The Database Emperor Has No Clothes

Hadoop's Inherent Advantages Over RDBMS for DW / BI; Especially in the "Big Data" Era

David Teplow, Integra Technology Consulting

Your Presenter

- Became an Oracle user in 1981 (Version 2)
- 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

RDBMS History

- 1970 - Relational Database Management Systems (RDBMS) were specified by IBM's E.F. Codd
- 1979 - First commercial RDBMS released by Oracle Corporation (then Relational Software, Inc.)
- 1986 - Oracle goes public; RDBMS is firmly entrenched as a standard
- late 1980s - Decision Support Systems (DSS) emerge (a.k.a. Data Warehouse (DW) or Business Intelligence (BI) systems)

The DW / BI Process

- EXTRACT: Moving data from operational systems into a Staging Area or Operational Data Store (ODS)
- TRANSFORM: Cleansing and transforming that data so it's consistent and suitable for loading into the DW schema, which is usually ***Dimensional rather than Relational***
- LOAD: Loading data into the DW either as a full refresh or an incremental load, along with any required "Housekeeping" operations such as:
  - Building/rebuilding aggregate tables
  - Recreating indexes that were dropped to speed up the load
  - Maintaining data partitions that are often used for fact tables

Extract Issues

- The Staging Area or ODS carries with it:
  - The ***expense*** of another database license
  - The ***time*** of a Database Administrator (DBA) to monitor and maintain it, plus the ***time*** it takes to physically move the data
- Multiply the above times however many Data Marts you may have

Transform Issues

- **COMPLEXITY** of de-duplicating, de-normalizing, translating, homogenizing and/or aggregating data
- Collecting and maintaining **meta data** (i.e. “data about the data” such as definitions, sources, lineage, derivations, etc.)
- **Iterations** that entail changes to the target schema along with careful and often significant changes to the ETL process
- ***De-normalized data is susceptible to update anomalies***

Load Issues

- Loading large volumes of data typically requires:
  - Parallel processing
  - Data partitioning
  - Dropping indexes pre-load, then recreating them post-load
- The above requires expensive software (Oracle Enterprise Edition) and hardware (CPU, Disk and Memory)

Big Data

- Until recently, data was all about ***transactions***
- We now have data about what happened before that transaction (or non-transaction)
  - Physical path: surveillance video, GPS, location service
  - Virtual path: server log files / "click-stream", product review, shopping cart removal or abandonment, jump to a competitor's site
- And about what happened after that transaction
  - RFID, tweets, likes, blogs, yelps, reviews, customer service calls, product returns

Big Data Issues

- VOLUME
- VELOCITY
- VARIETY

Velocity

- Big Data flows from the "Internet of Things" that are ***always on***: social media, mobile devices, RFID tags, web logs, sensor networks, on-board computers, etc.
- By default, RDBMS is ***not always on*** / available
- Oracle High-Availability requires RAC (server failover) and/or Data Guard (data replication)

  RAC will add over 48% to the cost of your Oracle license;
  Data Guard over 21%

  (Per Processor License costs from the Oracle Technology Global Price List dated 7/19/2012)

Variety

- Big Data is seldom **Structured** / usually **Semi-Structured** or **Unstructured**
  - **Structured**: most data about a product return; some data about a customer service call
  - **Semi-Structured**: server log files; likes on a Facebook page
  - **Unstructured**: surveillance video; product-related articles, reviews, comments or tweets
- Relational Tables are **Structured** – comprised of fields that are rigidly typed (6-digit integer, fixed or variable length character string of exactly X or no more than Y characters, etc.) and often come with constraints (range checks, foreign key lookups, etc.)

A New Approach

- Google publishes papers on its Google File System and MapReduce framework (December, 2004)
    - Think "divide and conquer", but massively parallel and infinitely scalable
- Yahoo! adopts the MapReduce framework and Doug Cutting contributes this work to the open source community by creating the Apache Hadoop project (named for his son's toy elephant)
- Facebook now uses Hadoop to process over 500 Petabytes of data

Hadoop's Advantages

- Data is stored in the Hadoop Distributed File System (HDFS) in its raw form
  - No database schema
  - No index schema
- No need to normalize your data, de-normalize your data, or to transform your data in any way

  ETL without the dreaded "T" - just Extract and Load!
- Data is readily stored in Hadoop regardless of the volume (inexpensive, commodity disks are the norm), velocity (no transformation process to slow things down) or variety (no database schema to conform to)

Hadoop's Advantages

- ***Sqoop*** (http://sqoop.apache.org/) is a tool to import/export data to/from HDFS from any RDBMS or other data source
  - Import example:

```
$ sqoop import --connect jdbc:mysql://localhost/acmedb \
  --table ORDERS --username test --password ****
```

  - Export example:

```
$ sqoop export --connect jdbc:mysql://localhost/acmedb \
 --table ORDERS --username test --password **** \
 --export-dir /user/arvind/ORDERS
```

Hadoop's Advantages

- Designed to be massively parallel (high performance) and fault tolerant (high availability)
- Data is replicated on three separate servers; if a node is unavailable or just slow, then one of the other nodes takes over the processing of that dataset
- New or recovered servers are automatically registered with the system and immediately taken advantage of for processing and storage

Hadoop's Disadvantage

- Data in Hadoop is accessed by MapReduce routines that are written in Java, Python, Ruby, etc.

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class WordCount {

  public static class Map extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
      }
    }
  }

  public static class Reduce extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }

  public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
  }
}
```

INTEGRA TECHNOLOGY CONSULTING

Hadoop-related Projects

- ***Pig*** (http://pig.apache.org/) - a scripting language that can be used to eliminate much of the complexity of a programming language like Java

➢ **Example:** Suppose we have a table urls (url, category, pagerank); the following SQL query finds, for each sufficiently large category, the average pagerank of high-pagerank urls in that category:

```
SELECT category, AVG(pagerank)
FROM urls
WHERE pagerank > 0.2
GROUP BY category HAVING COUNT(*) > 1000000
```

An equivalent ***Pig Latin*** program is the following:

```
good_urls = FILTER urls BY pagerank > 0.2;
groups = GROUP good_urls BY category;
big_groups = FILTER groups BY COUNT(good_urls)>1000000;
output = FOREACH big_groups GENERATE category, AVG(good_urls.pagerank);
```

Hadoop-related Projects

- ***Hive*** (http://hive.apache.org/) allows you to build a metadata layer (called "Metastore") on top of Hadoop, then access data using an SQL-like interface called ***HiveQL;*** ***Hue Beeswax*** provides a browser-based graphical UI for working in ***Hive***
- ***HBase*** (http://hbase.apache.org/) is a column-oriented data store modeled after Google's Bigtable, which leverages the distributed data storage provided by Hadoop and HDFS (as Bigtable does with the Google File System).

Hadoop-related Projects

- ***Impala*** (a.k.a. "Cloudera Enterprise RTQ") lets you run SQL queries against Hadoop in real time
- Unlike ***Pig*** and ***Hive***, which must be compiled into MapReduce routines then run in "batch mode", ***Impala*** runs interactively and directly against HDFS or HBase so that query results begin to return immediately
- **Prediction:** BI tool vendors will build ***Impala***-based interfaces to Hadoop; some already have
- Currently in beta with a production release planned for the first quarter of 2013

INTEGRA TECHNOLOGY CONSULTING

Learn about Hadoop

- http://hadoop.apache.org/
- http://www.cloudera.com/
- http://hortonworks.com/

Conclusion

- Relational databases have been around for over 30 years, and have proven to be a far better way to process data than their predecessors, especially for transactional systems
- For DSS, which has been around for the past 20 years, we adapted RDBMS in ways that are unnatural in terms of the Relational Model, and inefficient in terms of the data staging and transformation processes they create
- Google and Yahoo! found that Relational databases were simply unable to handle the massive volumes of data they have to deal with and came up with Hadoop (once again, necessity is the mother of invention)
- RDBMS are no longer the only choice for DSS / DW / BI and, in my opinion, no longer the best available option - *especially where Big Data is involved*

Thank You

Q & A

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