Fast Analytics on Fast Data
Kudu as Storage Layer for Banking Applications

Strata Data Conference
London 2018
Olaf Hein
Problem

Classic Credit Process

- A lot of paperwork
- Application in the subsidiary
- Costly process
- Manual approval
- Deferred disbursement
Solution

Digitised Credit Processes

- No paperwork
- Application using online banking
- Optimised processes
- Automated approval
- Immediate disbursement
## Automated Self-Disclosure

<table>
<thead>
<tr>
<th>Date</th>
<th>Posting Text</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.04.</td>
<td>ODX Inc. Salary, May 2018</td>
<td>2800</td>
</tr>
<tr>
<td>01.05.</td>
<td>GOV KG1234567890</td>
<td>200</td>
</tr>
<tr>
<td>01.05.</td>
<td>Rental Fee London Road, Redhill</td>
<td>800</td>
</tr>
<tr>
<td>13.05.</td>
<td>ACME says thank you 12.05.2018</td>
<td>45</td>
</tr>
<tr>
<td>15.05.</td>
<td>Gas &amp; Electricity GE0987654321</td>
<td>140</td>
</tr>
<tr>
<td>15.05.</td>
<td>Waterworks ABC W111222</td>
<td>60</td>
</tr>
<tr>
<td>31.05.</td>
<td>ODX Inc. Salary, May 2018</td>
<td>3000</td>
</tr>
<tr>
<td>31.05.</td>
<td>Transport for London top up</td>
<td>30</td>
</tr>
</tbody>
</table>

### Revenue

<table>
<thead>
<tr>
<th></th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary</td>
<td>2800</td>
</tr>
<tr>
<td>Child Benefit</td>
<td>200</td>
</tr>
<tr>
<td>Rental Income</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total Revenue</strong></td>
<td>3000</td>
</tr>
</tbody>
</table>

### Expenses

<table>
<thead>
<tr>
<th></th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rental</td>
<td>800</td>
</tr>
<tr>
<td>Additional Housing Costs</td>
<td>200</td>
</tr>
<tr>
<td>Other Regular Expenses</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total Expenses</strong></td>
<td>1000</td>
</tr>
</tbody>
</table>

### Summary

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Spendable Income</td>
<td>2000</td>
</tr>
<tr>
<td>Credit Limit</td>
<td>10000</td>
</tr>
</tbody>
</table>
Hadoop Storage Options (classic)

- **HBase**
  - Real-Time Updates
  - Interactive Queries
  - Real-Time Access

- **HDFS**
  - Only Inserts
  - Batch Jobs
  - Archive

**Pace of Analytics vs. Pace of Data**
- Static Data
- Frequent Updates
- Real-Time Updates
- Archive Jobs
HDFS

Distributed File System

- Data modelling
  - Large data files
  - Directories as tables
  - Subdirectories as partitions
- Several supported file formats

Characteristics

- Write once read many
- Bulk inserts
- Large table / partition scans
- No random access
- No updates and deletes
HBase

NoSQL Database

- Data modelling
  - Tables
  - Row key
  - Column families
  - Dynamic columns
- Data files
  - Stored in HDFS
  - Rows ordered by row key

Characteristics

- Random access
- Very fast inserts, updates and deletes
- Short range scans
Kudu

Storage Layer / NoSQL Database
- Data modelling
  - Tables
  - Primary key
  - Data types
  - Partitions
- Data files
  - Stored in Linux FS
  - Rows ordered by primary key
  - Columnar storage on disk

Characteristics
- Random access
- Fast inserts, updates and deletes
- Large (and short) range scans

Hadoop storage layer to enable fast analytics on fast data
Data Ingestion

Bulk Inserts

- Daily delivery of financial transactions
  - 5,000,000 median
  - 20,000,000 peak
- Delivered by batch jobs
- ETL processing
  - Cleansing
  - Conforming
  - Merging
Categorisation & Payment Flow Analysis (PFA)

Large Scans, Inserts und Updates
- Categorisation of new transactions
- 5,000,000 - 20,000,000 records
- Payment Flow Analysis for active customers
- 3,500,000 active customers

Very Large Scans & Updates
- Model changes occasionally
- Update transactions and self disclosures
- 1,500,000,000 transactions for 13 months
Online Interfaces

Random Read
- Credit limit
- Self disclosure

Random Writes
- Updated self disclosure
- Re-calculated credit limit
Delete

Random Deletes

- Delete data of particular customers
  - Transactions
  - Self disclosure
- Reasons
  - Outdated data (data economy)
  - Customer disabled credit limit option
  - Customer cancelled account
Analytics

Monitor Model
- Complete data set
- One account

Tools
- Impala SQL
- PySpark
  - Anaconda
  - Jupyter Notebook
Summary of Data Access Patterns

<table>
<thead>
<tr>
<th></th>
<th>Bulk Inserts</th>
<th>Large Scans</th>
<th>Short Scans</th>
<th>Random Reads</th>
<th>Updates</th>
<th>Deletes</th>
<th>Analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Elephant]</td>
<td>🌟</td>
<td>🌟</td>
<td>🚮</td>
<td>🚮</td>
<td>🚮</td>
<td>🚮</td>
<td>🌟</td>
</tr>
<tr>
<td>[HBase]</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
</tr>
<tr>
<td>[Kudu]</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
<td>🌟</td>
</tr>
</tbody>
</table>
High Level Architecture

- Online-Banking
- Subsidiaries
- Analytic Workplace
- REST Services
- Hadoop
- Anaconda

- RDBMS
- Linux FS
- HDFS
- HDFS
- HDFS
- Kudu
- Kudu

- Extract
- Transform
- Transform
- Load
- PFA
- Load

- Landing Zone
- Raw Zone
- Silver Zone
- Gold Zone
- PFA Data Store
- Analytical Data Store
Experiences with Kudu

Pros

- Easy data modelling
  - Partitioning
  - Key design
- Easy usage
  - Spark Connector
  - Java Client
  - Impala / SQL
- Easy operations
  - High availability
  - Stability
- Performance
  - Very versatile
  - Perfect fit for our access patterns

Cons

- Security
  - Only Kerberos authentication
  - No authorisation (so far)
- Backup
  - Not implemented (so far)
  - Workaround using Impala
    INSERT INTO parquet_table
    SELECT * FROM kudu_table;
- Operations
  - Sensitive to network problems
  - Tool set still growing
Thank you