IN-MEMORY DATABASES

Presentation
Seminar «Advanced Database Systems»
Springterm 2014
June 6, 2014
Pascal Roman Artho
University of Applied Sciences Rapperswil
AGENDA

Introduction
  • Motivation
  • Problem Statement

In-Memory Databases
  • Definition, Advantages, Disadvantages
  • Typical Concepts for In-Memory Databases

Products
  • Overview
  • Benchmark Example

Conclusion
INTRODUCTION [1][2]

Prof. Dr. h.c. Hasso Plattner

Michael Stonebraker
MOTIVATION [3]

Memory Hierarchy

- CPU Registers
- CPU Caches
- Main Memory
- Flash
- Hard Disk

lower price / higher latency

higher performance
MOTIVATION [4][5]

Doctor in a Hospital

State of an Online Store Order
Online Transaction Processing (OLTP)

- "Online transaction processing (OLTP) is a mode of processing that is characterized by short transactions recording business events and that normally requires high availability and consistent, short response times. This category of applications requires that a request for service be answered within a predictable period that approaches "real time"."
- "It is common for OLTP systems to have high concurrency requirements, with a read/write ratio ranging from 60/40 to as low as 98/2."

Online Analytical Processing (OLAP)

- "OLAP performs multidimensional analysis of business data and provides the capability for complex calculations, trend analysis, and sophisticated data modeling. It is quickly becoming the fundamental foundation for Intelligent Solutions including Business Performance Management, Planning, Budgeting, Forecasting, Financial Reporting, Analysis, Simulation Models, Knowledge Discovery, and Data Warehouse Reporting. OLAP enables end-users to perform ad hoc analysis of data in multiple dimensions, thereby providing the insight and understanding they need for better decision making."
MOTIVATION

Faster (/ instant) responses

Use available hardware like memory and CPU

Create business values
  ▪ Improve distribution and warehousing
  ▪ Optimize products / service
  ▪ Support management decisions
  ▪ Offer analyses as a service
What is an in-memory database?
What are the differences between a standard database and an in-memory database?
What is the difference between main-memory and in-memory database?
What are the advantages of an in-memory database?
What are the disadvantages of an in-memory database?
How should an in-memory database be designed?
How can a database be run in-memory?
Should every database be run in-memory?
IN-MEMORY DATABASES [3]

Definition

An in-memory database is a database where data is primarily stored in the main (primary) memory (RAM, random-access memory) of a computer.

In-memory emphasize that the database is not primarily stored on disk.
IN-MEMORY DATABASES ADVANTAGES \[3\]

- Faster access to the data without varies of access time
- Calculation power of one or multiple CPU’s can be used

<table>
<thead>
<tr>
<th>Type</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache reference (cached data word)</td>
<td>0.5 ns</td>
</tr>
<tr>
<td>Main memory reference</td>
<td>100 ns</td>
</tr>
<tr>
<td>SDD random read</td>
<td>150'000 ns</td>
</tr>
<tr>
<td>Read 1 MB sequentially from memory</td>
<td>250’000 ns</td>
</tr>
<tr>
<td>Disk seek</td>
<td>10'000’000 ns</td>
</tr>
<tr>
<td>Sending packet from California to Netherlands and back to California again</td>
<td>150’000’000 ns</td>
</tr>
</tbody>
</table>
IN-MEMORY DATABASES

DISADVANTAGES [9]

Volatile type of memory

Alternatives

Non-volatile main memory

Continuously backup the data to one or multiple disks or slave database systems with different power supplies

Uninterruptible power supply

Limitations

When the data is transferred over the network, the gain will be consumed by the additional latency

Changes in the program code and / or the database architecture to use it in-memory

Amount of main memory limits the database
### «WorldPopulation»-Table

<table>
<thead>
<tr>
<th>column name</th>
<th>item size</th>
<th>column size (for eight billion records)</th>
</tr>
</thead>
<tbody>
<tr>
<td>first name</td>
<td>49 bytes</td>
<td>≈ 373’840 MB</td>
</tr>
<tr>
<td>last name</td>
<td>50 bytes</td>
<td>≈ 381’470 MB</td>
</tr>
<tr>
<td>gender</td>
<td>1 byte</td>
<td>≈ 7’630 MB</td>
</tr>
<tr>
<td>city</td>
<td>49 bytes</td>
<td>≈ 373’840 MB</td>
</tr>
<tr>
<td>country</td>
<td>49 bytes</td>
<td>≈ 373’840 MB</td>
</tr>
<tr>
<td>birthday</td>
<td>2 bytes</td>
<td>≈ 15’260 MB</td>
</tr>
<tr>
<td>total</td>
<td>200 bytes</td>
<td>≈ 1’490.12 GB</td>
</tr>
</tbody>
</table>
«WorldPopulation»-Table Sample Data

<table>
<thead>
<tr>
<th>recordID</th>
<th>firstname</th>
<th>lastname</th>
<th>gender</th>
<th>city</th>
<th>country</th>
<th>birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>39</td>
<td>John</td>
<td>Smith</td>
<td>M</td>
<td>Chicago</td>
<td>USA</td>
<td>12.03.1964</td>
</tr>
<tr>
<td>40</td>
<td>Mary</td>
<td>Brown</td>
<td>F</td>
<td>London</td>
<td>UK</td>
<td>12.05.1964</td>
</tr>
<tr>
<td>41</td>
<td>Jane</td>
<td>Doe</td>
<td>F</td>
<td>Palo Alto</td>
<td>USA</td>
<td>23.04.1976</td>
</tr>
<tr>
<td>42</td>
<td>John</td>
<td>Doe</td>
<td>M</td>
<td>Palo Alto</td>
<td>USA</td>
<td>17.06.1952</td>
</tr>
<tr>
<td>43</td>
<td>Peter</td>
<td>Schmidt</td>
<td>M</td>
<td>Potsdam</td>
<td>GER</td>
<td>11.11.1975</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
OPTIMIZED INDEXES [22]

Skip Index

Define borders for a specific attribute

Skip records and use the index to search for specific records, but the table needs to be sorted by this column
### Row Store Data Layout «WorldPopulation»

<table>
<thead>
<tr>
<th></th>
<th>First Name</th>
<th>Last Name</th>
<th>Country</th>
<th>Gender</th>
<th>City</th>
<th>Birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 8 x 10^9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Column operation

- A
- B
- C
- A
- B
- C
- A
- B
- C
- ... (repeated)

#### Row operation

- A
- B
- C
- A
- B
- C
- A
- B
- C
- ... (repeated)

---

**June 6, 2014**

**IN-MEMORY DATABASES - PRESENTATION - PASCAL ROMAN ARTHO**
## DATA LAYOUT \[3\]

### Full Table Scan for gender (Row Store)

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Country</th>
<th>Birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row $8 \times 10^9$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Data loaded and used
- Data loaded but not used

### Full Table Scan with Stride Access for gender (Row Store)

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Country</th>
<th>Birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row $8 \times 10^9$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Data not loaded
- Data loaded used
- Data loaded but not used
DATA LAYOUT [3]

Full Column Scan for gender (Column Store)

Column Store Data Layout

Column operation

Row operation
**Trade off**

Most times programs think in records but can operate column wise

Best approach: Hybrid layout depending on the SQL queries
DATA COMPRESSION \([3][10]\)

**Dictionary encoding**

Represent one million cities with an index of 24 bits or three bytes instead of 49 bytes per record

<table>
<thead>
<tr>
<th>49 bytes</th>
<th>46 bytes (93.9%)</th>
</tr>
</thead>
</table>

**Lempel-Ziv 77**

Compresses the data by replacing repeated data with a reference to the existing data

<table>
<thead>
<tr>
<th>01234567</th>
<th>ANAN → (0,0,A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANANA</td>
<td>NANA → (0,0,N)</td>
</tr>
<tr>
<td>ANAS</td>
<td>ANAS → (6,2,A)</td>
</tr>
<tr>
<td>ANANA</td>
<td>S → (0,0,S)</td>
</tr>
<tr>
<td>ANANAS</td>
<td>→ (-,-,-,-)</td>
</tr>
</tbody>
</table>
BUFFER MANAGEMENT [3]

Not a good idea to overload a computer, otherwise the data begins to swap to disk and the capacity goes down.

Divide the data into two groups

- Hot data
  - important for the daily operative business
- Cold data
  - not important for the daily operative business, but relevant for history reasons and analyses

Reorganization depending on the life cycle of data and the amount of free memory.

June 6, 2014
Cost estimation depends on the parallelisms and the best access path to the data

Only the data that really is required should be accessed

Parallelize as much as possible

Materialization

- Early Materialization
  lookup the values early and work on the materialized values

- Late Materialization
  lookup the values as late as possible and work on the positional information
TRANSACTION MANAGEMENT [3]

Inserts and updates for the «WorldPopulation»-Table in case of a birth or decease

Invalidate the old value and insert a new one instead of updating

Set a flag respectively a timestamp to mark a record as invalid

Primary and differential table

- Primary table holds all records that have not been updated since the last merge process, large amount of records
- Differential table holds all records that have been updated since the last merge process, lower amount of records
- Merge process merges the primary and differential table and sorts the records
DURABILITY [3]

Obvious Approach: logical logging
Write all updating SQL statements and their parameters into a log file
Write-ahead logging technique (WAL): all modifications are written to a log before they are applied

Improved Approach: optimized logical logging
For dictionary-encoded columns: separating the logs to dictionary, value and transaction logs
Data can be recreated in parallel
Smaller log files
DURABILITY [3]

Snapshots

After each merge process, snapshots of the primary data are taken and written to disk.

Changes for the differential table are stored in logs.

Only log files after the snapshot are needed to be replayed.

Three way replication system

One master and two slaves.

A slave can be declared as new master and fulfill the work.

Allows to guarantee the availability.
ACCESSIBILITY \([3][11]\)

Increasing the speed of a database always increases the quality

**Do not block or prioritize users** while using the database system

Users start to analyze the data if the system is able to handle all queries with fast responses
PRODUCTS [12-17]

Commercial
- SAP HANA
- Oracle TimesTen In-Memory Database
- Microsoft SQL Server 2014

Open Source
- PostgreSQL
- VoltDB
- SQLite
**BENCHMARK EXAMPLE** \[18\]

SQLite

Microsoft SQL Server 2014

<table>
<thead>
<tr>
<th>System</th>
<th>Fujitsu CELSIUS W530</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel® Xeon® CPU E3-1245 v3 @ 3.4 GHz, Quadcore</td>
</tr>
<tr>
<td></td>
<td>L1-Cache: 256 KB, L2-Cache: 1 MB, L3-Cache: 8 MB</td>
</tr>
<tr>
<td>Main Memory</td>
<td>16 GB DDR3 RAM, 1600 MHz</td>
</tr>
<tr>
<td>Hard Disks</td>
<td>SSD: Samsung MZ7TD256HAFV, 256 GB</td>
</tr>
<tr>
<td></td>
<td>(Read: 456.498 MB/s, Write: 256.689 MB/s)</td>
</tr>
<tr>
<td></td>
<td>HDD: Seagate ST500DM002-1BD142, 500 GB</td>
</tr>
<tr>
<td></td>
<td>(Read: 102.741 MB/s, Write: 93.648 MB/s)</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows 8 Enterprise, 64bit</td>
</tr>
</tbody>
</table>
BENCHMARK EXAMPLE

SQLITE

Table «Person» with two attributes (id and name)

Create table and add 1’000 entries with random values (CREATE, INSERT)

Read all entries and write it into a file (SELECT)

<table>
<thead>
<tr>
<th></th>
<th>CREATE, INSERT</th>
<th>SELECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>93’365 ms</td>
<td>44 ms</td>
</tr>
<tr>
<td>in-memory</td>
<td>21 ms</td>
<td>12 ms</td>
</tr>
</tbody>
</table>

INSERT x Records into Table «Person»

Benchmark SQLite

INSERT x Records into Table "Person"
BENCHMARK EXAMPLE

SQLITE

Conclusion for this example

INSERT
• Increases linearly for disk-based and in-memory
• Factor > 4000 between disk-based and in-memory

SELECT
• Increases linearly for disk-based and in-memory
• Time for disk-based varies a lot

Limitation for the CPU usage

SELECT x Records from Table «Person»

Benchmark SQLite
SELECT x Records from Table "Person"
BENCHMARK EXAMPLE [19]
MICROSOFT SQL SERVER 2014

Tool «ostress»

Execute stored procedures in parallel

Simulates different clients that are inserting orders concurrently

Parameters to configure

- Server / Authentication / Database
- Statement to be executed
- Number of connections

Database «AdventureWorks2012»

Demo.usp_DemoInsertSalesOrders

- Generates typical sales orders

@use_inmem

- Disk-based or memory-optimized tables

@order_count

- Number of orders to be inserted

@include_update

- Insert only or add followed update
BENCHMARK EXAMPLE [19]
MICROSOFT SQL SERVER 2014

«ostress» command (disk-based)

```
ostress.exe
-S -E -dAdventureWorks2012
-Q"EXEC Demo.usp_DemoInsertSalesOrders
   @use_inmem = 0,
   @order_count=100000"
-n100
```

«ostress» command (in-memory)

```
ostress.exe
-S -E -dAdventureWorks2012
-Q"EXEC Demo.usp_DemoInsertSalesOrders
   @use_inmem = 1,
   @order_count=100000"
-n100
```
BENCHMARK EXAMPLE
MICROSOFT SQL SERVER 2014

Duration for 10'000'000 order inserts (disk-based)

Benchmark Microsoft SQL Server 2014
ostress (Disk-Based)

Duration for 10'000'000 order inserts (in-memory)

Benchmark Microsoft SQL Server 2014
ostress (In-Memory)

June 6, 2014
Conclusion for this example

Factor > 50 between disk-based and in-memory

Disk-based approach is not able to fully utilize the CPU, due to latch contention

In-memory approach is able to fully utilize the CPU, but can be spiky in case of log IO bottleneck
CONCLUSION

Characteristics of In-Memory Databases

An in-memory database is primarily stored in main memory

Fast access to the data

Offers different approaches to make the database persistence

try to parallelize as much as possible to speedup the transactions and use the available hardware
Results and Improvements

Impressive performance gain
Gain depends on the database structure and architecture
Ability to run analyses and calculations more often

Recommendation

Keep the entire database in-memory if possible, especially for new applications
Minimize the footprint of the databases
Optimize the SQL queries
Future

«During the next two to three years, IMC (In-Memory Computing) will become a key element in the strategy of organizations focused on improving effectiveness and business growth.»

Mr. Pezzini, vice president and Gartner Fellow, 2013
QUESTIONS AND ANSWERS [21]
REFERENCES

REFERENCES

THANK YOU

Thank you for your attention.
public class SQLiteExample {
    public static void main(String[] args) throws ClassNotFoundException, UnsupportedEncodingException, FileNotFoundException {
        Class.forName("org.sqlite.JDBC");
        Connection connection = null;
        try {
            // create a database connection (in-memory)
            connection = DriverManager.getConnection("jdbc:sqlite::memory:");
            Statement statement = connection.createStatement();
            statement.setQueryTimeout(30); // set timeout to 30 sec.

            statement.executeUpdate("drop table if exists person");
            long start = System.currentTimeMillis();
            statement.executeUpdate("create table person (id integer, name string"));

            [...]
        }
    }
}
**BENCHMARK EXAMPLE**

**SQLITE** [23]

```java
PreparedStatement prep = connection.prepareStatement("insert into person values(?,?)");
for (int i = 0; i < 1000; i++) {
    prep.setInt(1, i);
    prep.setString(2, UUID.randomUUID().toString());
    prep.execute();
}
long end = System.currentTimeMillis();
System.out.println("used Time [CREATE, INSERT]: " + (end - start) + " ms");
```

[..]
BENCHMARK EXAMPLE

SQLITE [23]

[..]

ResultSet rs = statement.executeQuery("select id, name from person");
while (rs.next()) {
    // read the result set
    System.out.println("name = " + rs.getString("name"));
    System.out.println("id = " + rs.getInt("id"));
}
end = System.currentTimeMillis();
System.out.println("used Time [SELECT]: " + (end - start) + " ms");

[..]
BENCHMARK EXAMPLE

SQLITE \[23\]

[...]

} catch (SQLException e) {
    System.err.println(e.getMessage());
} finally {
    try {
        if (connection != null) {
            connection.close();
        }
    } catch (SQLException e) {
        System.err.println(e);
    }
}

}
BENCHMARK EXAMPLE
MICROSOFT SQL SERVER 2014

CPU

% Utilization over 60 seconds

Intel(R) Xeon(R) CPU E3-1245 v3 @ 3.40GHz

Utilization: 79%

Speed: 3.40 GHz

Processes: 92

Threads: 1759

Handles: 30065

Maximum speed: 3.40 GHz

Sockets: 1

Cores: 4

Logical processors: 8

Virtualization: Enabled

L1 cache: 256 KB

L2 cache: 1.0 MB

L3 cache: 8.0 MB

June 6, 2014
Software AG: in-memory, real-time analyses have high priority for CIOs

For nearly 50 percent of the responders in-memory is an important value in the company. More than two-thirds of CIOs complain that companies do not prioritize the introduction of the in-memory technology despite all the advantages.
REFERENCES
