Firebird meets NoSQL (Apache HBase) Case Study

Firebird Conference 2011 – Luxembourg

Thomas Steinmaurer DI
+43 7236 3343 896
thomas.steinmaurer@scch.at
www.scch.at

Michael Zwick DI (FH)
+43 7236 3343 843
michael.zwick@scch.at
www.scch.at
ATTENTION – Think BIG

Source: http://www.telovation.com/articles/gallery-old-computers.html
ATTENTION – Think BIGGER

Source: Google pictures search engine
Agenda

- Big Data Trend
- Scalability Challenges
- Apache Hadoop/HBase
- Firebird meets NoSQL – Case study
- Q&A
Big Data Trend

- Data volume grows dramatically
  - 40% up to 60% per year is common
- What is big?
  - Gigabyte, Terabyte, Petabyte ...?
  - It depends on your business and the technology you use to manage this data
- Google, Facebook, Yahoo etc. are all managing petabytes of data
- Most of the data is unstructured or semistructured, thus (far) away from our perfect relational/normalized world
Big Data Trend
The Three Vs

- Big Data is not just about data volume

VOLUME
- Terabytes
- Records
- Transactions
- Tables, files

VELOCITY
- Batch
- Near time
- Real time
- Streams

VARIETY
- Structured
- Unstructured
- Semistructured
- All the above

Source: Big Data Analytics – TDWI Best Practices Report
Big Data Trend
Typical application domains

- Sensor networks
- Social networks
- Search engines
- Health care (medical images and patient records)
- Science
- Bioinformatics
- Financial services
- Condition Monitoring systems
Agenda

- Big Data Trend
- **Scalability Challenges**
- Apache Hadoop/HBase
- Firebird meets NoSQL – Case study
How do you tackle scalability issues in your RDBMS environment?

- **Scale-Up** database server
  - Faster/more CPU, more RAM, Faster I/O
  - Create indices, optimize statements, partition data
  - Reduce network traffic
  - Pre-aggregate data
  - Denormalize data to avoid complex join statements
  - Replication for distributed workload

**Problem:** This usually fails in the Big Data area due to technical or licensing issues

**Solution:** Big Data Management demands **Scale-Out**
Scalability Challenges
A True Story (not happened to me)

- A MySQL (doesn't matter) environment
- Started with a regular setup, a powerful database server
- Business grow, data volume increased dramatically
- The project team began to:
  - Partition data on several physical disks
  - Replicate data to several machines to distribute workload
  - Denormalize data to avoid complex joins
- Removed transaction support (ISAM storage engine)
- At the end, they gave up:
  - More or less a few (or even one) denormalized table
  - Data spread across several machines and physical disks
  - Expensive and hard to administrate
- Now they are using a NoSQL solution
Scalability Challenges
How can NoSQL help

- NoSQL – Not Only SQL
- NoSQL implementations target different user bases
  - Document databases
  - Key/Value databases
  - Graph databases
- NoSQL can Scale-Out easily
- In the previous true story, they switched to a distributed key/value store implementation called HBase (NoSQL database) on top of Apache Hadoop
Agenda

- Big Data Trend
- Scalability Challenges
- Apache Hadoop/HBase
- Firebird meets NoSQL – Case study
- Q&A
Apache Hadoop is an open source software for reliable, scalable and distributed computing

Consists of

- Hadoop Common: Common utilities to support other Hadoop sub-projects
- Hadoop Distributed Filesystem (HDFS): Distributed file system
- Hadoop MapReduce: A software framework for distributed processing of large data sets on compute clusters

Can run on commodity hardware

Widely used

- Amazon/A9, Facebook, Google, IBM, Joost, Last.fm, New York, Times, PowerSet, Veoh, Yahoo! ...
Other Hadoop-related Apache projects

- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- **Mahout™**: A Scalable machine learning and data mining library.
- **Pig™**: A high-level data-flow language and execution framework for parallel computation.
- **ZooKeeper™**: A high-performance coordination service for distributed applications.
Apache Hadoop/HBase
MapReduce Framework

Source: http://www.recessframework.org/page/map-reduce-anonymous-functions-lambdas-php
Apache Hadoop/HBase
MapReduce Framework

Apache Hadoop/HBase
HBase – Overview

- Open source Java implementation of Google’s BigTable concept
- Non-relational, distributed database
- Column-oriented storage, optimized for sparse data
- Multi-dimensional
- High Availability
- High Performance
- Runs on top of Apache Hadoop Distributed Filesystem (HDFS)

Goals
- Billions of rows * Million of columns * Thousands of Versions
- Petabytes across thousands of commodity servers
Apache Hadoop/HBase
HBase – Architecture

Apache Hadoop/HBase
HBase – Data Model

- A sparse, multi-dimensional, sorted map
  - \{rowkey, column, timestamp\} -> cell
  - Column = <column_family>:<qualifier>
- Rows are **sorted lexicographically** based on the rowkey

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-------------------+
| ColumnFamily1   | (CF2)           |
| Timestamp       | qualifier1      | qualifier2      | qualifier3      | qualifier4      | ...               |
+-----------------+-----------------+-----------------+-----------------+-----------------+-------------------+
| rowkey1         | ts3             | value           | value           |                 |                   |
|                 | ts2             |                 | value           | value           |                   |
|                 | ts1             | value           |                 |                 |                   |
+-----------------+-----------------+-----------------+-----------------+-----------------+-------------------+
| rowkey2         | ts5             | value           |                 |                 |                   |
|                 | ts4             |                 | value           |                 |                   |
|                 | ts3             | value           |                 | value           |                   |
|                 | ts2             |                 |                 | value           |                   |
+-----------------+-----------------+-----------------+-----------------+-----------------+-------------------+
```
Native Java
REST
Thrift
Avro
Ruby shell
Apache MapReduce
Hive
...
Apache Hadoop/HBase
HBase – Users

- Facebook
- Twitter
- Mozilla
- Trend Micro
- Yahoo
- Adobe
- ...

© Software Competence Center Hagenberg GmbH
Apache Hadoop/HBase

HBase vs. RDBMS

- HBase is not a drop-in replacement for a RDBMS
  - No data types just byte arrays, interpretation happens at the client
  - No query engine
  - No joins
  - No transactions
  - Secondary indexes problematic
  - Denormalized data

- HBase is bullet-proof to
  - Store key/value pairs in a distributed environment
  - Scale horizontally by adding more nodes to the cluster
  - Offer a flexible schema
  - Efficiently store sparse tables (NULL doesn‘t need any space)
  - Support semi-structured and structured data
Agenda

- Big Data Trend
- Scalability Challenges
- Apache Hadoop/HBase
- Firebird meets NoSQL – Case study
- Q&A
Firebird meets NoSQL – Case study
High-Level Requirements

- Condition Monitoring System in the domain of solar collectors and inverters
- Collecting measurement values (current, voltage, temperature ...)
- Expected data volume
  - ~ 1300 billions measurement values (rows) per year
- Long-term storage solution necessary
- Web-based customer portal accessing aggregated and detailed data
- Backend data analysis, data mining, machine learning, fault detection
Initial problem: You can’t manage this data volume with a RDBMS

Measurement values are a good example for key/value pairs

Evaluated Apache Hadoop/HBase for storing detailed measurement values

Virtualized Hadoop/HBase test cluster with 12 nodes

RDBMS (Firebird/Oracle) for storing aggregated data
Firebird meets NoSQL – Case study Prototypical implementation

- HBase data model based on the web application object model
  - Row-Key: `<DataLogger-ID>~-<Device-ID>-<Timestamp>`
  - Column Families and Qualifiers based on classes and attributes in the object model
- Test data generator (TDG) and scanner (TDS) implementing the HBase data model for performance tests
  - TDG: ~70000 rows per second
  - TDS: <150ms response time with 400 simulated clients querying detail values for a particular DataLogger-ID/Device-ID for a given day for a HBase table with ~5 billions rows
MapReduce job implementation for data aggregation and storage
- Throughput: 600000 rows per second
- Pre-aggregation of ~15 billions rows in ~7h, including storage in RDBMS

Prototypical implementation based on a fraction of the expected data volume
- Simply add more nodes to the cluster to handle more data
Live Demonstration
Agenda

- Big Data Trend
- Scalability Challenges
- Apache Hadoop/HBase
- Firebird meets NoSQL – Case study
- Q&A
Questions and Answers
Thanks for your attention!

thomas.steinmaurer@scch.at
Resources