What Does ‘Big Data’ Mean and Who Will Win?

Michael Stonebraker

Database Group
MIT Computer Science and Artificial Intelligence Lab
The Meaning of Big Data - 3 V’s

- Big Volume
  - Business stuff with simple (SQL) analytics
  - Business stuff with complex (non-SQL) analytics
  - Science stuff with complex analytics

- Big Velocity
  - Drink from the fire hose

- Big Variety
  - Large number of diverse data sources to integrate
Big Volume - Little Analytics

• Well addressed by data warehouse crowd

• Who are pretty good at SQL analytics on
  – Hundreds of nodes
  – Petabytes of data
Table Stakes

• Scalability
  – Shared nothing architecture
  – On commodity parts

• Don’t care if packaged as an appliance or not
  – As long as it’s not proprietary iron
The Participants

- Row storage and row executor
  - Microsoft Madison, DB2, Netezza, Oracle(!)

- Column store grafted onto a row executor (wannabees)
  - Terradata/Asterdata, EMC/Greenplum

- Column store and column executor
  - HP/Vertica, Sybase/IQ, Paracel

Oracle Exadata is not:
  - a column store
  - a scalable shared-nothing architecture
Performance

- Row stores -- x1
- Column stores -- x50
- Wannabees - somewhere in between
My Prediction

The only successful ‘Big data-Small analytics’ architecture will be a column executor on shared-nothing hardware

- All the successful vendors will have to get there sooner or later
- The “elephants” are in a serious “Innovator’s Dilemma” dilemma

P.S. I started Vertica but I have no current relationship with HP/Vertica
Big Data - Big Analytics

- Complex math operations (machine learning, clustering, trend detection, ....)
  - The world of the “quants” and the “rocket scientists”
  - Mostly specified as linear algebra on array data

- A dozen or so common ‘inner loops’
  - Matrix multiply
  - QR decomposition
  - SVD decomposition
  - Linear regression
Big Data - Big Analytics
An Example

- Consider closing price on all trading days for the last 5 years for two stocks A and B

- What is the covariance between the two time-series?

\[
(1/N) \times \sum (A_j - \text{mean}(A)) \times (B_j - \text{mean}(B))
\]
Now Make It Interesting ...

- Do this for all pairs of 4000 stocks
  - The data is the following 4000 x 1000 matrix

<table>
<thead>
<tr>
<th>Stock</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t_6$</th>
<th>$t_7$</th>
<th>$\ldots$</th>
<th>$t_{1000}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{4000}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hourly data? All securities?
Array Answer

• Ignoring the \( \frac{1}{N} \) and subtracting off the means ....

\[ \text{Stock} \ast \text{Stock}^T \]

• Try this in SQL with some relational simulation of the stock array!!!!
Solution Options
R, SAS, Matlab, et al

• Weak or non-existent data management
  - Do the correlation only for companies with revenue > $1B?

• File system storage

• R doesn’t scale and is not a parallel system
  - Revolution does a bit better
Solution Options
RDBMS alone

• SQL simulator (MadLib) is sloooooow
  - And only does some of the required operations

• Coding operations as UDFs still requires you to simulate arrays on top of tables --- sloooow
  - And current UDF model not powerful enough to support iteration
Solution Options
R + RDBMS

- Have to extract and transform the data from RDBMS table to math package data format (e.g. data frames)
- ‘move the world’ nightmare
- Need to learn 2 systems
- And R still doesn’t scale and is not a parallel system
Solution Options

• New Array DBMS designed with this market in mind
Array Databases beat Relational Database tables on storage efficiency & array computations

### Relational Database

<table>
<thead>
<tr>
<th>I</th>
<th>J</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>32.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>90.9</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>42.1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>96.7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>46.3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>35.4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>35.7</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>41.3</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>81.7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>35.9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>35.3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>89.9</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>53.6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>86.3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>45.9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>27.6</td>
</tr>
</tbody>
</table>

### Array Database

<table>
<thead>
<tr>
<th>I</th>
<th>J</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.5</td>
<td>46.3</td>
<td>81.7</td>
</tr>
<tr>
<td>90.9</td>
<td>35.4</td>
<td>35.9</td>
</tr>
<tr>
<td>42.1</td>
<td>35.7</td>
<td>35.3</td>
</tr>
<tr>
<td>96.7</td>
<td>41.3</td>
<td>89.9</td>
</tr>
</tbody>
</table>

#### 16 cells

- Math functions run directly on native storage format
- Dramatic storage efficiencies as # of dimensions & attributes grows
- High performance on both sparse and dense data

### 48 cells
An Example Array Engine DB
Paradigm4/SciDB

- All-in-one: data management with massively scalable advanced analytics

- Data is updated; not overwritten
  - Supports reproducibility for research and compliance
  - Time-series data
  - Scenario testing

- Supports uncertain data, provenance

- Open source

- Runs in cloud or private grid of commodity HW
Solution Options: Hadoop

- Simple analytics (Hive queries)
  - 100 times slower than a parallel DBMS
- Complex analytics (Mahout or roll-your-own)
  - X100 times Scalapack
- Parallel programming
  - Parallel grep (great)
  - Everything else (awful)
- Hadoop lacks
  - Stateful computations
  - Point-to-point communication
Solution Options: Hadoop

- Lot and lots and lots of people are piloting Hadoop
- Many will hit a scalability wall when they get to production
  - Unless they are doing parallel grep

- My prediction: the bloom will come off the rose
Science Data

• Lots of arrays
  – With complex UDFs
• Some graphs
  – Model as sparse arrays to get a common environment
• RDBMS suck on both
• Under pain of death, do not use the file system!!!!
  – Metadata
  – Provenance
  – Sharing
Big Velocity

• Trading volumes going through the roof
• Breaking all your infrastructure
• And it will just get worse
Big Velocity

• Sensor tagging everything of value sends velocity through the roof
  – E.g. car insurance

• Smart phones as a mobile platform sends velocity through the roof

• State of multi-player internet games must be recorded - sends velocity through the roof
Two Different Solutions

- Big pattern - little state (electronic trading)
  - Find me a ‘strawberry’ followed within 100 msec by a ‘banana’

- Complex event processing (CEP) is focused on this problem
  - Patterns in a firehose

P.S. I started StreamBase but I have no current relationship with the company
Two Different Solutions

- Big state - little pattern
  - For every security, assemble my real-time global position
  - And alert me if my exposure is greater than X

- Looks like high performance OLTP
  - Want to update a database at very high speed
My Suspicion

• There are 3-4 Big state - little pattern problems for every one Big pattern - little state problem
Solution Choices for New OLTP

- Old SQL
  - The elephants

- No SQL
  - 75 or so vendors giving up both SQL and ACID

- New SQL
  - Retain SQL and ACID but go fast with a new architecture
Why Not Use Old SQL?

• Sloooow
  - By a couple orders of magnitude

• Because of
  - Disk
  - Heavy-weight transactions
  - Multi-threading

• See “Through the OLTP Looking Glass”
  - VLDB 2007
No SQL

- Give up SQL
  - Interesting to note that Cassandra and Mongo are moving to (yup) SQL

- Give up ACID
  - If you need ACID, this is a decision to tear your hair out by doing it in user code
  - Can you guarantee you won’t need ACID tomorrow?
VoltDB: an example of New SQL

- A main memory SQL engine
- Open source
- Shared nothing, Linux, TCP/IP on jelly beans
- Light-weight transactions
  - Run-to-completion with no locking
- Single-threaded
  - Multi-core by splitting main memory
- About 100x RDBMS on TPC-C
Big Velocity

- CEP

- Next generation OLTP
  - So-called New SQL
  - Very different architecture than the elephants
  - One-size-does-not-fit-all
Big Variety

• Typical enterprise has 5000 operational systems
  – Only a few get into the data warehouse
  – What about the rest?

• And what about all the rest of your data?
  – Spreadsheets
  – Access data bases
  – Web pages

• And public data from the web?
The World of Data Integration

the rest of your data

enterprise
data warehouse
text
Summary

• The rest of your data (public and private)
  – Is a treasure trove of incredibly valuable information
  – Largely untapped
Data Tamer

• Integrate the rest of your data

• Has to
  – Be scalable to 1000s of sites
  – Deal with incomplete, conflicting, and incorrect data
  – Be incremental

• Task is never done
Data Tamer in a Nutshell

• Apply machine learning and statistics to perform automatic:
  – Discovery of structure
  – Entity resolution
  – Transformation

• With a human assist if necessary
  – Crowd sourcing
  – WYSIWYG tool (Wrangler)
Data Tamer

- MIT research project
- Looking for more integration problems
  - Wanna partner?
Take away

• One size does not fit all

• Plan on (say) 6 DBMS architectures
  – Use the right tool for the job

• Elephants are not competitive
  – At anything
  – Have a bad ‘innovator’s dilemma’ problem