Getting Jiggy with Change Data Capture and Slowly Changing Dimensions on Hadoop

Venkat Ankam
Hadoop Architect
CenturyLink
CenturyLink: Proven Strength and Stability

- **A Top 3** communications provider
- **Top 150** of Fortune 500
- **$18.7B** in annual pro-forma revenues
- **33,000+** business and government customers
- **Highest** investor returns in industry for 15 years
- **48,000+** employees worldwide
- **$3.045B** in free cash flow
- **98%** of the Fortune 500 choose CenturyLink
IT Systems and Strategic Consulting: Who We Are

Team of 100+ strategy consultants with deep experience in broad-ranging IT disciplines featuring:

✓ Proven experience in advising on and shaping diverse IT initiatives within CenturyLink and commercial clients

✓ Direct access to subject matter experts in various IT disciplines

✓ Skilled in discovery and assessment for any IT discipline

✓ One-stop-shop for all systems and strategy consulting needs

✓ Experienced in managing complex and rigorous consulting engagements
How we Build and Improve IT for Your Applications

- **Solutions**
- **Network & Telco**
- **Hosting**
- **Cloud**

**IT**

- **IT Systems & Strategic Planning**
- **Application Development & Maintenance**
- **Infrastructure Management**
- **Quality Assurance & Testing**
## Presentation Outline

<table>
<thead>
<tr>
<th>Topic</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional Data Warehouse Problems</td>
<td></td>
</tr>
<tr>
<td>Drivers for Data Warehouse on Hadoop</td>
<td></td>
</tr>
<tr>
<td>Technology and Integrations</td>
<td></td>
</tr>
<tr>
<td>Change Data Capture &amp; Slowly Changing Dimensions</td>
<td></td>
</tr>
<tr>
<td>Change Data Capture using Sqoop and Hive</td>
<td></td>
</tr>
<tr>
<td>Slowly Changing Dimensions on Hive</td>
<td></td>
</tr>
<tr>
<td>Tools and Future</td>
<td></td>
</tr>
<tr>
<td>Q&amp;A</td>
<td></td>
</tr>
</tbody>
</table>
Traditional Data Warehouse Problems

- Low-value data that consumes warehouse space: Over time, many warehouses have become bloated with both raw data staged for preprocessing and rarely accessed data that provides little business value.

- Inadequate data for business demands: Because of capacity and performance constraints, some warehouses contain only summary data, not the granular and detailed information that the business needs.

- In-warehouse transformations that impair performance: Running data transformations within a warehouse on data staged for preprocessing (i.e., ELT) consumes valuable CPUs, hindering query performance and further diminishing a warehouse’s business value.

- Limitations in multi-structured data and schema flexibility: Warehouses based on relational databases are not built to handle the multi-structured datatypes from new big data sources, while schema changes can trigger disruptions and delays.

- Network Performance: Cannot scale well with commodity hardware and parallel processing and limitations amount of data processed in a reasonable time.

- We want to use SQL and SQL-Like languages, but we don’t want RDBMS Constraints

- The Disruptive solution is Hadoop
Drivers for Data Warehouse on Hadoop

➢ Strategic:
  o Open flood gates to new types of data
  o New kinds of analysis impossible on RDBMS
  o Schema-on-read for exploratory BI
    • Attack same data from multiple perspectives
    • Choose SQL and Non-SQL approaches
    • Accelerate data driven innovation
  o Keep granular data in “Active Archive” forever
  o Simultaneous incompatible analysis modes on same data
  o Enterprise Data Hub: One location for all data
Drivers for Data Warehouse on Hadoop

➢ Tactic:
  o Dramatically Lowered operational costs
  o Linear Scaling across response time, concurrency and data size well beyond peta bytes
  o Highly reliable write once, redundantly stored data
  o Meet ETL SLAs
How does Hadoop fits in Data Warehouse?

No Hadoop

Adjacent System

Archive/Tiered Solution

ETL Engine

Staging/Offload

Tiered DWH

Replace

Query Engine

Replace
Technology and Integrations for Building a Hadoop Data Warehouse

**Sqoop** for Importing Data  
**Hive** for ETL Processes  
**Impala** for end-user interactive processing  
**Oozie** for Workflow automation

**HBase** already supports Inserts, updates and deletes? So, Why not HBase?

- HBase doesn’t support consistency at any level other than a single row update. A transaction model to support distributed transactions would need to be added.
- Many Hive users don’t have HBase installed and requiring it imposes a significant operational overhead.
- HBase is designed to support low latency point lookups rather than high throughput scans over large ranges of the table.
- HBase has a single sort key that is partitioned by range and doesn’t support Hive’s richer partition, sort order, and bucketing strategies. This limits the ability of the user to layout the table for various varieties of queries.
- **But, some use cases might need HBASE** – Think of impossibly wide schemas, Billions of rows and millions of columns with versioning.
Most important tools are well integrated with Hadoop

- ETL Tool Suites: Informatica, Data Stage, Oracle etc
- BI Tools: Qlikview, WebFocus, Talend, SAP BO, Pentaho, Tableau, Excel etc
- Job Schedulers: Control-M, Autosys etc

Looks good. But, what are the challenges?

- Skill Challenges – Yes, needs some learning curve
- Biggest challenge – HDFS is a WORM (Write Once and Read Many Times) and cannot be updated. How to achieve change data capture and maintain slowly changing dimensions?
What is Change Data Capture (CDC) and Slowly Changing Dimensions (SCD)?

- Ability to detect the changed data in source systems and capture these changes is called as Change Data Capture (CDC).
- Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse.
  - **Type 0**: New Records only; Discard Changes & Deletes
  - **Type 1**: Overwrite
  - **Type 2**: Expire and Create New Record
  - **Type 3**: Overwrite special “Current” Columns
  - **Type 4**: Maintain data in current and history tables
  - **Type 5**: Add Mini-Dimension and Type 1 Outrigger
  - **Type 6**: This combines 1+2+3
  - **Type 7**: Dual Type 1 and Type 2 Dimensions
Why CDC and SCD are needed?

- Captures all changes
- Maintains history
- Low impact on source system
- Low cost and better performance – since capturing changes only
Techniques of CDC

- Full comparison of source and target
- Comparing source and target by using CRC (Cyclic Redundancy Checksum)
- Using transaction files
- Using commercial CDC applications
- Using DBMS’s CDC feature
- Using triggers
- Using DBMS replication
- Using an indicator of update on source records – Then Incremental import those changes. Can you trust this?
Let’s consider a simple use case...

### Source Table

<table>
<thead>
<tr>
<th>id</th>
<th>fname</th>
<th>lname</th>
<th>dob</th>
<th>mstatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>135001</td>
<td>john</td>
<td>Doe</td>
<td>1960-01-12 00:00:00</td>
<td>married</td>
</tr>
<tr>
<td>135002</td>
<td>John</td>
<td>Doe Jr</td>
<td>1985-12-31 00:00:00</td>
<td>single</td>
</tr>
<tr>
<td>135003</td>
<td>John Q</td>
<td>Public</td>
<td>1985-05-06 00:00:00</td>
<td>divorced</td>
</tr>
<tr>
<td>135004</td>
<td>Judy</td>
<td>Doe</td>
<td>1980-10-22 00:00:00</td>
<td>single</td>
</tr>
<tr>
<td>135005</td>
<td>Jane</td>
<td>Roe</td>
<td>1982-08-16 00:00:00</td>
<td>married</td>
</tr>
</tbody>
</table>

### Changes in Source Table

<table>
<thead>
<tr>
<th>id</th>
<th>fname</th>
<th>lname</th>
<th>dob</th>
<th>mstatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>135001</td>
<td>john</td>
<td>Doe</td>
<td>1960-01-12 00:00:00</td>
<td>married</td>
</tr>
<tr>
<td>135002</td>
<td>John Jr</td>
<td>Doe Jr</td>
<td>1985-12-31 00:00:00</td>
<td>single</td>
</tr>
<tr>
<td>135003</td>
<td>John Q</td>
<td>Public</td>
<td>1985-05-06 00:00:00</td>
<td>divorced</td>
</tr>
<tr>
<td>135004</td>
<td>Judy</td>
<td>Doe</td>
<td>1980-10-22 00:00:00</td>
<td>single</td>
</tr>
<tr>
<td>135005</td>
<td>Jane</td>
<td>Roe</td>
<td>1982-08-16 00:00:00</td>
<td>married</td>
</tr>
<tr>
<td>135006</td>
<td>Joe</td>
<td>Bloggs</td>
<td>1970-07-14 00:00:00</td>
<td>married</td>
</tr>
</tbody>
</table>

<< 135002 Deleted
<< Updated
<< Inserted
CDC Using Sqoop

- **Incremental Import – Import only new rows**
  
  ```
  sqoop import --connect jdbc:oracle:thin:@xxxxx.com:1521/dbname --table tablename  
  --username xxx --password xxx \
  --check-column xxx \  
  --incremental append \  
  --last-value xxxxxxxx
  ```

- **Incremental Import – Import all updated rows**
  
  ```
  --check-column xxxx \  
  --incremental lastmodified \  
  --last-value xxxxxxxx
  ```

- **Full extract**
  
  ```
  sqoop import -connect jdbc:oracle:thin:@xxxxx.com:1521/dbname --table tablename  
  --username xxx --password xxx
  ```

- **Full extract with added metadata**
  
  ```
  columns="id, fname,lname,dob,mstatus,'$load_date' as dt_begin, '$end_date' as dt_end,'$curr_flag' as st_cd"  
  query="select $columns from $table where $CONDITIONS"  
  sqoop import --options-file "/xxx/sqoop-options.txt" --query "$query" --split-by "$column_to_split_by" --target-dir "$table-$load_date" \
  ```
CDC Using Sqoop and Hive – Full extract case

➢ **Find Inserts**
```
create table inserts as
select a.id, a.fname, a.lname,
a.dob, a.mstatus,
FROM_UNIXTIME_UNIX_TIMESTAMP(), 'yyyy-MM-dd hh:mm:ss') as
dt_begin, '' as dt_end, 'I' as st_cd from stagingtable a
left outer join targettable b on a.id=b.id
where b.id IS NULL;
```

➢ **Find Deletes**
```
create table deletes as
select a.id, a.fname, a.lname, a.dob, a.mstatus, a.dt_begin,
FROM_UNIXTIME_UNIX_TIMESTAMP(), 'yyyy-MM-dd hh:mm:ss') dt_end, 'D' as st_cd from targettable a
left outer join stagingtable b on a.id=b.id
where b.id is null;
```

➢ **Find Updates**
```
create table updates as
select a.id, a.fname, a.lname, a.dob, a.mstatus,
FROM_UNIXTIME_UNIX_TIMESTAMP(), 'yyyy-MM-dd hh:mm:ss') as dt_begin,
'' as dt_end, 'U' as st_cd
from stagingtable a
join targettable b
on a.ID = b.ID
where
(a.FName <> b.FName OR a.LName <> b.LName
OR a.DoB <> b.DoB OR a.MStatus <> b. MStatus);
```
SCD Type-2

Apply Updates

create table updatedrecs as
select a.id, a.fname, a.lname, a.dob, a.mstatus, a.dt_begin, '2014-08-03 00:00:00' as dt_end, a.st_cd from targettable a
right outer join updates b on a.id=b.id;

Put all together

insert overwrite table targettable as
select a.id, a.fname, a.lname, a.dob, a.mstatus, a.dt_begin, a.dt_end, st_cd from targettable a
left outer join updates b on a.id=b.id
left outer join deletes c on a.id=c.id
where b.id is null and
c.id is null
union all
select * from updatedrecs
union all
select * from updates
union all
select * from deletes
union all
select * from inserts
SCD Type-2

Final Data looks like this...

<table>
<thead>
<tr>
<th>id</th>
<th>fname</th>
<th>lname</th>
<th>dob</th>
<th>mstatus</th>
<th>dt_begin</th>
<th>dt_end</th>
<th>st_cd</th>
</tr>
</thead>
<tbody>
<tr>
<td>135001</td>
<td>john</td>
<td>Doe</td>
<td>1960-01-12 00:00:00</td>
<td>married</td>
<td>2014-08-01 00:00:00</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>135003</td>
<td>John Q</td>
<td>Public</td>
<td>1985-05-06 00:00:00</td>
<td>divorced</td>
<td>2014-08-01 00:00:00</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>135005</td>
<td>Jane</td>
<td>Roe</td>
<td>1982-08-16 00:00:00</td>
<td>married</td>
<td>2014-08-01 00:00:00</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>135004</td>
<td>Judy</td>
<td>Doe</td>
<td>1980-10-22 00:00:00</td>
<td>single</td>
<td>2014-08-01 00:00:00</td>
<td>2014-08-03 00:00:00</td>
<td>I</td>
</tr>
<tr>
<td>135004</td>
<td>Judy</td>
<td>Smith</td>
<td>1980-10-22 00:00:00</td>
<td>married</td>
<td>2014-08-03 07:27:02</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>135002</td>
<td>Joh</td>
<td>Doe Jr</td>
<td>1985-12-31 00:00:00</td>
<td>single</td>
<td>2014-08-01 00:00:00</td>
<td>2014-08-03 00:00:00</td>
<td>D</td>
</tr>
<tr>
<td>135006</td>
<td>Joe</td>
<td>Bloggs</td>
<td>1970-07-14 00:00:00</td>
<td>single</td>
<td>2014-08-03 07:15:02</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>
Thinking from Hadoop Perspective...

- Let’s take a step back
- Do we really need this complex process?
- Whole purpose of CDC on RDBMS was…
  - Instead of capturing complete data, capture changes and apply them
    - Saves space
    - Saves time
    - Saves cost

- Does this really matter on Hadoop?
Hadoop Snapshot Approach

- Full extract the table on daily basis into partitions of staging table. Find changes using queries.

/staging_table1/load-date=2014-08-01/part-00001
/part-00002
/load-date=2014-08-02/part-00001
/part-00002

Advantages:

- Easy to implement
- History maintained for a longer period of time
- Changes can be found in a range
- Happy business user
Tools

➢ Tools with CDC feature on Hadoop

- Informatica Big Data Edition
- Talend
- Syncsort DMX-h
- IBM InfoSphere BigInsights
- Pentaho
- Databus
- Shareplex connector for Hadoop

©2013 CenturyLink. All Rights Reserved. The CenturyLink mark, pathways logo and certain CenturyLink product names are the property of CenturyLink, Inc. All other marks are the property of their respective owners.
Future

- **Hadoop community is actively working on enabling ACID support on HIVE** ([HIVE-5317](#))

  ```
  INSERT INTO tbl SELECT ...
  INSERT INTO tbl VALUES ...
  UPDATE tbl SET ... WHERE ...
  DELETE FROM tbl WHERE ...
  MERGE INTO tbl USING src ON ... WHEN MATCHED THEN ...
  WHEN NOT MATCHED THEN ...
  SET TRANSACTION LEVEL ...
  BEGIN/END TRANSACTION
  ```

- **Hive on Spark** ([HIVE-7292](#))
  Spark as an open-source data analytics cluster computing framework has gained significant momentum recently. This new feature enable Spark as a third execution backend, parallel to MapReduce and Tez.
Key Takeaways

- Hadoop's role in data warehousing is evolving rapidly in use cases like archiving data, staging data, offloading data and replacing existing data warehousing systems.
- Hadoop will be the strategic environment of choice for new data types, new analytics at low cost.
- Easy to implement and integrate. But, challenges exists. Skills to be upgraded and existing systems needs changes too.
- CDC and SCD can be implemented on hadoop using manual coding or using tools(no coding).
- CDC and SCD needs can be achieved using Hadoop way.
- Future enhancements of Hive will make life easy.
Questions?

If you have additional questions, please contact ankam.venkat@centurylink.com

Thank you!