EECS4415: Big Data Systems
Data-Driven Organizations (DDOs)
Overview

- Data Driven Organizations
- Reference Model for DDO Solutions
Data Driven Organizations (DDOs)
How non-DDOs make decisions?

- Intuition
- Ad-hoc or based on few customers feedback
- Look at competition
- Try to be different
- Based on assumptions, that may be wrong
- Without knowing how to validate if it was the right decision
What do DDO’s do?

- Make decisions based on data not intuition
- More precise on what they want to achieve
- Measure and validate with data
There are organizations that have been DDO’s for a long-time

- Walmart
- GE
- Airlines

More data and better tools are enabling more companies to become DDO’s

You have to become a DDO to compete
How do DDO’s do it?

- Collect data
- Develop intuition of the data they got
- Pose questions that they try to answer; Or, search the data for new insights
- Run experiments
- Make decisions and draw insights
Example 1: Email Marketing

Pre DDO
- Did not measure effectiveness of campaigns
- Did not cluster customers
- Did not have tailored campaigns based on data

Result
- Cannibalized their own market
- Offered discounts to customer that would have bought at full price
- Significant loss revenue

Post DDO
- Behavioral clustering
- Predictive analytics
- LTV analysis (Life-time Value)
- Targeted campaigns
- Measure effectiveness

Result
- Increased revenue
Example 2: Application Feature

Pre DDO

- Introduced features on intuition
- No measurable goals

Result

- Sometimes features decreased engagement
- Offered discounts to customer that would have bought at full price
- Occasional lost revenue
- Many features, unknown value

Post DDO

- Experiments, measure
- Do not launch unless measurable benefit

Result

- Fewer failed features
- More successful feature introductions (increased engagement)
- Remove features that do not contribute to metrics
DDO’s
- collect data
- make decisions based on data, not intuition
- use data to drive applications

To be a DDO, you need an efficient way of storing and retrieving data
Reference Model for DDO Solutions

Thanks to Jari Koister
UC Berkeley
A variety of solutions/technologies available

There is no one solution/technology that solves all possible data analytics problems

Most solutions solve a range of problems, but are outstanding on a specific type

How to map problems to DDO solutions?
How to compare alternative DDO solutions?

Need for a Reference Model
Purpose of the Reference Model

- Provides a framework for
  - understanding your needs
  - comparing solutions
- Not complete, but gives an approach to understanding data analytics systems
- Handles **certain type of data** well
- Handles **certain ranges of data size** well
- Performs **certain types of queries and computations** well
A Big Data Approach

- Index/Serving Technology
- Index/Serving Technology
- Index/Serving Technology
- Index/Serving Technology

Processing Technology

Fundamental Data Store Technology
System of Record
Difference in Approach

Traditional Approach
Structured & Repeatable Analysis

Business Users
Determine what question to ask

IT
Structures the data to answer that question

- Monthly sales reports
- Profitability analysis
- Customer surveys

Big Data Approach
Iterative & Exploratory Analysis

IT
Delivers a platform to enable creative discovery

Business
Explores what questions could be asked
- Brand sentiment
- Product strategy
- Maximum asset utilization
- Preventative care

Notice the difference!
To be able to evaluate a solution you need to understand your needs

- What is the structure of your data?
- How big is your data?
- What is the velocity?
- What kind of processing is needed?
- What kind of queries do you want to answer?
- What is the expected latency?
- ...

How to Evaluate a Solution?
Dimensions

Data
What characteristics should be considered with respect to data?

Processing
What characteristics should be considered with respect to processing?

Other dimensions (not covered): cost, implementation complexity
Data Dimension
Data related characteristics

- Structure
- Size
- Sink Rate
- Source Rate
- Quality
- Completeness
What is the type of the data (Variety)?

**Structured**: Well defined schema, data types, understandable by machine

**Unstructured**: Loosely typed (text, pics)

**Semi-structured**: Mix of structured and unstructured. Ex. Well defined schema, but some attributes are unstructured
What is the size of the data (Volume)?

**S**: Megabytes

**M**: Gigabytes

**L**: Tera Bytes

**XL**: 100’s of Tera Bytes

**XXL**: Peta Bytes
Sink Rate/Speed

How fast the data are coming in (Velocity)?

**Very High**: > hundreds of updates per second
**High**: > tens of updates per hours
**Medium**: a few updates per hour
**Low**: Updates daily or less frequently
Source Rate/Speed

How updated is the indexing/speed layer?

**High**: updated in “real-time” as data arrives

**Medium**: Updated on an hourly basis

**Low**: Updates on a daily or less frequently
Quality

How well does the system deal with bad or low quality data (Veracity)?

**High**: can compensate and handle in an automated fashion

**Medium**: can handle but results may be unreliable

**Low**: cannot handle bad or low quality data. Will not provide any results
How well does the system deal with incomplete data?

**Incomplete**: can enrich and complete data efficiently

**Semi-complete**: provides some capabilities for completing and enriching data

**Complete**: requires data to be complete before processing
Processing Dimension

Processing related characteristics
- Query Selectivity
- Query Execution Time
- Aggregation
- Processing Time
- Join
- Precision
Selectivity

Is it better at high or low query selectivity scenarios? (In a High Selectivity scenario a query predicate is more selective, meaning that only small percentage of data rows satisfy the query)

High: expect < 20% of data to be selected
Medium: expect 20-80% of data to be selected
Low: expect > 80% of data to be selected
What query response time is the system designed to meet?

**Short**: milliseconds or less than a few seconds
**Medium**: speed of thought, or most 30 seconds
**Long**: minutes or tens of minutes
What is the level of expressiveness and computational capabilities of aggregations?

**Advanced**: Roll-ups, drill-downs, lattice, cuboids

**Medium**: Aggregations over multiple dimensions

**Basic**: Simple counters
What processing time is expected for batch jobs? (24 hours is an important limitation for many applications)

**Short**: < 1 hour  
**Medium**: < 12 hours  
**Long**: > 24 hours
What is the level of expressiveness and computational capabilities of joins? (Join is a common operation; there is a variety of joins that are suitable for different data distributions, data sizes etc.)

**Advanced**: a variety of joins for different functional and optimization scenarios

**Basic**: limited capability for join

**None**: No join supported
What is the expected output precision? (May be impacted by potential loss of data, approximations, sampling, etc.)

**Exact**: Always exact, includes full data set

**Approximate**: Approximates result for example through sampling

**Lossy**: May miss some data for the benefit of speed or scale. Or may count data twice in the event of recovery
Example DDO Solutions
Dimensions: Examples

**DDO solutions to investigate**

**RDBMS:** Relational model with powerful querying capabilities

**HDFS+M/R:** Batch oriented system for processing and storing large data sets

**Storm:** A stream processing system that can compute in real-time over large streams

**BlinkDB:** Experimental system for approximate query answering over large data that trade error over response time
## RDBMS: Data

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Structured</td>
<td>Good with structured data. Can store unstructured too</td>
</tr>
<tr>
<td>Size</td>
<td>S--&gt;L</td>
<td>Efficiently deals with up to L size data</td>
</tr>
<tr>
<td>Sink Rate</td>
<td>High</td>
<td>Depending on the number of records being pushed into a system the ingest capacity will vary. Databases can deal with frequent updates up to a point, but when updates are in hundreds per second the data base will have trouble keeping up</td>
</tr>
<tr>
<td>Source Rate</td>
<td>High</td>
<td>Can update results computations quickly and be triggered in real-time</td>
</tr>
<tr>
<td>Quality</td>
<td>Medium</td>
<td>Databases in themselves are not good at handling low quality data. But they can be programmed to do cleaning and other tasks to prepare the data</td>
</tr>
<tr>
<td>Completeness</td>
<td>Incomplete</td>
<td>Databases can deal with missing values or be used to complete data before processing</td>
</tr>
</tbody>
</table>
## RDBMS: Processing

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Selectivity</td>
<td>High, Low</td>
<td>Normally databases can deal with both low and high selectivity queries. They have facilities such as indices to optimize for specific use cases</td>
</tr>
<tr>
<td>Query Time</td>
<td>Short, Long</td>
<td>Normally intended for quick queries, but also used for long running queries</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Advanced</td>
<td>Has advanced facilities for aggregating and grouping data in batch or in realtime</td>
</tr>
<tr>
<td>Processing Time</td>
<td>Short, Long</td>
<td>Facilitates both short and long running processes</td>
</tr>
<tr>
<td>Join</td>
<td>Advanced</td>
<td>Relational databases normally support a variety of join’s for different functional and optimization scenarios</td>
</tr>
<tr>
<td>Precision</td>
<td>Exact</td>
<td>Queries and processes are normally over the complete dataset</td>
</tr>
</tbody>
</table>
## HDFS + M/R: Data

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Structured and unstructured</td>
<td>Generally used to handle both structured and unstructured data</td>
</tr>
<tr>
<td>Size</td>
<td>XL, XXL</td>
<td>Intended for very large data sets</td>
</tr>
<tr>
<td>Sink Rate</td>
<td>very high, high</td>
<td>Can be used to store incoming data at high rate. No ACID properties and immutable data facilitates a fast storage process</td>
</tr>
<tr>
<td>Source Rate</td>
<td>medium, low</td>
<td>Updates are not fast due to longer processing cycles</td>
</tr>
<tr>
<td>Quality</td>
<td>medium</td>
<td>Can be used to deal with lower quality data</td>
</tr>
<tr>
<td>Completeness</td>
<td>incomplete</td>
<td>Can be used to enrich and complete data</td>
</tr>
</tbody>
</table>
# HDFS + M/R: Processing

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Selectivity</td>
<td>low</td>
<td>general a more efficient method when selectivity is low. But can of course deal with high selectivity as well, has not indices though</td>
</tr>
<tr>
<td>Query Time</td>
<td>long</td>
<td>Queries take a long time to execute</td>
</tr>
<tr>
<td>Aggregation</td>
<td>medium</td>
<td>almost anything can be done, but certain types of operations may not be as efficient</td>
</tr>
<tr>
<td>Processing Time</td>
<td>long, medium</td>
<td>suitable for long and medium length processes</td>
</tr>
<tr>
<td>Join</td>
<td>basic</td>
<td>There are many abstractions such as Pig that provide powerful Join capabilities on M/R</td>
</tr>
<tr>
<td>Precision</td>
<td>exact</td>
<td>Normally operates on the full data set</td>
</tr>
</tbody>
</table>
## Storm: Data

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Structured</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>XL, XXL</td>
<td>Designed to efficiently deal with large sets of streaming data</td>
</tr>
<tr>
<td>Sink Rate</td>
<td>Very high</td>
<td></td>
</tr>
<tr>
<td>Source Rate</td>
<td>High</td>
<td>A serving layer can be updated in real-time</td>
</tr>
<tr>
<td>Quality</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Completeness</td>
<td>Complete</td>
<td>Generally expects data to be complete for processing. But it can be augmented</td>
</tr>
</tbody>
</table>
# Storm: Processing

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Selectivity</td>
<td>high to low</td>
<td>Selectivity is not the major factor. Although high selectivity would result in larger streaming graphs</td>
</tr>
<tr>
<td>Query Time</td>
<td>N/A</td>
<td>Is not queried directly, rather results are pushed to a serving component</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Medium</td>
<td>Generally better at simpler aggregations over incoming data streams</td>
</tr>
<tr>
<td>Processing Time</td>
<td>short</td>
<td>Processing is designed to take place in real-time</td>
</tr>
<tr>
<td>Join</td>
<td>basic</td>
<td>Streams can be joined, but there are limitations such as over which datasets joins can be made etc</td>
</tr>
<tr>
<td>Precision</td>
<td>lossy</td>
<td>Provides at-least-once semantics</td>
</tr>
</tbody>
</table>
## BlinkDB: Data

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Structured</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>XL, XXL</td>
<td>Is designed to handle interactive queries over large datasets. No reason to approximate if datasets or smaller</td>
</tr>
<tr>
<td>Sink Rate</td>
<td>N/A</td>
<td>Uses HDFS as underlying storage</td>
</tr>
<tr>
<td>Source Rate</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Quality</td>
<td>low</td>
<td>Designed to process mainly quality data</td>
</tr>
<tr>
<td>Completeness</td>
<td>Complete</td>
<td></td>
</tr>
</tbody>
</table>
## BlinkDB: Processing

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Characterization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Selectivity</td>
<td>High, Low</td>
<td></td>
</tr>
<tr>
<td>Query Time</td>
<td>Short</td>
<td>It is designed to give shortest possible response time, but with bounded errors</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Medium</td>
<td>Same basic capabilities as Hive and other big data systems</td>
</tr>
<tr>
<td>Processing Time</td>
<td>Short, medium</td>
<td>Designed to support shorter processing time over big data sets</td>
</tr>
<tr>
<td>Join</td>
<td>Basic</td>
<td>Basic join support as provided by Hive and other systems</td>
</tr>
<tr>
<td>Precision</td>
<td>Approximate</td>
<td>Allows errors within bounds by design</td>
</tr>
</tbody>
</table>
# Dimensions: Summary

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Dimension</th>
<th>RDBMS</th>
<th>M/R</th>
<th>Storm</th>
<th>Blink DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Structure</td>
<td>structured</td>
<td>all</td>
<td>all</td>
<td>structured</td>
</tr>
<tr>
<td></td>
<td>Size</td>
<td>S→L</td>
<td>XL,XXL</td>
<td>S→XXL, streaming</td>
<td>XL,XXL</td>
</tr>
<tr>
<td></td>
<td>Sink Rate</td>
<td>high</td>
<td>very high, high</td>
<td>very high</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Source Rate</td>
<td>high</td>
<td>medium, low</td>
<td>high</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Quality</td>
<td>medium</td>
<td>medium</td>
<td>high, low</td>
<td>low</td>
</tr>
<tr>
<td></td>
<td>Completeness</td>
<td>incomplete</td>
<td>Incomplete</td>
<td>complete</td>
<td>complete</td>
</tr>
<tr>
<td>Processing</td>
<td>Selectivity</td>
<td>high, low</td>
<td>low</td>
<td>high, low</td>
<td>high, low</td>
</tr>
<tr>
<td></td>
<td>Query Execution time</td>
<td>short,long</td>
<td>long</td>
<td>N/A</td>
<td>medium</td>
</tr>
<tr>
<td></td>
<td>Aggregation</td>
<td>advanced</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
</tr>
<tr>
<td></td>
<td>Processing time</td>
<td>short, long</td>
<td>long, short</td>
<td>short</td>
<td>short, medium</td>
</tr>
<tr>
<td></td>
<td>Join</td>
<td>advanced</td>
<td>basic</td>
<td>basic</td>
<td>basic</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>exact</td>
<td>exact</td>
<td>lossy</td>
<td>approximate</td>
</tr>
</tbody>
</table>