CIS 602-01: Scalable Data Analysis

Machine Learning

Dr. David Koop
## OLTP vs. OLAP

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>users</strong></td>
<td>clerk, IT professional</td>
<td>knowledge worker</td>
</tr>
<tr>
<td><strong>function</strong></td>
<td>day to day operations</td>
<td>decision support</td>
</tr>
<tr>
<td><strong>DB design</strong></td>
<td>application-oriented</td>
<td>subject-oriented</td>
</tr>
<tr>
<td><strong>data</strong></td>
<td>current, up-to-date</td>
<td>historical, summarized, multidimensional</td>
</tr>
<tr>
<td></td>
<td>detailed, flat relational</td>
<td>integrated, consolidated</td>
</tr>
<tr>
<td></td>
<td>isolated</td>
<td></td>
</tr>
<tr>
<td><strong>usage</strong></td>
<td>repetitive</td>
<td>ad-hoc</td>
</tr>
<tr>
<td><strong>access</strong></td>
<td>read/write</td>
<td>lots of scans</td>
</tr>
<tr>
<td></td>
<td>index/hash on prim. key</td>
<td></td>
</tr>
<tr>
<td><strong>unit of work</strong></td>
<td>short, simple transaction</td>
<td>complex query</td>
</tr>
<tr>
<td><strong># records accessed</strong></td>
<td>tens</td>
<td>millions</td>
</tr>
<tr>
<td><strong>#users</strong></td>
<td>thousands</td>
<td>hundreds</td>
</tr>
<tr>
<td><strong>DB size</strong></td>
<td>100MB-GB</td>
<td>100GB-TB</td>
</tr>
<tr>
<td><strong>metric</strong></td>
<td>transaction throughput</td>
<td>query throughput, response</td>
</tr>
</tbody>
</table>

[Han et al., 2011]
From Tables and Spreadsheets to Data Cubes

• A data warehouse is based on a multidimensional data model which views data in the form of a data cube

• A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - Dimension tables, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - Fact table contains measures (such as dollars_sold) and keys to each of the related dimension tables

• In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

[Han et al., 2011]
Data Cube: A Lattice of Cuboids

0-D (apex) cuboid

1-D cuboids

2-D cuboids

3-D cuboids

4-D (base) cuboid

[Han et al., 2011]
Data Cube Measures: Three Categories

- **Distributive**: if the result derived by applying the function to n aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., `count()`, `sum()`, `min()`, `max()`

- **Algebraic**: if it can be computed by an algebraic function with M arguments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., `avg()`, `min_N()`, `standard_deviation()`

- **Holistic**: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., `median()`, `mode()`, `rank()`

[Han et al., 2011]
Multidimensional Data

- Sales volume as a function of product, month, and region

Dimensions: *Product, Location, Time*
Hierarchical summarization paths

[Han et al., 2011]
A Sample Data Cube

Total annual sales of TVs in U.S.A.

<table>
<thead>
<tr>
<th>Product</th>
<th>Date</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td>1Qtr</td>
<td>U.S.A.</td>
</tr>
<tr>
<td></td>
<td>2Qtr</td>
<td>Canada</td>
</tr>
<tr>
<td></td>
<td>3Qtr</td>
<td>Mexico</td>
</tr>
<tr>
<td>VCR</td>
<td>4Qtr</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>sum</td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OLAP Operations

[Han et al., 2011]
Efficient Processing of OLAP Queries

• Determine which operations should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., dice = selection + projection

• Determine which materialized cuboid(s) for OLAP operation:
  - Query: \{brand, province_or_state\} with “year = 2004”
  - 4 materialized cuboids available:
    1. \{year, item_name, city\}
    2. \{year, brand, country\}
    3. \{year, brand, province_or_state\}
    4. \{item_name, province_or_state\} where year = 2004
  - Which should be selected to process the query?

[Han et al., 2011]
NanoCubes: Interactive Data Cube Exploration

Linked view of tweets in San Diego, US [Lins et al., 2013]
Data Cube Aggregations

Relation A

<table>
<thead>
<tr>
<th>Country</th>
<th>Device</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>Android</td>
<td>en</td>
</tr>
<tr>
<td>US</td>
<td>iPhone</td>
<td>ru</td>
</tr>
<tr>
<td>South Africa</td>
<td>iPhone</td>
<td>en</td>
</tr>
<tr>
<td>India</td>
<td>Android</td>
<td>en</td>
</tr>
<tr>
<td>Australia</td>
<td>iPhone</td>
<td>en</td>
</tr>
</tbody>
</table>

Aggregation B

<table>
<thead>
<tr>
<th>Country</th>
<th>Device</th>
<th>Language</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>All</td>
<td>5</td>
</tr>
</tbody>
</table>

Group By on Device, Language C

<table>
<thead>
<tr>
<th>Country</th>
<th>Device</th>
<th>Language</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Android</td>
<td>en</td>
<td>2</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</td>
<td>en</td>
<td>2</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</td>
<td>ru</td>
<td>1</td>
</tr>
</tbody>
</table>

Cube on Device, Language D

<table>
<thead>
<tr>
<th>Country</th>
<th>Device</th>
<th>Language</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>All</td>
<td>5</td>
</tr>
<tr>
<td>All</td>
<td>Android</td>
<td>All</td>
<td>2</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</td>
<td>All</td>
<td>3</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>en</td>
<td>4</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>ru</td>
<td>1</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</td>
<td>ru</td>
<td>1</td>
</tr>
<tr>
<td>All</td>
<td>Android</td>
<td>en</td>
<td>2</td>
</tr>
<tr>
<td>All</td>
<td>iPhone</td>
<td>en</td>
<td>2</td>
</tr>
</tbody>
</table>

Equivalent to Group By on all possible subsets of \{Device, Language\}

[Lins et. al, 2013]
Building a Nanocube

Five Tweets: Location and Device

Indexing Schema

\[
S = [\ell_{\text{spatial1}}, \ell_{\text{spatial2}}, \ell_{\text{device}}]
\]
TopKube: Rankings

- Shots by time, number of points scored, and location on the court

<table>
<thead>
<tr>
<th>team</th>
<th>player</th>
<th>time</th>
<th>pts</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLE</td>
<td>L. James</td>
<td>5</td>
<td>0</td>
<td>13</td>
<td>28</td>
</tr>
<tr>
<td>BOS</td>
<td>R. Rondo</td>
<td>5</td>
<td>2</td>
<td>38</td>
<td>26</td>
</tr>
<tr>
<td>CLE</td>
<td>L. James</td>
<td>7</td>
<td>3</td>
<td>42</td>
<td>35</td>
</tr>
</tbody>
</table>

- Query: Ranked list of the 50 players who took the most shots
  - SELECT player, count(*) AS shots FROM table GROUP BY player ORDER BY shots DESC LIMIT 50

- Query: Rank the top 50 players by points made:
  - SELECT player, sum(pts) AS points FROM table GROUP BY player ORDER BY points DESC LIMIT 50

[F. Miranda et al., 2017]
Example: One Spatial Dim. and A,B,C events

![Diagram](image)

[F. Miranda et al., 2017]
Three Algorithms

• **Sweep**: Use a priority queue where the product bin with the current smallest key is on the top
• **Threshold**: don't do a full scan, use extra information about ranking
• **Hybrid**:
  - Threshold has best theoretical guarantee but some sparse cases can be faster
  - Use Sweep on small input lists, Threshold on denser problem

[F. Miranda et al., 2017]
Top Hashtags in Paris related to Charlie Hebdo

1. Select Paris Area
3. Select this Spike and Observe Top-10 Hashtags
4. Select Charlie Hebdo’s Top Hashtags and Observe its Temporal Volume Pattern

- #jesuischarlie 4,456
- #charliehebdo 4,190
- #lrt 1,146
- #paris 607
- #gagnetaplace 447
- #charliehebdo 418
- #off 402
- #lt 335
- #noussommescharlie 197
- #rip 187

[F. Miranda et al., 2017]
Assignment 3

- Analysis of the IRS Form 990 Filings Data
- Sign up for AWS Educate
- Part 1: Run Spark Locally over Index
- Part 2: Run Spark Locally over Subset
- Part 3: Run Spark on AWS over Full Dataset (Year)
- Windows Issues:
  - Install Spark separately from conda's pyspark install
  - Run notebook normally and use findspark to link to the pyspark libraries
Test 2

• This Thursday at 5pm
• Topics
  - Cloud Computing
  - Cloud Workloads
  - MapReduce
  - Cluster Computing
  - Data Cleaning
  - Data Integration
  - NoSQL Databases
  - NewSQL Databases
  - Data Cubes
Project Proposals

• Due Monday, November 20
• Turn in via myCourses
• Identify dataset or project and provide detailed writeup
• Must have scalability concerns and those must be described
• AWS Public Datasets
• Awesome Public Datasets
• Specific Datasets:
  - AIS Data
  - NBA Data
  - IRS 990 Data
  - Taxi Data
Machine Learning

P. Domingos
Scalable Machine Learning

- Tom Augspurger's Post:
  - https://tomaugspurger.github.io/scalable-ml-01.html
  - Think about why you need to scale…
    - Size?
    - Time?
    - Don't forget statistics
  - Using pipelines in scikit-learn
  - Using dask to scale to multiple cores or the cloud
TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems

Google (M. Abadi et al.)
TensorFlow

- TensorFlow is:
  - An **interface** for expressing machine learning algorithms
  - An **implementation** for executing such algorithms
- Runs on lots of different systems
- Google Brain: very-large-scale deep **neural networks**
- Was internal tool, now open-source
- Precursor was DistBelief
- Others: Torch, Caffe, GraphLib, Theano
TensorFlow Programming Model

- Operations and Kernels: abstract computation vs. implementation
- Session: construct graph and run
- Variables: keep state, survive across multiple executions
- Example:

```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
C = [...] # Cost computed as a function of Relu

s = tf.Session()
for step in xrange(0, 10):
    input = ...construct 100-D input array ...
    result = s.run(C, feed_dict={x: input})
    print step, result
```

Figure 1: Example TensorFlow code fragment
Figure 2: Corresponding computation graph for Figure 1

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element-wise</td>
<td>Add, Sub, Mul, Div, Exp, Log, Greater, Less,</td>
</tr>
<tr>
<td>mathematical</td>
<td>Equal, ...</td>
</tr>
<tr>
<td>operations</td>
<td></td>
</tr>
<tr>
<td>Array operations</td>
<td>Concat, Slice, Split, Constant, Rank, Shape,</td>
</tr>
<tr>
<td></td>
<td>Shuffle, ...</td>
</tr>
<tr>
<td>Matrix operations</td>
<td>MatMul, MatrixInverse, MatrixDeterminant, ...</td>
</tr>
<tr>
<td>Stateful</td>
<td>Variable, Assign, AssignAdd, ...</td>
</tr>
<tr>
<td>operations</td>
<td></td>
</tr>
<tr>
<td>Neural-net</td>
<td>SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...</td>
</tr>
<tr>
<td>building blocks</td>
<td></td>
</tr>
<tr>
<td>Checkpointing</td>
<td>Save, Restore</td>
</tr>
<tr>
<td>operations</td>
<td></td>
</tr>
<tr>
<td>Queue and</td>
<td>Enqueue, Dequeue, MutexAcquire, MutexRelease,</td>
</tr>
<tr>
<td>synchronization</td>
<td></td>
</tr>
<tr>
<td>operations</td>
<td></td>
</tr>
<tr>
<td>Control flow</td>
<td>Merge, Switch, Enter, Leave, NextIteration</td>
</tr>
<tr>
<td>operations</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Example TensorFlow operation types

by the session interface is Run, which takes a set of output names that need to be computed, as well as an optional set of tensors to be fed into the graph in place of certain outputs of nodes. Using the arguments to Run, the TensorFlow implementation can compute the transitive closure of all nodes that must be executed in order to compute the outputs that were requested, and can then arrange to execute the appropriate nodes in an order that respects their dependencies (as described in more detail in 3.1). Most of our uses of TensorFlow set up a Session with a graph once, and then execute the full graph or a few distinct subgraphs thousands or millions of times via Run calls.
import tensorflow as tf

b = tf.Variable(tf.zeros([100])) # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x") # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b) # Relu(Wx+b)
C = [...] # Cost computed as a function of Relu

s = tf.Session()
for step in xrange(0, 10):
    input = ...construct 100-D input array ... # Create 100-d vector for input
    result = s.run(C, feed_dict={x: input}) # Fetch cost, feeding x=input
    print step, result

Figure 1: Example TensorFlow code fragment

Figure 2: Corresponding computation graph for Figure 1

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element-wise math</td>
<td>Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...</td>
</tr>
<tr>
<td>Array operations</td>
<td>Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...</td>
</tr>
<tr>
<td>Matrix operations</td>
<td>MatMul, MatrixInverse, MatrixDeterminant, ...</td>
</tr>
<tr>
<td>Stateful operations</td>
<td>Variable, Assign, AssignAdd, ...</td>
</tr>
<tr>
<td>Neural-net building blocks</td>
<td>SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...</td>
</tr>
<tr>
<td>Checkpointing operations</td>
<td>Save, Restore</td>
</tr>
<tr>
<td>Queue and synchronization operations</td>
<td>Enqueue, Dequeue, MutexAcquire, MutexRelease, ...</td>
</tr>
<tr>
<td>Control flow operations</td>
<td>Merge, Switch, Enter, Leave, NextIteration</td>
</tr>
</tbody>
</table>

Table 1: Example TensorFlow operation types
TensorFlow

• Tensor: think higher-dimensional than scalar, vector
  - Just a matrix

• Devices:
  - Can run on lots of different devices
  - Can specify types of or concrete devices
TensorFlow

J. Dean