Your users don’t care where data is stored or processing happens: Hadoop or Data Warehouse

And now they don’t have to.
Introducing Teradata QueryGrid.
Teradata’s View
Big Data and Data in General

DATA enables INSIGHTS which drive ACTIONS to provide BUSINESS ADVANTAGE thru BETTER DECISIONS

Big Data is DATA. It “adds to” the depth and types of INSIGHTS ... But the rest of the story is the SAME.

However,
• the economics have changed, increasing the amount of data you can capture
• the tools have been enhanced, expanding the data types you can analyze
• The architecture framework has evolved so you can use the right tool for the right job
Analysts Recommend:
Shift from a Single Platform to an Ecosystem

"Logical" Data Warehouse

- Content & XML
- Complex Data types
- Cloud DBMS
- Data in HDFS
- Column-Store DBMS
- In-Memory DBMS
- CEP

Hybrid Ecosystem

- One Node: 27%
- Two Nodes: 26%
- Three Nodes: 6%
- Four Nodes: 8%
- Five Nodes: 2%
- Six Nodes: 2%
- Seven Nodes: 2%
- Eight Nodes: 1%

"We will abandon the old models based on the desire to implement for high-value analytic applications."

"Big Data requirements are solved by a range of platforms including analytical databases, discovery platforms, and NoSQL solutions beyond Hadoop."

Fabric Based Computing

Optimized for BI

• The backbone of UDA
  > High performance infrastructure
  > Aggregate Teradata IDW, Aster Discovery and Hadoop
  > Industry approach optimized for Big Analytics use

• Key Teradata Elements
  > BYNET V5 on InfiniBand interconnect
  > Teradata Managed Servers
  > System management across all of the FBC
Teradata QueryGrid™

Optimize, simplify, and orchestrate processing across and beyond the Teradata UDA

• Run the right analytic on the right platform
  > Take advantage of specialized processing engines while operating as a cohesive analytic environment

• Automated and optimized work distribution through “push-down” processing across platforms
  > Minimize data movement, process data where it resides
  > Minimize data duplication
  > Transparently automate analytic processing and data movement between systems
  > Bi-directional data movement

• Integrated processing; within and outside the UDA

• Easy access to data and analytics through existing SQL skills and tools
When fully implemented, the Teradata Database or the Teradata Aster Database will be able to intelligently use the functionality and data of multiple heterogeneous processing engines.
# Quickly Building Out the QueryGrid Family

<table>
<thead>
<tr>
<th>Connector</th>
<th>One-Way</th>
<th>Bi-Directional</th>
<th>Push-Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata – Hortonworks Hadoop</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Teradata – Cloudera Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Aster - Teradata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aster 6 - Hortonworks Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Aster 6 - Cloudera Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
</tbody>
</table>
Leverage Hadoop resources, Reduce data movement

- Bi-directional to Hadoop
- Query push-down
- Easy configuration of server connections

- Query through Teradata
- Sent to Hadoop through Hive
- Results returned to Teradata
- Additional processing joins data in Teradata
- Final results sent back to application/user
Teradata QueryGrid Teradata-Hadoop

• QueryGrid
  > Server grammar
    – Simplify via “server name”
  > Hadoop import operator
    – Load_from_hcatalog
    – Added server grammar
  > Hadoop export operator (new)
    – Load_to_hcatalog
    – Supports files:
      • Delimited Text, JSON, RCFile
      • Sequence File, ORCfile, Avro

• Query push-down
• Bi-directional data transfer
• Provide access rights
CREATE FOREIGN SERVER Hadoop_sysd USING
HOSTTYPE('hadoop')
SERVER ('sysd.labs.teradata.com')
PORT ('9083')
HIVESERVER ('sysd.labs.teradata.com')
HIVEPORT ('10000')
USERNAME('Hive')
DEFAULT_STRING_SIZE('2048')
HADOOP_PROPERTIES('org.apache.hadoop.io.compress.GzipCodec');
DO IMPORT WITH syslib.load_from_hcatalog_hdp1_3_2,
DO EXPORT WITH syslib.load_to_hcatalog_hdp1_3_2
  Merge_hdfs_files('True')
  Compression_codec('org.apache.hadoop.io.compress.GzipCodec');

SELECT source, session FROM
  clickstream@Hadoop_sysd
  WHERE session_ts = '2013-12-01';
QueryGrid Server Objects and Privileges

• TD_SERVER_DB contains all servers objects

• Servers are global objects

• Users have SELECT and INSERT granted to them
  > GRANT SELECT ON hdp132 TO tom;
  > GRANT INSERT ON hdp143 TO lisa;

• Being able to create and drop a server is a privilege
  > GRANT CREATE SERVER ...
  > GRANT DROP SERVER...
Remote SQL Execution

- Push SQL to remote Hive system
- Hive filters data on non-partitioned columns
- Foreign table ‘Select’ executed on remote system

```sql
SELECT source, session FROM FOREIGN TABLE(select session, source from clickstream where source = "Mozilla")@Hadoop_sysd WHERE session = current_date AS dt;
```
QueryGrid Data Transfer

**Import**

```
SELECT source, session FROM clickstream@Hadoop_sysd WHERE session_ts = '2013-01-01';
```

“insert/select” & “create table as” to instantiate data locally.

**Export**

```
INSERT INTO cust_loc@Hadoop_sysd
SELECT cust_id, cust_zip FROM cust_data
WHERE last_update = current_date;
```
EXPLAIN INSERT INTO  cardata@hdp132 SELECT * FROM newcars;

***Success: Activity Count = 41
Explanation

1) First, we lock a distinct ut1."pseudo table" for read on a RowHash to prevent global deadlock for ut1.tab1.

2) Next, we lock ut1.tab1 for read.

3) We do an all-AMPs RETRIEVE step from ut1.newcars by way of an all-rows scan with no residual conditions executing table operator SYSLIB.load_to_hcatalog with a condition of ("(1=1)") into Spool 2 (used to materialize view, derived table, table function or table operator drvtab_inner) (all_amps), which is built locally on the AMPs. The size of Spool 2 is estimated with low confidence to be 8 rows (11,104 bytes). The estimated time for this step is 0.16 seconds.
QueryGrid Explained Part 2

4) We do an all-AMPs RETRIEVE step from Spool 2 (Last Use) by way of an all-rows scan into Spool 3 (used to materialize view, derived table, table function or table operator TblOpInputSpool) (all_amps), which is redistributed by hash code to all AMPs. The size of Spool 3 is estimated with low confidence to be 8 rows (11,104 bytes). The estimated time for this step is 0.16 seconds.

5) We do an all-AMPs RETRIEVE step from Spool 3 (Last Use) by way of an all-rows scan executing table operator SYSLIB.load_to_hcatalog with a condition of "(1=1)" into Spool 4 (used to materialize view, derived table, table function or table operator h4) (all_amps), which is built locally on the AMPs.

< BEGIN EXPLAIN FOR REMOTE QUERY -->
TD: 3 column(s); Hadoop: 3 column(s), with 2 partition column(s);
doors(INTEGER) -> doors(STRING); make(VARCHAR) -> make*(STRING);
model(VARCHAR) -> model*(STRING); * denotes partition column;
<--- END EXPLAIN FOR REMOTE QUERY >

The size of Spool 4 is estimated with low confidence to be 8 rows (200 bytes). The estimated time for this step is 0.16 seconds.
Create and Drop Hadoop Tables

- Stored procedures to create and drop Hadoop tables
- Allows SQL scripts to export data in stand alone fashion

CALL SYSLIB.HDROP('t3','hdp132');

CALL SYSLIB.HCTAS('t3','c2,c3','LOCATION "/user/hive/table_t12"','hdp132','default');
Requirements for QueryGrid to Hadoop

• Teradata 15.0
  > Node memory > 96GB

• Network
  > All Teradata nodes able to connect to all Hadoop data nodes

• Proxy user on Hadoop
Connection Flow

- The client connects to the system through the PE in node 1. The query is parsed in the PE. During the parsing phase, the table operator’s contract function contacts the HCatalog component through the External Access Handler (EAH), which is a one-per-node Java® Virtual Machine.
- The HCatalog returns the metadata about the table, the number of columns, and the types for the columns. The parser uses this info and also uses this connection to obtain the Hadoop splits of data that underlie the Hadoop table.
- The splits are assigned to the AMPs in a round-robin fashion so that each AMP gets a split.
- The parser phase completes and produces an AMP step containing the table operator. This is sent to all the AMPs in parallel.
- Each AMP then begins to execute the table operator’s execute function providing a parallel import of Hadoop data.
- The execute function opens and reads the split data reading in Hadoop rows. These are converted to Teradata data types in each column, and the rows are written to spool.
- When all the data has been written, the spool file is redistributed as input into the next part of the query plan.
# Quickly Building Out the QueryGrid Family

<table>
<thead>
<tr>
<th>Connector</th>
<th>One-Way</th>
<th>Bi-Directional</th>
<th>Push-Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata – Hortonworks Hadoop</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Teradata – Cloudera Hadoop</td>
<td>August</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Teradata – Oracle (11g)</td>
<td>✓</td>
<td>3Q 2014</td>
<td>3Q 2014</td>
</tr>
<tr>
<td>Teradata - MongoDB</td>
<td>1Q2015</td>
<td>1Q2015</td>
<td>1Q2015</td>
</tr>
<tr>
<td>Teradata – DB2</td>
<td>1H2015</td>
<td>1H2015</td>
<td>1H2015</td>
</tr>
<tr>
<td>Aster - Teradata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aster 6 - Hortonworks Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Aster 6 - Cloudera Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
</tbody>
</table>
Unity Source Link -- High Level Description

- USL enables dynamic SQL access to foreign database (currently Oracle only) from Teradata.
- The first phase 14.10 was released with Import capability only.
  - Import is selecting data from a foreign database
  - Export is inserting data into a foreign database
- This release includes
  - Export capability
  - Dynamic database links for both Import and Export via SQL-H Foreign Server grammar
  - Single sign-on across local and remote databases.
## Quickly Building Out the QueryGrid Family

<table>
<thead>
<tr>
<th>Connector</th>
<th>One-Way</th>
<th>Bi-Directional</th>
<th>Push-Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata – Hortonworks Hadoop</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Teradata – Cloudera Hadoop</td>
<td>August</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Teradata – Oracle (11g)</td>
<td>✓</td>
<td>3Q 2014</td>
<td>3Q 2014</td>
</tr>
<tr>
<td>Teradata - Teradata</td>
<td>Oct</td>
<td>Oct</td>
<td>Oct</td>
</tr>
<tr>
<td>Teradata - MongoDB</td>
<td>1Q2015</td>
<td>1Q2015</td>
<td>1Q2015</td>
</tr>
<tr>
<td>Teradata – DB2</td>
<td>1H2015</td>
<td>1H2015</td>
<td>1H2015</td>
</tr>
<tr>
<td>Aster - Teradata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aster 6 - Hortonworks Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Aster 6 - Cloudera Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
</tbody>
</table>
Teradata-to-Teradata: Overview

- Provides SQLs to read from or write to a remote Teradata Database
- The feature will be released asynchronously
- Teradata-to-Teradata is supported when both local & remote boxes are at least on Teradata 15.00
- When 15.10 is released, user can use the connector across releases 15.00 -> 15.10 or 15.10 -> 15.00,
  - NOTE if there is data type mismatch it may lead to error
- Both the local and remote system should have the connector software installed
Bi-directional
Operate with either IB or Ethernet connection of any speeds
Supports Teradata 15.0 on SUSE Linux 10 and 11
Parallel import & export
Buffered mode Vs. Row mode
Remote execution (Pushdown processing, Predicate pushdown)
Rights to execute Select, Insert
Trusted user
Partial data type support. 15.0 supports basic data types. On roadmap is support for LOB and JSON
<table>
<thead>
<tr>
<th>Feature</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to create links to remote Teradata system</td>
<td>CREATE SERVER object</td>
</tr>
<tr>
<td>Ability to query on another Teradata system using Server object</td>
<td>SELECT statement</td>
</tr>
<tr>
<td>Push execution on to remote system</td>
<td>SELECT FOREIGN TABLE statement</td>
</tr>
<tr>
<td>Push or insert data to remote system</td>
<td>INSERT statement</td>
</tr>
<tr>
<td>Ability to see data definitions of objects on remote system</td>
<td>HELP FOREIGN [SERVER</td>
</tr>
<tr>
<td>Ability to execute SQL on the remote system from the local</td>
<td>Execute Foreign SQL</td>
</tr>
<tr>
<td>Ability to stich execution plan, from local and remote system</td>
<td>EXPLAIN on INSERT / SELECT statements</td>
</tr>
</tbody>
</table>
Teradata-to-Teradata: Create Server Object

• Create Server Statement:

```sql
CREATE AUTHORIZATION sdll7151proxy AS
   DEFINER TRUSTED USER 'proxyuser' PASSWORD 'secret';

CREATE FOREIGN SERVER sdll7151
EXTERNAL SECURITY DEFINER TRUSTED sdll7151proxy USING
   Hosttype('Teradata')
   remotehost ('sdll7151.labs.teradata.com')
   local_ips('sdlc0062.labs.teradata.com')
   port('5000')
   read_timeout(200)
   listen_timeout(60)
   concurrentstreams(1)
   DO IMPORT WITH syslib.load_from_td,
   DO EXPORT WITH syslib.load_to_td;
```

• TD_SERVER_DB contains all servers objects.
  > Servers are global objects.
Teradata-to-Teradata: Query

• Provides capability to access remote Teradata data easily using the server grammar
• Import examples:

```sql
SELECT * FROM store.pricelist@sdll7151 WHERE part_price<2.00;
SELECT * FROM store.pricelist@sdll7151 UNION SELECT * FROM newitems;
```

• Query can be part of a larger query
• User can limit the columns and rows imported in order to control the amount of data imported and speed the import process.
• Returned rows can be joined with local tables
• Import can be used in joins and any place a normal table is referenced including views, macros, and stored procedures.
Teradata-to-Teradata: Import using Foreign table syntax

```sql
SELECT * FROM FOREIGN TABLE(
    SELECT p.partno, SUM(quantity * part_price) FROM
    store.pricelist p, store.inventory i
    GROUP BY p.partno WHERE p.partno=i.partno
)@sdl7151 as dt;
```

- Select FOREIGN TABLE provides a grammar pass through mechanism.
- Grammar in the parenthesis is unchecked but tokenized and then passed to the remote for execution.
- Limitations:
  - ON Clause sub-query
  - Order By not supported in Foreign Table Select. Ordering would be lost during the data transfer.
EXPLAIN SELECT * FROM store.pricelist@sdll7151 WHERE part_price<2.00;

1) First, we do an all-AMPs RETRIEVE step executing table ope
syslib.load_from_td with a condition of ("pricelist.PART_P
2.00").

< BEGIN EXPLAIN FOR REMOTE QUERY -->
1) First, we lock a distinct store."pseudo table" for read
RowHash to prevent global deadlock for store.pricelist.
2) Next, we lock store.pricelist for read. 3) We do an all
RETRIEVE step from store.pricelist by way of an all-rows s
a condition of ("store.pricelist.part_price < 2.00") into
(group_amps), which is built locally on the AMPs. The size
Spool 1 is estimated with no confidence to be 2 rows (202
The estimated time for this step is0.07 seconds. -> The co
of Spool 1 are flushed to disk and sent back to the user a
result of statement 1. The total estimated time is 0.07 se
<--- END EXPLAIN FOR REMOTE QUERY >
The size of Spool 1 is estimated with low confidence to be
(272 bytes). The estimated time for this step is 0.06 sec
SELECT p.partno, SUM(quantity * part_price) FROM store.pricelists@sdll7151 p, inventory i WHERE p.partno=i.partno;

< BEGIN EXPLAIN FOR REMOTE QUERY -->

2) Next, we lock store.pricelist for read. 3) We do an all-AMPS RETRIEVE step from store.pricelist by way of an all-rows scan with no residual conditions into Spool 1 (group_amps), which is built locally on the AMPS. The size of Spool 1 is estimated with low confidence to be 4 rows (132 bytes). The estimated time for this step is 0.07 seconds. -> The contents of Spool 1 are flushed to disk and sent back to the user as the result of statement 1. The total estimated time is 0.07 seconds.

<-- END EXPLAIN FOR REMOTE QUERY >

The size of Spool 1 is estimated with low confidence to be 4 rows (132 bytes). The estimated time for this step is 0.06 seconds.

4) We do an all-AMPS RETRIEVE step from Spool 1 (Last Use) by way of an all-rows scan with a condition of ("NOT (p.PARTNO IS NULL)") into Spool 5 (all_amps), which is redistributed by hash code to all AMPS. Then we do a SORT to order Spool 5 by row hash. The size of Spool 5 is estimated with low confidence to be 4 rows (100 bytes). The estimated time for this step is 0.03 seconds.

5) We do an all-AMPS JOIN step from UT1.i by way of a RowHash match scan, which is joined to Spool 5 (Last Use) by way of a RowHash match scan. UT1.i and Spool 5 are joined using a merge join, with a join condition of ("PARTNO = UT1.i.partno"). The result goes into Spool 4 (all_amps), which is built locally on the AMPS.
Teradata-to-Teradata: Export

- Use INSERT statement to export data to remote system
- Supports Insert-Select from local table syntax:
  
  ```sql
  INSERT INTO store.pricelist@sdll7151 SELECT * FROM newitems;
  ```

- Also, supports inserting specific named columns into the remote table as well as constant data
  
  ```sql
  INSERT INTO store.pricelist@sdll7151 VALUES (1223315, 'UP Flat Car', 3.99);
  ```

- Table may be empty or an existing table with data. Data is appended to the existing table’s data
EXPLAIN INSERT INTO store.pricelist@sdl7151
SELECT * FROM newitems;

3) We do an all-AMPs RETRIEVE step from UT1.newitems by way of an all-rows scan with no residual conditions executing table operator syslib.load_to_td with a condition of "(1=1)" into Spool 2 (used to materialize view, derived table, table function or table operator pricelist) (all_amps), which is built locally on the AMPs.

< BEGIN EXPLAIN FOR REMOTE QUERY -->
1) First, we lock a distinct store."pseudo table" for write on a RowHash to prevent global deadlock for store.pricelist.
2) Next, we lock store.pricelist for write. 3) We do an all-AMPs RETRIEVE step executing table operator SYSLIB.load_from_tdRemote with a condition of "(1=1)". The size of Spool 1 is estimated with low confidence to be 4 rows (272 bytes). The estimated time for this step is 0.06 seconds. 4) We do an all-AMPs RETRIEVE step from Spool 1 (Last Use) by way of an all-rows scan into Spool 2 (all_amps), which is redistributed by hash code to all AMPs. Then we do a SORT to order Spool 2 by row hash. The size of Spool 2 is estimated with low confidence to be 4 rows (240 bytes). The estimated time for this step is 0.03 seconds. 5) We do an all-AMPs MERGE into store.pricelist from Spool 2 (Last Use).

<-- END EXPLAIN FOR REMOTE QUERY >
Teradata-to-Teradata: Monitoring data movement

- Bytestransferred in and out are logged in DBQL
  - Show size of data transfer

```
Select Bytestransferred from dbc.dbqlsteptbl;
```

- Query Monitor portlet in Viewpoint can show data imported and exported per query
## Quickly Building Out the QueryGrid Family

<table>
<thead>
<tr>
<th>Connector</th>
<th>One-Way</th>
<th>Bi-Directional</th>
<th>Push-Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata – Hortonworks Hadoop</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Teradata – Cloudera Hadoop</td>
<td>August</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Teradata – Oracle (11g)</td>
<td>✓</td>
<td>3Q 2014</td>
<td>3Q 2014</td>
</tr>
<tr>
<td>Teradata - Teradata</td>
<td>Oct</td>
<td>Oct</td>
<td>Oct</td>
</tr>
<tr>
<td>Teradata - Aster</td>
<td>4Q2014</td>
<td>4Q2014</td>
<td>4Q2014</td>
</tr>
<tr>
<td>Teradata - MongoDB</td>
<td>1Q2015</td>
<td>1Q2015</td>
<td>1Q2015</td>
</tr>
<tr>
<td>Teradata – DB2</td>
<td>1H2015</td>
<td>1H2015</td>
<td>1H2015</td>
</tr>
<tr>
<td>Aster - Teradata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aster 6 - Hortonworks Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
<tr>
<td>Aster 6 - Cloudera Hadoop</td>
<td>✓</td>
<td>Planned</td>
<td>Planned</td>
</tr>
</tbody>
</table>

Even more connectors being planned (Plans change though of course!)
BI Tools Run SQL-MR Through Teradata
Teradata Unity and Teradata QueryGrid

- Teradata Differentiators
- Work together to provide seamless user experience
- Simplify multi-system analytics

**Teradata Unity**

**Multi-System Management**
- Route queries
- Synchronize data
- UDA ecosystem mgmt

**Teradata QueryGrid**

**Orchestrate processing across and beyond the Teradata UDA**
- Right analytic, right platform
- Push-down processing
- Bi-directional data movement
Your users don’t care where data is stored or processing happens: Hadoop or Data Warehouse.

And now they don’t have to. Introducing Teradata QueryGrid.
QUESTIONS?