support For Temporal Tables
- “time travel”, time slider applications, reporting using historical data, change recording

Main memory text search engine
SAP HANA database – A Distributed Database
Different Options

Data Distribution – No Tenant Separation

Multi Tenant System

Default: tenants are not distributed across multiple servers
SAP HANA database – A Distributed Database
Topology Information / metadata

Topology and distribution Information

Master Name Server

Host

Slave Name Server

Topography and distribution Information (replicated)

NewDB Database Server

Fast read access via shared memory

Host

Slave Name Server

Topography and distribution Information (replicated)

NewDB Database Server

Host

Slave Name Server

Topography and distribution Information (replicated)

NewDB Database Server
## Tenant Separation

### Table T1

<table>
<thead>
<tr>
<th>Client</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>765</td>
<td>y</td>
<td>c</td>
</tr>
<tr>
<td>200</td>
<td>770</td>
<td>z</td>
<td>a</td>
</tr>
<tr>
<td>100</td>
<td>701</td>
<td>w</td>
<td>c</td>
</tr>
<tr>
<td>300</td>
<td>733</td>
<td>x</td>
<td>b</td>
</tr>
<tr>
<td>300</td>
<td>777</td>
<td>z</td>
<td>a</td>
</tr>
<tr>
<td>200</td>
<td>800</td>
<td>z</td>
<td>b</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### T1 in Tenant 100

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>765</td>
<td>y</td>
<td>c</td>
</tr>
<tr>
<td>701</td>
<td>w</td>
<td>c</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### T1 in Tenant 200

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>770</td>
<td>z</td>
<td>a</td>
</tr>
<tr>
<td>800</td>
<td>z</td>
<td>b</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### T1 in Tenant 300

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>733</td>
<td>x</td>
<td>b</td>
</tr>
<tr>
<td>777</td>
<td>z</td>
<td>a</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**NewDB tenant**

- Separate database process
- Separate data volumes
- Separate instances of tenant-dependent tables
- Separate transaction domain

Defined by central metadata
**Temporal Tables**

**Temporal Table (Logical View)**

<table>
<thead>
<tr>
<th>Row</th>
<th>ID</th>
<th>Text</th>
<th>Size</th>
<th>Valid-From</th>
<th>Valid-To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>546</td>
<td>Shirt</td>
<td>XL</td>
<td>2010-09-11 08:30</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>712</td>
<td>Shoe</td>
<td>M</td>
<td>2010-10-06 11:42</td>
<td>∞</td>
</tr>
<tr>
<td>3</td>
<td>913</td>
<td>Hat</td>
<td>L</td>
<td>2010-10-15 16:11</td>
<td>∞</td>
</tr>
</tbody>
</table>

UPDATE article SET SIZE = ‘S’ WHERE ID=’712’

<table>
<thead>
<tr>
<th>Row</th>
<th>ID</th>
<th>Text</th>
<th>Size</th>
<th>Valid-From</th>
<th>Valid-To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>546</td>
<td>Shirt</td>
<td>XL</td>
<td>2010-09-11 08:30</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>712</td>
<td>Shoe</td>
<td>M</td>
<td>2010-10-06 11:42</td>
<td>2010-10-26 17:05</td>
</tr>
<tr>
<td>3</td>
<td>913</td>
<td>Hat</td>
<td>L</td>
<td>2010-10-15 16:11</td>
<td>∞</td>
</tr>
<tr>
<td>4</td>
<td>712</td>
<td>Hat</td>
<td>S</td>
<td>2010-10-26 17:05</td>
<td>∞</td>
</tr>
</tbody>
</table>

DELETE FROM article WHERE ID=’546’

<table>
<thead>
<tr>
<th>Row</th>
<th>ID</th>
<th>Text</th>
<th>Size</th>
<th>Valid-From</th>
<th>Valid-To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>546</td>
<td>Shirt</td>
<td>XL</td>
<td>2010-09-11 08:30</td>
<td>2010-10-26 17:30</td>
</tr>
<tr>
<td>2</td>
<td>712</td>
<td>Shoe</td>
<td>M</td>
<td>2010-10-06 11:42</td>
<td>2010-10-26 17:05</td>
</tr>
<tr>
<td>3</td>
<td>913</td>
<td>Hat</td>
<td>L</td>
<td>2010-10-15 16:11</td>
<td>∞</td>
</tr>
<tr>
<td>4</td>
<td>712</td>
<td>Hat</td>
<td>S</td>
<td>2010-10-26 17:05</td>
<td>∞</td>
</tr>
</tbody>
</table>

Currently only in column store
“In-Memory” Computing

Business Database
Move data-intensive operations to the data layer
• Do the relational operations at database level (i.e. select for all entries)
• Hierarchy handling (i.e. HR cost center hierarchy)
• Reporting and planning functionality
• Planning engine

Transactional behavior
• Unify DB and application transaction
• Push snapshot isolation to the database / remove transactional buffer in the appserver layer

Business Function Library
• Provide business functions in the DB layer like
  – Currency /Unit conversion
  – Date / time /fiscal period /calendar calculation
  – Statistical functionality

Deep integration with the application server
• Fast communication layer
• SQL extensions (SQL script)
• New data types (text, GUID, …)
Thank You!

Questions?

Jürgen Primsch
SAP AG