Towards a Comprehensive End-to-End Benchmark for Big Data
Tilmann Rabl - bankmark UG (haftungsbeschränkt)
02/04/2015 @ SPEC RG Big Data
The BigBench Proposal

• End to end benchmark
  • Application level
• Based on a product retailer (TPC-DS)
• Focused on Parallel DBMS and MR engines
• History
  • Launched at 1st WBDB, San Jose
  • Published at SIGMOD 2013
  • Spec at WBDB proceedings 2012 (queries & data set)
  • Full kit at WBDB 2014
• Collaboration with Industry & Academia
  • First: Teradata, University of Toronto, Oracle, InfoSizing
  • Now: bankmark, CLDS, Cisco, Cloudera, Hortonworks, Infosizing, Intel, Microsoft, Oracle, Pivotal, SAP, IBM, UoFT, ...
Before BigBench

• **Micro-Benchmarks**
  • System level measurement
  • Illustrative not informative

• **Functional Benchmarks**
  • Better than micro-benchmarks
  • Simplified approach – limited representation in e2e

• **Benchmark suites**
  • Collection of micro and functional
  • Standardization problems
Specs and Standards

- **TPC xHS**
  - Very first Industry standard
  - Detailed metrics and run rules
  - Model framework

- **TPC-DS BigData**
  - Derive as-is from TPC-DS
  - Query based for SQL on Hadoop

- **BigBench**
  - Based on new specification with some reuse
  - Complex batch analytics
  - Long term bridge
BigBench - 2013

- **Collaborative industry effort**
  - Sigmod 2013
  - Address 3V’s of big data
  - Very first concept for a big data benchmark specification
  - Wide industry support

- **Use case sampling**
  - Retail use case example
  - End to end and component

- **Framework Agnostic**
  - Well defined specification
  - SW based reference implementation
Derived from TPC-DS

- Multiple snowflake schemas with shared dimensions
- 24 tables with an average of 18 columns
- 99 distinct SQL ‘99 queries with random substitutions
- Representative skewed database content
- Sub-linear scaling of non-fact tables
- Ad-hoc, reporting, iterative and extraction queries
- ETL-like data maintenance
BigBench Data Model

- Structured: TPC-DS + market prices
- Semi-structured: website click-stream
- Unstructured: customers’ reviews
Data Model – 3 Vs

- **Variety**
  - Different schema parts

- **Volume**
  - Based on scale factor
  - Similar to TPC-DS scaling, but continuous
  - Weblogs & product reviews also scaled

- **Velocity**
  - Refresh for all data with different velocities
Scaling

- **Continuous scaling model**
  - Realistic
- **SF 1 ~ 1 GB**
- **Different scaling speeds**
  - Adapted from TPC-DS
    - Static
    - Square root
    - Logarithmic

\[ LF = SF + (SF - (\log_5(SF) \times \sqrt{SF})) = 2SF - \log_5(SF) \times \sqrt{SF} \]
Generating Big Data

• **Repeatable computation**
  • Based on XORSHIFT random number generators

• **Hierarchical seeding strategy**
  • Enables independent generation of every value in the data set
  • Enables independent re-generation of every value for references

• **User specifies**
  • Schema – data model
  • Format – CSV, SQL statements, ...
  • Distribution – multi-core, multi-node, partially

• **PDGF generates**
  • High quality data – distributed, in parallel, in the correct format
  • Large data – terabytes, petabytes
Workload

• **Workload Queries**
  • 30 “queries”
  • Specified in English (sort of)
  • No required syntax (first implementation in Aster SQL MR)
  • Kit implemented in Hive, HadoopMR, Mahout, OpenNLP

• **Business functions (adapted from McKinsey report)**
  • **Marketing**
    • Cross-selling, customer micro-segmentation, sentiment analysis, enhancing multichannel consumer experiences
  • **Merchandising**
    • Assortment optimization, pricing optimization
  • **Operations**
    • Performance transparency, product return analysis
  • **Supply chain**
    • Inventory management
  • **Reporting (customers and products)**
Towards a Comprehensive End-to-End Benchmark for Big Data - Tilmann Rabl

2/4/2015

Workload - Technical Aspects

Generic Characteristics

<table>
<thead>
<tr>
<th>Data Sources</th>
<th>#Queries</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured</td>
<td>18</td>
<td>60%</td>
</tr>
<tr>
<td>Semi-structured</td>
<td>7</td>
<td>23%</td>
</tr>
<tr>
<td>Un-structured</td>
<td>5</td>
<td>17%</td>
</tr>
</tbody>
</table>

Hive Implementation Characteristics

<table>
<thead>
<tr>
<th>Query Types</th>
<th>#Queries</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure HiveQL</td>
<td>14</td>
<td>46%</td>
</tr>
<tr>
<td>Mahout</td>
<td>5</td>
<td>17%</td>
</tr>
<tr>
<td>OpenNLP</td>
<td>5</td>
<td>17%</td>
</tr>
<tr>
<td>Custom MR</td>
<td>6</td>
<td>20%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th>Input Datatype</th>
<th>Processing Model</th>
<th>Query</th>
<th>Input Datatype</th>
<th>Processing Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Structured</td>
<td>Java MR</td>
<td>#16</td>
<td>Structured</td>
<td>Java MR (OpenNLP)</td>
</tr>
<tr>
<td>#2</td>
<td>Semi-Structured</td>
<td>Java MR</td>
<td>#17</td>
<td>Structured</td>
<td>HiveQL</td>
</tr>
<tr>
<td>#3</td>
<td>Semi-Structured</td>
<td>Python Streaming MR</td>
<td>#18</td>
<td>Unstructured</td>
<td>Java MR (OpenNLP)</td>
</tr>
<tr>
<td>#4</td>
<td>Semi-Structured</td>
<td>Python Streaming MR</td>
<td>#19</td>
<td>Structured</td>
<td>Java MR (OpenNLP)</td>
</tr>
<tr>
<td>#5</td>
<td>Semi-Structured</td>
<td>HiveQL</td>
<td>#20</td>
<td>Structured</td>
<td>Java MR (Mahout)</td>
</tr>
<tr>
<td>#6</td>
<td>Structured</td>
<td>HiveQL</td>
<td>#21</td>
<td>Structured</td>
<td>HiveQL</td>
</tr>
<tr>
<td>#7</td>
<td>Structured</td>
<td>HiveQL</td>
<td>#22</td>
<td>Structured</td>
<td>HiveQL</td>
</tr>
<tr>
<td>#8</td>
<td>Semi-Structured</td>
<td>HiveQL</td>
<td>#23</td>
<td>Structured</td>
<td>HiveQL</td>
</tr>
<tr>
<td>#9</td>
<td>Structured</td>
<td>HiveQL</td>
<td>#24</td>
<td>Structured</td>
<td>HiveQL</td>
</tr>
<tr>
<td>#10</td>
<td>Unstructured</td>
<td>Java MR (OpenNLP)</td>
<td>#25</td>
<td>Structured</td>
<td>Java MR (Mahout)</td>
</tr>
<tr>
<td>#11</td>
<td>Unstructured</td>
<td>HiveQL</td>
<td>#26</td>
<td>Structured</td>
<td>Java MR (Mahout)</td>
</tr>
<tr>
<td>#12</td>
<td>Semi-Structured</td>
<td>HiveQL</td>
<td>#27</td>
<td>Unstructured</td>
<td>Java MR (OpenNLP)</td>
</tr>
<tr>
<td>#13</td>
<td>Structured</td>
<td>HiveQL</td>
<td>#28</td>
<td>Unstructured</td>
<td>Java MR (Mahout)</td>
</tr>
<tr>
<td>#14</td>
<td>Structured</td>
<td>HiveQL</td>
<td>#29</td>
<td>Structured</td>
<td>Python Streaming MR</td>
</tr>
<tr>
<td>#15</td>
<td>Structured</td>
<td>Java MR (Mahout)</td>
<td>#30</td>
<td>Semi-Structured</td>
<td>Python Streaming MR</td>
</tr>
</tbody>
</table>
SQL-MR Query 1

```
SELECT category_cd1 AS category1_cd, 
    category_cd2 AS category2_cd, COUNT(*) AS cnt 
FROM basket_generator ( 
    ON 
    ( SELECT i.i_category_id AS category_cd, 
        s.ws_bill_customer_sk AS customer_id 
    FROM web_sales s INNER JOIN item i 
    ON s.ws_item_sk = i.item_sk ) 
    PARTITION BY customer_id 
    BASKET_ITEM ('category_cd') 
    ITEM_SET_MAX (500) 
) 
GROUP BY 1,2 
ORDER BY 1, 3, 2;
```
HiveQL Query 1

```sql
SELECT pid1, pid2, COUNT (*) AS cnt
FROM (
    FROM ( 
        SELECT s.ss_ticket_number AS oid, s.ss_item_sk AS pid
        FROM store_sales s
        INNER JOIN item i ON s.ss_item_sk = i.i_item_sk
        WHERE i.i_category_id in (1, 2, 3) and s.ss_store_sk in (10, 20, 33, 40, 50)
    ) q01_temp_join
    MAP q01_temp_join.oid, q01_temp_join.pid
    USING 'cat'
    AS oid, pid
    CLUSTER BY oid
    ) q01_map_output
    REDUCE q01_map_output.oid, q01_map_output.pid
    USING 'java -cp bigbenchqueriesmr.jar:de.bankmark.bigbench.queries.q01.Red'
    AS (pid1 BIGINT, pid2 BIGINT)
) q01_temp_basket
GROUP BY pid1, pid2
HAVING COUNT (pid1) > 49
ORDER BY pid1, cnt, pid2;
```
Benchmark Process

Adapted to batch systems
- No trickle update

Measured processes
- Loading
- Power Test (single user run)
- Throughput Test I (multi user run)
- Data Maintenance
- Throughput Test II (multi user run)

Result
- Additive Metric
Metric

• Throughput metric
  • BigBench queries per hour

• Number of queries run
  • $30 \times (2 \times S + 1)$

• Measured times
  - $T_L$: Execution time of the loading process;
  - $T_P$: Execution time of the power test;
  - $T_{TT1}$: Execution time of the first throughput test;
  - $T_{DM}$: Execution time of the data maintenance task.
  - $T_{TT2}$: Execution time of the second throughput test.

- Metric

$$BBQpH = \frac{30 \times 3 \times 3600}{T_L + T_P + \frac{T_{TT1}}{S} + T_{DM} + \frac{T_{TT2}}{S}}$$

$$BBQpH = \frac{30 \times 3 \times S \times 3600}{S \times T_L + S \times T_P + T_{TT1} + S \times T_{DM} + T_{TT2}}$$

Source: www.wikipedia.de
BigBench Experiments

• Tests on
  • Cloudera CDH 5.0, Pivotal GPHD-3.0.1.0, IBM InfoSphere BigInsights

• In progress: Spark, Impala, Stinger, ...

• 3 Clusters (+)
  • 1 node: 2x Xeon E5-2450 0 @ 2.10GHz, 64GB RAM, 2 x 2TB HDD
  • 6 nodes: 2 x Xeon E5-2680 v2 @ 2.80GHz, 128GB RAM, 12 x 2TB HDD
  • 546 nodes: 2 x Xeon X5670 @ 2.93GHz, 48GB RAM, 12 x 2TB HDD
BigBench reference implementation - 2014

• Hadoop Map-Reduce and Hive
  • Hadoop Map-Reduce 2.0
  • HIVE, Mahout
  • Java 1.7

• Reference Kit Queries
  • All 30 queries are implemented.
  • Represents Structured, Semi-Structured, Un-Structured data types.

• Complete runnable kit
  • Data generator, queries, benchmark driver
  • Tested on various Hadoop implementation
  • Easy to configure and run, detailed setup instructions
  • https://github.com/intel-hadoop/Big-Bench

• Bring some time
  • Full BigBench run on Hive takes 2 days+
  • Will verify if your cluster is setup correctly
BigBench 2015+ - Towards a Big Data Pipeline

- **Extensive Procedural Coverage**
  - Large input Datasets
  - Web-based procedural
  - Functional component level

- **Machine Learning**
  - Dedicated machine learning workloads
  - Algorithm coverage
  - Segmented

- **Others**
  - Key-value pair representation
  - Cover graph
  - Streaming applications
  - Multimedia
BigBench 2015 – Metrics

Metric
• Current metric based on geometric mean
• Represent performance / $ - scaling factor
• Segmented
  • Structural
  • Unstructured/procedural
  • Machine learning
  • Other components
  • All components = E2E
• Machine learning accuracy
• Fault tolerant performance
Contact

Tilmann Rabl
tilmann.rabl@bankmark.de
www.bankmark.de