- Whose to Choose

Distribution Differences Between

[Images of Cloudera, Hortonworks, and MapR logos]

David Teplow

DTeplow@IntegraTC.com
Your Presenter

- Database architect / developer since the very first release of the very first RDBMS (Oracle v2; 1981)
- Helped launch the Northeast OUG in 1983
- Founded Database Technologies in 1986
- NOUG President from 1992 – 1999
- Founded Integra Technology Consulting in 2000
- Served on the IOUG board from 2003 – 2008
- Focused on Hadoop / Big Data since 2012
The Three Contenders

- CDH (Cloudera Distribution of Apache Hadoop)
- HDP (Hortonworks Data Platform)
- MapR
Differences between CDH, HDP and MapR

1. The Companies behind them
2. Management/Administration Tools
3. SQL-on-Hadoop Offerings
4. Performance Benchmarks
## Company Profiles

<table>
<thead>
<tr>
<th></th>
<th>Cloudera</th>
<th>Hortonworks</th>
<th>MapR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Year Founded</strong></td>
<td>2009</td>
<td>2011</td>
<td>2009</td>
</tr>
<tr>
<td><strong>$ Raised</strong></td>
<td>$1,041M</td>
<td>$358M</td>
<td>$174M</td>
</tr>
<tr>
<td><strong>Valuation</strong></td>
<td>$4.1B</td>
<td>$1.1B</td>
<td>?</td>
</tr>
<tr>
<td><strong>Annual Revenues</strong></td>
<td>$100M</td>
<td>$46M</td>
<td>$42M</td>
</tr>
<tr>
<td><strong>Price / Sales</strong></td>
<td>41X</td>
<td>24X</td>
<td>?</td>
</tr>
<tr>
<td><strong>Hadoop PMC Members</strong></td>
<td>12</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td><strong>Hadoop Committers</strong></td>
<td>14</td>
<td>28</td>
<td>0</td>
</tr>
</tbody>
</table>
Poll #1 – Which Company Do You Like?

A. Cloudera
B. Hortonworks
C. MapR Technologies
Management / Administration Tools

- Cloudera ➔ Cloudera Manager
- Hortonworks ➔ Ambari
- MapR ➔ MapR Control System (MCS)
Cloudera Manager Advanced Features
(available only with Cloudera Enterprise)

- **Quota Management** for setting/tracking user and group-based quotas/usage
- **Configuration History / Rollbacks** for tracking all actions and configuration changes, with the ability to roll back to previous states
- **Rolling Updates** for staging service updates and restarts to portions of the cluster sequentially to minimize downtime during upgrades/updates
- **AD Kerberos and LDAP/SAML Integration**
- **SNMP Support** for sending Hadoop-specific events/alerts to monitoring tools as SNMP traps
- **Scheduled Diagnostics** for sending a snapshot of the cluster state to Cloudera support for optimization and issue resolution
- **Automated Backup and Disaster Recovery** for configuring/managing snapshotting and replication workflows for HDFS, Hive and HBase
Ambari Advanced Features (always free and enabled)

- **Configuration versioning and history** provides visibility, auditing and coordinated control over configuration changes, and management of all services and components deployed on your Hadoop Cluster (rollback will be supported in the next release of Ambari)

- **Views Framework** provides plug-in UI capabilities for custom visualization, management and monitoring features in the Ambari Web console, for example Tez View, which gives you visibility into all the jobs on your cluster, allowing you to quickly identify which jobs consume the most resources and which are the best candidates to optimize

- **Blueprints** provide declarative definitions of a cluster, which allows you to specify a Stack, the Component layout and the configurations to materialize a Hadoop cluster instance (via a REST API) without the need for any user interaction
MCS Advanced Features
(available only with MapR Enterprise Edition)

- **Advanced Multi-Tenancy** with control over job placement and data placement

- **Consistent Point-In-Time Snapshots** for hot backups and to recover data from deletions or corruptions due to user or application error

- **Disaster Recovery** through remote replicas created with block level, differential mirroring with multiple topology configurations
Poll #2 – Which Management / Administration Tool Do You Like?

A. Cloudera Manager
B. Apache Ambari
C. MapR Control System
SQL-on-Hadoop

- Cloudera ➔ Impala
- Hortonworks ➔ Hive
- MapR ➔ Drill
Impala

- Based on Dremel, a real-time, distributed query and analysis technology from Google
- Impala accesses data in HDFS or HBase directly through a specialized distributed query engine (instead of batch MapReduce access)
- Uses the same metadata that Hive uses
- Query results are streamed whenever they’re available rather than all at once upon query completion
Impala

- Impala is not fault-tolerant (queries must be restarted if a node fails) and, therefore, may not be suitable for long-running queries.
- Impala requires the working set of a query to fit into the aggregate physical memory of the cluster it’s running on and, therefore, may not be suitable for multi-terabyte datasets (Impala 2.0 / CDH 5.2 introduced a “Spill to Disk” option that may avoid this).
- User-Defined Functions (UDFs) must be written in Java or C++.
- Claims to be open source but is really invented / developed by Cloudera.
Hive

- Hive is a declarative abstraction layer (HiveQL) that uses metadata from the Hive metastore and a processing framework in the background (originally MapReduce but now other options exist)

- **Open source** project under the Apache Software Foundation (ASF)
  - Originally developed at Facebook; became a subproject of Hadoop in 2008; graduated to top-level status in September 2010
  - 40 Committers from 14 Organizations (17 from Hortonworks, 5 from Facebook, 4 from Cloudera)

- As part of the “Stinger Initiative”, Tez (introduced in Hive 0.13 / HDP 2.1) provides performance improvements for Hive by assembling many tasks into a single MR job rather than many using Directed Acyclic Graphs (DAG)

- From Hive 0.10 (released in January 2013) to Hive 0.13 (April 2014), performance improved an average of 52X on 50 TPC-DS Queries
  - Total time to run all 50 queries decreased from 187.2 hours to 9.3 hours
**Hive**

- Hive 0.14 (released in November 2014 / comes with HDP 2.2) has support for INSERT, UPDATE and DELETE statements via **ACID** transactions.
- Hive 0.14 includes a Cost-Based Optimizer (CBO) called **Calcite (f.k.a. Optiq)**, another open source project under ASF.
- Writes intermediate results to disk (unlike Impala, which streams data between stages of a query, or Spark SQL, which holds data in memory).
Drill

- Like Impala, Drill is also based on Google’s Dremel
- **Open source** project under the Apache Software Foundation (ASF)
  - Started in August 2012 as an incubator project but graduated to top-level status in December 2014
  - 17 Committers from 4 Organizations (13 from MapR, 1 from Hortonworks)
- Uses the same metadata that Hive and Impala use (but doesn’t *need* metadata)
- Schema can be discovered on the fly (as opposed to RDBMS schema on write or Hive/Impala schema on read) by taking advantage of self-describing data (i.e. XML, JSON, BSON, Avro, Parquet, etc.)
Spark / Spark SQL (f.k.a. “Shark”)

- Another implementation of the DAG approach (like Tez)
- Spark adds Resilient Distributed Datasets (RDDs), an abstraction that makes it easy to work with distributed data \textit{in memory}
- \textit{Open source} project under the Apache Software Foundation (ASF)
  - Originally developed at the UC Berkeley AMPLab; became an incubator project in June 2013; graduated to top-level status in February 2014
  - 35 Committers from 14 Organizations (12 from Databricks, 7 from UC Berkley, 4 from Yahoo!)
Spark / Spark SQL (f.k.a. “Shark”)

- CDH 5.3, HDP 2.2 and MapR 4.1 all include Apache Spark 1.2 (MapR 4.1 also includes Impala 1.4.1)
- Most major tool vendors have native Spark SQL connectors, including MicroStrategy, Pentaho, QlikView, Tableau, Talend
- In addition to HDFS, Spark can run against HBase, MongoDB, Cassandra, JSON, and Text Files
- Not only provides database access (with Spark SQL), but also has built-in libraries for continuous data processing (with Spark Streaming), machine learning (with MLlib), and graphical analytics (with GraphX)
Poll #3 – Which SQL-on-Hadoop Do You Like?

A. Impala
B. Hive
C. Drill
D. Spark SQL
Performance Benchmarks

- In August 2014, the Transaction Processing Performance Council (www.tpc.org) announced the TPC Express Benchmark HS (TPCx-HS) “to provide an objective measure of hardware, operating system and commercial Apache Hadoop File System API compatible software distributions, and to provide the industry with verifiable performance, price-performance and availability metrics.”

- Simply stated, the TPCx-HS benchmark measures the time it takes a Hadoop cluster to load and sort a given dataset, which can have a Scale Factor (SF) of 1TB, 3TB, 10TB, 30TB, 100TB, 300TB, 1000TB, 3000TB or 10000TB.

- The workload consists of the following modules:
  - HSGen - generates the data at a particular Scale Factor (based on TeraGen)
  - HSDataCheck - checks the compliance of the dataset and replication
  - HSSort - sorts the data into a total order (based on TeraSort)
  - HSVValidate - validates the output is sorted (based on TeraValidate)
TPC\textsubscript{x}-HS Results

- The first TPC\textsubscript{x}-HS result was published by MapR (w/Cisco) in January 2015. Running MapR M5 Edition 4.0.1 on RHEL 6.4 on a 16-node cluster, their results for sort throughput (higher is better) and price-performance (lower is better) were:
  - 5.07 HSph and $121,231.76/HSph @1TB Scale Factor
  - 5.10 HSph and $120,518.63/HSph @3TB Scale Factor
  - 5.77 HSph and $106,524.27/HSph @10TB Scale Factor

- Cloudera (w/Dell) responded in March 2015. Running CDH 5.3.0 on Suse SLES 11 SP3 on a 32-node cluster, their results were:
  - 19.15 HSph and $48,426.85/HSph @30TB Scale Factor (no virtualization)
  - 20.76 HSph and $49,110.55/HSph @30TB Scale Factor (w/virtualization)

- We shall see if and how Hortonworks responds
TPC-H-like Benchmark

- In August 2014, three IBM Researchers published a paper in *Proceedings of the VLDB Endowment* (Volume 7, No. 12) that compares Impala to Hive. They used the 22 queries specified in the TPC-H Benchmark but left out the refresh streams. They ran this workload against a 1TB database / Scale Factor of 1,000 (a 3TB database / Scale Factor of 3,000 would have exceeded Impala’s limitation that requires the workload’s working set to fit in the cluster’s aggregate memory).

**Results:** Compared to Hive on MapReduce:
- Impala is on average 3.3X faster with compression
- Impala is on average 3.6X faster without compression

Compared to Hive on Tez:
- Impala is on average 2.1X faster with compression
- Impala is on average 2.3X faster without compression
## Hive vs. Impala Execution

<table>
<thead>
<tr>
<th>TPC-H Query</th>
<th>Hive-MR ORC (secs)</th>
<th>Hive-MR ORC Snappy (secs)</th>
<th>Hive-Tez ORC (secs)</th>
<th>Hive-Tez ORC Snappy (secs)</th>
<th>Impala Parquet (secs)</th>
<th>Impala Parquet Snappy (secs)</th>
<th>Impala w/Snappy improvement over Hive-Tez w/Snappy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>266</td>
<td>228</td>
<td>154</td>
<td>172</td>
<td>25</td>
<td>24</td>
<td>7.2X</td>
</tr>
<tr>
<td>Q2</td>
<td>299</td>
<td>294</td>
<td>132</td>
<td>111</td>
<td>29</td>
<td>29</td>
<td>3.8X</td>
</tr>
<tr>
<td>Q3</td>
<td>543</td>
<td>462</td>
<td>286</td>
<td>280</td>
<td>122</td>
<td>115</td>
<td>2.4X</td>
</tr>
<tr>
<td>Q4</td>
<td>277</td>
<td>232</td>
<td>220</td>
<td>214</td>
<td>135</td>
<td>147</td>
<td>1.5X</td>
</tr>
<tr>
<td>Q5</td>
<td>833</td>
<td>772</td>
<td>415</td>
<td>409</td>
<td>189</td>
<td>190</td>
<td>2.2X</td>
</tr>
<tr>
<td>Q6</td>
<td>109</td>
<td>85</td>
<td>88</td>
<td>81</td>
<td>6</td>
<td>6</td>
<td>13.5X</td>
</tr>
<tr>
<td>Q7</td>
<td>1096</td>
<td>1038</td>
<td>597</td>
<td>589</td>
<td>179</td>
<td>182</td>
<td>3.2X</td>
</tr>
<tr>
<td>Q8</td>
<td>822</td>
<td>770</td>
<td>392</td>
<td>369</td>
<td>140</td>
<td>148</td>
<td>2.5X</td>
</tr>
<tr>
<td>Q9</td>
<td>2883</td>
<td>2848</td>
<td>1682</td>
<td>1692</td>
<td>FAILED</td>
<td>FAILED</td>
<td>–</td>
</tr>
<tr>
<td>Q10</td>
<td>526</td>
<td>502</td>
<td>236</td>
<td>224</td>
<td>59</td>
<td>50</td>
<td>4.5X</td>
</tr>
<tr>
<td>Q11</td>
<td>234</td>
<td>216</td>
<td>141</td>
<td>134</td>
<td>22</td>
<td>22</td>
<td>6.1X</td>
</tr>
<tr>
<td>Q12</td>
<td>308</td>
<td>241</td>
<td>186</td>
<td>184</td>
<td>38</td>
<td>40</td>
<td>4.6X</td>
</tr>
<tr>
<td>Q13</td>
<td>266</td>
<td>228</td>
<td>155</td>
<td>156</td>
<td>123</td>
<td>127</td>
<td>1.2X</td>
</tr>
<tr>
<td>Q14</td>
<td>175</td>
<td>155</td>
<td>123</td>
<td>120</td>
<td>13</td>
<td>14</td>
<td>8.6X</td>
</tr>
<tr>
<td>Q15</td>
<td>272</td>
<td>233</td>
<td>159</td>
<td>156</td>
<td>21</td>
<td>22</td>
<td>7.1X</td>
</tr>
<tr>
<td>Q16</td>
<td>235</td>
<td>239</td>
<td>143</td>
<td>133</td>
<td>44</td>
<td>43</td>
<td>3.1X</td>
</tr>
<tr>
<td>Q17</td>
<td>779</td>
<td>813</td>
<td>945</td>
<td>724</td>
<td>385</td>
<td>395</td>
<td>1.8X</td>
</tr>
<tr>
<td>Q18</td>
<td>925</td>
<td>859</td>
<td>723</td>
<td>672</td>
<td>185</td>
<td>189</td>
<td>3.6X</td>
</tr>
<tr>
<td>Q19</td>
<td>677</td>
<td>656</td>
<td>734</td>
<td>569</td>
<td>703</td>
<td>710</td>
<td>0.8X</td>
</tr>
<tr>
<td>Q20</td>
<td>379</td>
<td>332</td>
<td>179</td>
<td>175</td>
<td>195</td>
<td>191</td>
<td>0.9X</td>
</tr>
<tr>
<td>Q21</td>
<td>1605</td>
<td>1495</td>
<td>FAILED</td>
<td>FAILED</td>
<td>854</td>
<td>857</td>
<td>–</td>
</tr>
<tr>
<td>Q22</td>
<td>385</td>
<td>452</td>
<td>132</td>
<td>130</td>
<td>29</td>
<td>30</td>
<td>4.3X</td>
</tr>
</tbody>
</table>
TPC-DS-like Benchmark

- Next, they used a subset of the TPC-DS Benchmark, consisting of 20 queries that access a single fact table (STORE_SALES) and 6 dimension tables.
  Note: the full TPC-DS benchmark involves 99 queries that access 7 fact tables and 17 dimension tables.

- They ran this workload against a 3TB database / Scale Factor of 3,000 and found:
  - Impala is on average 8.2X faster than Hive on MapReduce
  - Impala is on average 4.3X faster than Hive on Tez
TPC-DS-like Benchmark #2

- Cloudera published a benchmark in September 2014 after the release of Impala 1.4.0 that compares the performance of Impala versus Hive on Tez versus Spark SQL using a subset of the TPC-DS Benchmark, consisting of 8 “Interactive” queries, 6 “Reporting” queries, and 5 “Analytics” queries, for a total of 19 queries that access a single fact table (STORE_SALES) and 9 dimension tables. They ran this workload against a 15TB database / Scale Factor of 15,000, which is not one of the Scale Factors allowed by TPC.

Results: With a single-user workload that runs the 19 queries:
- Impala is on average 7.8X faster than Hive on Tez
- Impala is on average 3.3X faster than Spark SQL

With 10 concurrent users running just the 8 Interactive queries:
- Impala is on average 18.3X faster than Hive on Tez
- Impala is on average 10.6X faster than Spark SQL
Benchmark #3

- Another Big Data Benchmark was performed by the UC Berkley AMPLab in February 2014. It compares the performance of Impala 1.2.3 versus Hive 0.12 on MapReduce versus Hive 0.12 on Tez 0.2.0 versus Spark SQL 0.8.1 on Amazon EC2 clusters with small, medium and large datasets.

- Based on the paper “A Comparison of Approaches to Large-Scale Data Analysis” by Pavlo et al. (from Brown University, M.I.T., etc.), it uses 3 tables with data from CommonCrawl.org, which contains petabytes of data collected over 7 years of web crawling.
Benchmark #3

- Tables:
  
  CREATE TABLE Documents ( 
  url VARCHAR(100) 
  PRIMARY KEY, 
  contents TEXT );

  CREATE TABLE Rankings ( 
  pageURL VARCHAR(100) 
  PRIMARY KEY, 
  pageRank INT, 
  avgDuration INT );

  CREATE TABLE UserVisits ( 
  sourceIP VARCHAR(16), 
  destURL VARCHAR(100), 
  visitDate DATE, 
  adRevenue FLOAT, 
  userAgent VARCHAR(64), 
  countryCode VARCHAR(3), 
  languageCode VARCHAR(6), 
  searchWord VARCHAR(32), 
  duration INT );

- Queries:

1. SELECT pageURL, pageRank FROM Rankings WHERE pageRank > X;

2. SELECT SUBSTR(sourceIP, 1, X), SUM(adRevenue) FROM UserVisits GROUP BY SUBSTR(sourceIP, 1, X);

3. SELECT sourceIP, avgPageRank, totalRevenue FROM (SELECT sourceIP, AVG(pageRank) as avgPageRank, SUM(adRevenue) as totalRevenue FROM Rankings as R, UserVisits as UV WHERE R.pageURL = UV.destURL AND UV.visitDate BETWEEN Date('1980-01-01') AND Date('X') GROUP BY UV.sourceIP) ORDER BY totalRevenue DESC LIMIT 1;
Benchmark #3 Results

### Median Response Times

<table>
<thead>
<tr>
<th>Query 1 (Scan)</th>
<th>Small (32,888 results)</th>
<th>Medium (3.3M results)</th>
<th>Large (90M results)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impala</td>
<td>12.0</td>
<td>12.0</td>
<td>37.1</td>
</tr>
<tr>
<td>Hive on MR</td>
<td>50.5</td>
<td>59.9</td>
<td>43.3</td>
</tr>
<tr>
<td>Hive on Tez</td>
<td>28.2</td>
<td>36.4</td>
<td>26.4</td>
</tr>
<tr>
<td>Spark SQL</td>
<td>6.6</td>
<td>7.0</td>
<td>22.4</td>
</tr>
</tbody>
</table>

### Median Response Times

<table>
<thead>
<tr>
<th>Query 2 (Aggregation)</th>
<th>Small (2M groups)</th>
<th>Medium (31M groups)</th>
<th>Large (254M groups)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impala</td>
<td>113.7</td>
<td>155.3</td>
<td>277.5</td>
</tr>
<tr>
<td>Hive on MR</td>
<td>730.6</td>
<td>765.0</td>
<td>833.3</td>
</tr>
<tr>
<td>Hive on Tez</td>
<td>377.5</td>
<td>438.0</td>
<td>427.6</td>
</tr>
<tr>
<td>Spark SQL</td>
<td>151.4</td>
<td>164.3</td>
<td>196.5</td>
</tr>
</tbody>
</table>

### Median Response Times

<table>
<thead>
<tr>
<th>Query 3 (Join)</th>
<th>Small (485,312 rows)</th>
<th>Medium (53M rows)</th>
<th>Large (533M rows)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impala</td>
<td>108.7</td>
<td>129.8</td>
<td>431.3</td>
</tr>
<tr>
<td>Hive on MR</td>
<td>561.1</td>
<td>717.6</td>
<td>2374.2</td>
</tr>
<tr>
<td>Hive on Tez</td>
<td>323.1</td>
<td>402.3</td>
<td>1361.9</td>
</tr>
<tr>
<td>Spark SQL</td>
<td>111.7</td>
<td>135.6</td>
<td>382.6</td>
</tr>
</tbody>
</table>
Poll #4 – Which Performance Benchmark is Most Relevant?

A. TPCx-HS (Raw Sort)
B. TPC-H & TPC-DS (Supplier DB & DW)
C. 3 Tables (from CommonCrawl.org) / 3 Queries (scan, aggregation, join)
SQL-on-Hadoop Conclusions

- **Impala** is faster than **Hive**.

- **Impala** is sometimes faster than **Spark SQL** but **Spark SQL** is faster than **Impala** with large datasets and also with simple range scans regardless of number of rows queried.

- **Hive** supports a fuller set of SQL commands, including INSERT, UPDATE and DELETE with ACID compliance. It also supports windowing functions and rollup (not yet supported by **Impala**), **UDFs written in any language**, and provides fault tolerance. **Hive** also has a new Cost-Based Optimizer (Calcite) that should deliver performance improvements and guard against poorly written queries degrading performance.

- **Drill** holds an advantage over **Impala**, **Hive** and **Spark SQL** in its ability to discover a datafile’s schema on the fly without needing the Hive metastore or other metadata.
Other Conclusions

- **Hortonworks** is the most active contributor within the Hadoop open source community; **Cloudera** next most active; and **MapR** is least active.

- **Hortonworks** is also the leader in terms of keeping all of its software “in the open”, including:
  - Management / Administration tool (Ambari)
  - SQL-on-Hadoop offering (Hive)

- **MapR** has made proprietary extensions to one of the core Hadoop projects, HDFS (Hadoop Distributed File System).
Final Poll – Which Distribution Do You Like?

A. CDH
B. HDP
C. MapR
Thank You

Q & A

I welcome your further questions or comments:
DTeplow@IntegraTC.com