Accelerating TPCx-BigBench on SQL-on-Hadoop*

Yi Zhou
Intel SSG/STO/Big Data Technology
Contact us: sto-bigdata-qa-prc@intel.com
About US

We're Intel, We're BDT

- Optimize big data on IA thru leveraging partners like Cloudera
- Lead Hadoop in open source community
- Link industry innovations for complete IA experience
Agenda

- Big Data Benchmarks
- What's Inside BigBench
- Tuning Hive on Spark with Big-Bench
- Tuning Spark SQL with Big-Bench
- Scaling Experience of Big-Bench
- Reference
Big Data Benchmarks

Micro Benchmarks

End-to-End Benchmark (BigBench)

Synthetic, Illustrative benchmarks suitable for Regression testing

• Real-world, Instructive benchmarks suitable for End-to-End benchmark
• Open source standards based
• Industry consortium proposed
• Support from 10+ ecosystem partners

TPCx-BigBench entered TPC public review phase in Nov. 2015
http://www.tpc.org/tpcx-bb/default.asp
What’s Inside BigBench
BigBench Retail Business Functions (Mckinsey Report)

- **Retail**
  - Cross-selling
  - Customer micro-segmentation
  - Sentiment analysis
  - Enhancing multichannel consumer experiences

- **Retail**
  - Assortment optimization
  - Pricing optimization

- **Retail**
  - Performance transparency
  - Product return analysis

- **Retail**
  - Inventory management

- **Retail**
  - Customers and Products

### Departments

- **Marketing** (~60%)
- **Merchandising** (~17%)
- **Operations** (~13%)
- **Supply Chain** (~7%)
- **Reporting** (~3%)
BigBench Data Model

- **Variety**
  - Structured: TPC-DS + market prices
  - Semi-structured: website click-stream
  - Unstructured: customers' reviews

- **Volume**
  - Based on scale factor(1TB/3TB/10TB/100TB)
  - Similar to TPC-DS scaling, but continuous
  - Weblogs & product reviews also scaled

- **Velocity**
  - Refresh for all data with different velocities
  - Different velocity for different data type

- Derived from TPC-DS: star schema with fact tables, representing store sales, and online sales channels
- Additional big data specific dimensions to analyze user behavior
# BigBench Workload Characteristics

<table>
<thead>
<tr>
<th>Data Sources</th>
<th>Number of Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured</td>
<td>18</td>
</tr>
<tr>
<td>Semi-structured</td>
<td>7</td>
</tr>
<tr>
<td>Un-structured</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analytic techniques</th>
<th>Number of Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics analysis</td>
<td>6</td>
</tr>
<tr>
<td>Data mining</td>
<td>17</td>
</tr>
<tr>
<td>Reporting</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Types</th>
<th>Number of Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure HiveQL</td>
<td>14</td>
</tr>
<tr>
<td>Mahout (Spark MLlib)</td>
<td>5</td>
</tr>
<tr>
<td>OpenNLP</td>
<td>5</td>
</tr>
<tr>
<td>Custom MR</td>
<td>6</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 (Open NLP)</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 (Open NLP)</td>
</tr>
<tr>
<td>#4</td>
<td>Semi-structured</td>
<td>Python Streaming MR</td>
<td>#19</td>
<td>Structured</td>
<td>Java MR (Open NLP)</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 (Open NLP)</td>
<td>#25</td>
<td>Structured</td>
<td>Java MR (Mahout)</td>
</tr>
<tr>
<td>#11</td>
<td>Unstructured</td>
<td>Java MR (Open NLP)</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 (Open NLP)</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>
BigBench Benchmark Process

- **Data Generation**

- **Benchmark Phase**
  - Loading Test Phase
  - Power Test Phase
  - Throughput Test I Phase (Multi-user)
  - Data Maintenance (Refresh Data)
  - Throughput Test II Phase (Multi-user)

- **Benchmark Result**
  - Evaluate the performance of Big Data system
BigBench SQL-on-Hadoop Engines

- Extend the mainstream Hadoop engines: Spark SQL, Hive on Spark, Impala, Hive on Tez...
- Query optimization
- Query level engine setting
Tuning Hive on Spark with Big-Bench
Tuning Hive on Spark with Big-Bench
- Improve Resource Utilization by Spark 1.5 Dynamic Allocation

Problem
CPU usage low in multi-user throughput for Hive on Spark engine

Solution
- Set “spark.dynamicAllocation.enabled” to true
- Set “spark.shuffle.service.enabled” to true

Software and Services
- Preserve all the shuffle files written by executors

System Technologies and Optimization
Tuning Hive on Spark with Big-Bench
- Improve Resource Utilization by Spark 1.5 Dynamic Allocation

Performance Improvement

![CPU/Memory Utilization](chart1)

**CPU/Memory Utilization**
- Enable Dynamic Allocation

![Throughput Test Score Comparison](chart2)

**Throughput Test Score Comparison**

- Enable Dynamic Allocation
- Disable Dynamic Allocation
Tuning Hive on Spark with Big-Bench
- Improve Resource Utilization by Spark 1.5 Dynamic Allocation

Why It Matters

Static Allocation

Resource Scheduling Problems
• Underutilized of cluster resources
• Starvation of other applications
• Lack of elastic resource scaling ability

Dynamic Allocation

• Number of executors are decided by workloads in run-time
• More efficient utilization of cluster resources
Tuning Spark SQL with Big-Bench
Tuning Spark SQL with BigBench - JOIN

**Problem**
- Most workloads in Big-Bench have JOIN & Left Semi Join operation - Join a Large table (Fact Table) with a Small table (Dimension Table)
- ShuffledHashJoin
  - Shuffle (Disk IO, Network etc.)
  - Uneven sharding
  - Limited Parallelism

```
SELECT *
FROM inventory inv
JOIN (  
    SELECT  
    i_item_id,  
    i_item_sk  
    FROM item  
    WHERE i_current_price > ${hiveconf:Q22_i_current_price_min}  
    AND i_current_price < ${hiveconf:Q22_i_current_price_max}  
) items  
ON inv.inv_item_sk = items.i_item_sk  
JOIN warehouse w ON inv.inv_warehouse_sk = w.w_warehouse_sk  
JOIN date_dim d ON inv.inv_date_sk = d.d_date_sk  
WHERE datediff(d_date, '${hiveconf:Q22_date}') > -30  
AND datediff(d_date, '${hiveconf:Q22_date}') < 30  
) q22_coalition_22
```
Tuning Spark SQL with BigBench - JOIN(Cont.)

• **Solution**
  - BroadcastHashJoin (AKA Map Join) – broadcast the small RDD to all worker nodes.
  - Tune "spark.sql.autoBroadcastJoinThreshold" to enable the broadcast Join

![Diagram showing broadcast of RDDs and improved query time with broadcast hash join]

<table>
<thead>
<tr>
<th>Query Time (s)</th>
<th>(Lower is Better)</th>
</tr>
</thead>
<tbody>
<tr>
<td>270</td>
<td>x 4.22 speedup</td>
</tr>
<tr>
<td>64</td>
<td></td>
</tr>
</tbody>
</table>

Big-Bench Query 22
(scale factor=1000)

- Shuffled Hash Join
- Broadcast Hash Join

```
spark.sql.autoBroadcastJoinThreshold=209715200;
```
### Detecting
- Tasks take long time to complete.
- Some tasks OOM
- Lost spark executors

### Solution
- Tune “spark.sql.shuffle.partition”
- Too small partition number may cause OOM
- Too large partition number may cause performance degradation.

![Query Time (s) (Lower is Better)](chart)

- **X 7.53 Speedup**
- 1785
- 237

**Big-Bench Query 18 (scale factor=1000)**
- 200 Shuffle Partition Number
- 10 Shuffle Partition Number
Scaling Experience of Big-Bench
Our Scaling Experience of Big-Bench  
- Data Scaling Experiments

Data Scale: 1TB, 3TB, 10TB

Engine: Hive on MapReduce
Our Scaling Experience of Big-Bench
- Data Scaling Experiments

Enable Large Data Scale (30TB) for Hive on MapReduce

• Reduce Stage
  
  \[\text{hive.exec.reducers.bytes.per.reducer} \]
  
  \[\text{hive.exec.reducers.max} \]

• DataNode Throughput
  
  \[\text{dfs.datanode.handler.count} \]
  
  \[\text{dfs.namenode.handler.count} \]
  
  \[\text{dfs.namenode.service.handler.count} \]

  at least \(\ln(\text{number of datanodes}) \times 20\)
Our Scaling Experience of Big-Bench
- Cluster Nodes Scaling Experiments

Cluster Nodes Scale: 6 nodes -> 10 nodes  = 66.7%

Data Scale: 3TB, 10TB

Engine: Hive on MapReduce

![Benchmark Execution Time(s)](benchmark_chart.png)
Our Scaling Experience of Big-Bench
- Stream Experiments

Data Scale: 3TB

Engine: Hive on MapReduce
Customer Cases with using Big-Bench

- **Top 2 online e-commerce**
  Leverage Big-Bench with their business supports to predict technical trends

- **Top 3 CSP**
  Use Big-Bench to help evaluate their system resource utilization
References

• BigBench Kit:
  - https://github.com/intel-hadoop/Big-Data-Benchmark-for-Big-Bench.git

• BigBench Google Groups:
  - https://groups.google.com/forum/#!forum/big-bench

• TPCx-BigBench Home
Notices and Disclaimers

● No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
● Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
● This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.
● The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.
● Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.
● Intel, the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.
● Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

*Other names and brands may be claimed as the property of others

© 2016 Intel Corporation.
Software and Services
System Technologies and Optimization