Enterprise Operational Analytics Data Management Systems
From Vision to Reality

Hinnerk Gildhoff, SAP SE
December 14, 2016
Agenda

• Recap: Traditional Enterprise Data Management Architectures
• Vision: Enterprise Operational Analytics Data Management Systems
• Reality: Design and Implementation Aspects - Case Study: SAP HANA
• SAP HANA Platform
Traditional Enterprise Data Management Architectures

- **Transactional System for Online Transaction Processing (OLTP)**
  - Short-running, trivial statements; frequent updates; high amount of concurrent users

- **Data Warehouse for Online Analytical Processing (OLAP)**
  - Long-running, complex statements; (almost) read-only; few users

- **Nightly Data Extraction, Transfer, and Loading Between Systems (ETL)**
  - Transformation from OLTP-friendly to OLAP-friendly format; indexing; pre-aggregation

![Diagram of Traditional Enterprise Data Management Architectures]

- OLTP
- Extract, Transform, Load (ETL)
- OLAP
Challenges of Traditional Architectures

- **High Costs / TCO**
  - Two (different?) database management systems
  - Two times the license costs
  - Two times the hardware
  - Two times the administration

- **No Data Freshness for Reporting**
  - Data is updated only nightly
  - Reporting during the day always sees stale data

- **ETL Processes are a Pain**
  - Complex to design and maintain
  - Need to complete in daily maintenance window (e.g. <8 hours)
  - Data preparation necessary for reporting
  - Slow reaction to changing business demands (no ad-hoc reporting)
Vision: Enterprise Operational Analytics Data Management Systems

- Vision: One Combined DBMS for OLTP and OLAP Workloads
- Foundation: In-Memory Columnar Database
- Goals: Flexible Reporting, Lower TCO, No ETL
- SIGMOD 2009 Keynote: “Information at your fingertips”
Stonebraker: One Size does not Fit All!

- **Specialized Engines Beat General-Purpose DBMS by Factors**

- **Proofpoints**
  - StreamBase – Event stream processing
  - Vertica – Columnar Storage for OLAP
  - VoltDB – Lock-free, highly efficient OLTP processing
  - SciDB – Array-processing for multidimensional datasets

- **A.M. Turing Award 2014 (ACM)**
  - Many significant contributions to the DBMS field
  - Successful both in academia and commercially
From Vision To Reality: Objectives for HANA

- **Good Enough for OLTP**
  - Unrealistic to beat a specialized, pure OLTP engine
  - But: Be able to sustain a typical enterprise workload
  - Example: 40,000 SQL statements / second

- **Excel in Analytics**
  - Flexible reporting without pre-computation / aggregates
  - Leverage modern hardware (multicore + large DRAM capacity)
  - Marketing: „Subsecond everything“

- **Academia Highly Sceptical**
  - „I wouldn't want to waste a PhD on this“
  - In-memory column stores only used for analytics so far
Technical Deep-Dive

Design and Implementation Aspects of SAP HANA
In-Memory DBMS: Modern Hardware / Technological Context

- **Multi-Core CPUs**
  - Clock speed does not increase
  - More CPU cores
  - Small cache in CPU

- **Large Memory**
  - 1 TB RAM widely available
  - Slow compared to CPU

- **Disk**
  - “Unlimited” Size
  - Increasing Latency gap
HANA Table Types

- **In-memory row-based storage**
  - Fast OLTP operations (SELECT SINGLE, small K/FK joins, ...)
  - No compression

- **In-memory columnar storage**
  - Fast OLAP operations (massive scans, aggregation, …)
  - Read-optimized, immutable Main Store
  - Write-optimized Differential Store (Delta)
  - Slow on OLTP operations

- **Focus is on columnar storage**
Main: Fast Queries

• **Read-Optimized, Immutable Data Store**

• **Dictionary Compression**
  • All data in columnar tables is dictionary compressed
  • Domain coding (aka dictionary compression) as compression scheme
  • Redundant literals only stored once (dictionary)
  • Dictionary is prefix-compressed
  • Dictionary is sorted (only for main)

• **Reduces Storage Consumption Significantly**
  • Example: Up to factor 5 for typical SAP data schemes
Delta: Update Support

- **Write-Enabled Table Fragment Taking Updates**
  - Only update operation on main is deleting rows
  - UPDATEs modelled as DELETE+INSERT

- **Dictionary not Sorted**
  - No need to recode indexvector upon delete/insert

- **Additional B-Tree for Efficient Lookup**
  - Allows to quickly retrieve VID for value
  - Essential for fast unique checks upon insert
  - Can be used for range queries
Delta Merge

- **Consolidation of Delta and Main into new Main**
  - Improves query performance (especially for analytics)
  - Reduces memory footprint (no B-Tree necessary)

- **Automatically Done by the System Based on Cost-Based Decision Function**
  - Considers delta:main ratio, size in RAM and disk, system workload
  - Done on a per table-basis (actually: partition-based), parallelized on column-level
Optimize Compression for the Main Table Fragment

- **Beyond Dictionary Compression**
  - Sparse coding, cluster coding, indirect coding, run-length coding, …
  - Best compression scheme to use depends on data distribution and ordering

- **Example: Sparse-Coding**
  - Remove the most frequent value
  - Store the removed positions
  - Additionally: Shorten the positions bit vector if the most frequent value is also at the beginning

![Sparse-Coding Example](image)
Challenges for Achieving High Compression Rates

Most Compression Techniques Require Data to be Sorted Appropriately
• Move all appearances of the most frequent value to the top (Prefix-Coding)
• Form blocks of as few as possible values

Issues
• Sorting of one column depends on the sorting of another column
• Solution space grows exponentially with the number of columns
  → Finding the optimal solution is infeasible (NP-complete)

Solution
• Greedy heuristics for a good approximation
• Find a compromise between reduced run time and reduced memory consumption
Parallelization at All Levels

- **Query Execution**
  - Multiple User Sessions → Inter-Query Parallelism
  - Concurrent Operations Within a Query → Intra-Query Parallelism/Inter-Operator Parallelism
  - Multiple Threads for one Operator (e.g. aggregation) → Intra-Operator Parallelism

- **Hardware**
  - Multi-threading at processor core level
  - Vector processing (SIMD)
Single Instruction Multiple Data (SIMD)

Scalar processing
- Traditional mode
- One instruction produces one result

SIMD processing
- With Intel® SSE / AVX / AVX2
- One instruction produces multiple results
Algorithm Example: Concurrent Aggregation

• Very Similar to Map and Reduce

• Map Step Processes Fact Table and Generates Local Results

• Reduce Step Joins Local Results

• No Synchronization Required

• Scales Linearly with Available CPUs
Performance Results for Parallel Aggregation

**Benchmark**
- Cubesize: 228 213 237 rows
- Resultsize: 4 469 335 rows
- 32 physical CPUs (64 HT)
Conclusion

- **Operational Analytics DBMS are a Game-Changer for Enterprise Applications**
  - Fast reporting even without indexes / aggregates / materialized views (CRM)
  - Additional insights from reporting on transactional data (HANA Live / Suite on HANA)
  - Complete application redesign (S4HANA)
    - No aggregates (simplification)
    - Code pushdown
    - Fast analytics on current data in real-time
# SAP HANA - The Platform Powers the Digital Transformation

## SAP HANA Platform

**Application Services**
- Web Server
- JavaScript
- Fiori UX
- Graphic Modeler
- Application Lifecycle Management

**Processing Services**
- Spatial
- Graph
- Predictive
- Search
- Text Analytics
- Data Enrichment
- Series Data
- Business Functions

**Integration & Quality Services**
- Data Virtualization
- ELT & Replication
- Data Quality
- Hadoop & Spark Integration
- Remote Data Sync

**Database Services**
- Columnar OLTP+OLAP
- Multi-Core & Parallelization
- Advanced Compression
- Multi-tenancy
- Multi-Tier Storage
- Data Modeling
- Openness
- Admin & Security
- High Availability & Disaster Recovery

© 2016 SAP SE or an SAP affiliate company. All rights reserved.
Spatial Predicates
SQL/MM

$g_1 \text{.ST\_Within}(g_2)$$
\begin{align*}
g_1 \cap g_2 &= g_1 \land l(g_1) \cap E(g_2) = \emptyset
\end{align*}$

$g_1 \text{.ST\_Contains}(g_2)$$
\begin{align*}
g_1 \cap g_2 &= g_2 \land l(g_1) \cap l(g_2) \neq \emptyset
\end{align*}$

$g_1 \text{.ST\_Intersects}(g_2)$$
\begin{align*}
g_1 \cap g_2 &\neq \emptyset
\end{align*}$

$g_1 \text{.ST\_Equals}(g_2)$$
\begin{align*}
g_1 = g_2
\end{align*}$

$g_1 \text{.ST\_Covers}(g_2)$$
\begin{align*}
g_1 \cap g_2 &= g_2 \\
I(g_1) \cap I(g_2) &\neq \emptyset
\end{align*}$

$g_1 \text{.ST\_Covers}(g_2) ^*$
\begin{align*}
g_1 \cap g_2 &= g_2 \\
I(g_1) \cap I(g_2) &\neq \emptyset
\end{align*}

$g_1 \text{.ST\_Crosses}(g_2)$$
\begin{align*}
I(g_1) \cap I(g_2) &\neq \emptyset \land (g_1 \cap g_2 \neq g_1) \land (g_1 \cap g_2 \neq g_2)
\end{align*}$

$g_1 \text{.ST\_Touches}(g_2)$$
\begin{align*}
(g_1 \cap g_2 \neq \emptyset) \land (B(g_1) \cap B(g_2) = \emptyset)
\end{align*}$

$g_1 \text{.ST\_Overlaps}(g_2)$$
\begin{align*}
(l(g_1) \cap l(g_2) \neq \emptyset) \land (l(g_1) \cap E(g_2) \neq \emptyset) \land (E(g_1) \cap l(g_2) \neq \emptyset)
\end{align*}$

$g_1 \text{.ST\_Disjoint}(g_2)$$
\begin{align*}
g_1 \cap g_2 &= \emptyset
\end{align*}$

* No OGC standard
Spatial Column Scan

MULTICORE + SIMD + RAM
Point Column
Normalization

\[ P_1 (13.5, 12.2) \]
\[ P_2 (10.4, 5.50) \]
\[ P_3 (1.10, 1.00) \]
\[ P_4 (14.3, 7.50) \]
\[ P_5 (10.0, 5.50) \]
\[ P_6 (14.3, 5.10) \]
\[ P_7 (1.00, 14.5) \]
\[ P_8 (16.0, 8.00) \]
Window Query
Point Column

\[ \text{scan}(x) \quad \text{AND} \quad \text{scan}(y) \quad \text{AND} \quad \text{full column scan} \]
Sometimes it is not about a tree…
Mapping space to memory location
Window Query

Point Column

scan(y) → AND

scan(x) → AND

scan reduction

© 2016 SAP SE or an SAP affiliate company. All rights reserved.
It even gets better 😊
Point Column – Revisited
Compression

\[ p_1 (13.5, 12.2) \]
\[ p_2 (10.4, 5.50) \]
\[ p_3 (1.10, 1.00) \]
\[ p_4 (14.3, 7.50) \]
\[ p_5 (10.0, 5.50) \]
\[ p_6 (14.3, 5.10) \]
\[ p_7 (1.00, 14.5) \]
\[ p_8 (16.0, 8.00) \]
By Example of X

- *equidistant*
- *sparse values*
By Example of X

compressed curve

equidistant
LIVE demo
**Open HANA Projects**

**Spatial, Graph, and much more…**

**Spatial Web Applications** (Apps, Demos, PoC, …)
- OSM (content, tile server, maps, …)

**GPU** (ST_predicates execution, visualization, …)
- Open Source (QGIS, Geoserver, Mapserver, …)

**OGC Web Services** (WPS, WMS, …)
- 3D Tools Support (AutoCad, Blender, …)

**Cloud Services** (Routing, Mapping, …)
- Spatial Inices (SFC for all geometries, 4D, …)

**Graph Indices** (Reachability, Index updates, …)
- Graph Query Languages
- 3D Maps (point clouds, satelite images, …)
- Big Data (open data, platform, …)
- Graph Query Processing
- Sensor data / streaming (rasberry pi, drones, …)
- Compression

**3D Tools Support** (AutoCad, Blender, …)
- Spatial Inices (SFC for all geometries, 4D, …)
- Rasterdata (Octree, SFC, image processing, …)
- Storage Optimizations
- Temporal Analysis
- Visualization
- Topologies, Networks, Point Clouds
- Spatial Simplifier
Thank You!

SAP HANA
https://sap.hana.com/hana

SAP HANA Express Edition (free)

SAP HANA Academy
www.saphana.com/academy

hinnerk.gildhoff@sap.com