REAL-TIME ANALYTICS ON A HIGH PERFORMANCE DATABASE PLATFORM

PETER MILNE
DIRECTOR OF APPLICATION ENGINEERING

BIG DATA STRATEGY
VILNIUS
MAY 2014
What is: “Analytics”?

Where is the wisdom we have lost in knowledge? Where is the knowledge we have lost in information?
-T.S. Elliot 1934

Analytics is:

■ Finding Knowledge in Information
■ Finding Signal in Noise
Signal in Noise

We see/hear noise

But its all signal
How do we find Signal in Noise

- Smoothing & Filtering
- Classification
- Similarity
- Dimension Reduction
- Aggregation
- Voodoo + Alchemy (just joking)
Smoothing & Filtering
Smoothing

- **Moving average** - each point in the signal is replaced with the average of “m” adjacent points, where “m” is a positive integer called the smooth width.
Filtering

- Pass Filters – High, Low, Band, Butterworth, etc

- Recursive Filters – Kalman Filter or linear quadratic estimation
Classification
Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).
A naïve Bayes classifier is a simple \textbf{probabilistic classifier} based on applying Bayes' theorem with strong (naive) independence assumptions. - Wikipedia

\[
P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}
\]

\[
P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)
\]
Similarity
Cosine Similarity

“Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. .... Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in $[0,1]$.” - Wikipedia
Dynamic Time Warping - DTW

“In time series analysis, dynamic time warping (DTW) is an algorithm for measuring similarity between two temporal sequences which may vary in time or speed.

... A well known application has been automatic speech recognition, to cope with different speaking speeds.

... Other applications include speaker recognition and online signature recognition. Also it is seen that it can be used in partial shape matching application.” - Wikipedia
Dimension Reduction
The PCA method finds the directions with the greatest variance in the data, called principal components.

Eigenfaces - Facial recognition, OpenCV
Linear discriminant analysis (LDA) and the related Fisher's linear discriminant are methods used in statistics, pattern recognition and machine learning to find a linear combination of features which characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.” - Wikipedia

Bankruptcy prediction
Facial recognition
Marketing
My Brain is Full
Technologies
Complex Event Processing (CEP)

Storm

- Weather satellite
- Stationary camera
- Smart city
- Mobile phone
- Logistics center
- Health device
- Traffic

Data Collection/Sensing

- Sensing (Stream)
- Event input
- Status transition
- Call

Navigation (Control)

- Rules
- No intruders in house
- Unanticipated downpour expected soon
- Half-off coupon issued by store D
- Street A is currently congested, we recommend street B
- Your child detoured from usual route to and from school, and is now at park C

Extracting useful information from Big Data

Offering useful information as appropriate
Map Reduce

Distributed Database Cluster
Hadoop

➤ Large
➤ Powerful
➤ Capable
➤ Methodical
➤ Batch

- Input: HDFS - PetaBytes of “RAW” data
- Output: NoSQL – Signal from Noise
HOT ANALYTICS – In Real-time
Key Challenges

- Handle extremely high rates of read/write transactions with concurrent real-time analytics
- Avoid hot spots
  - On a node
  - An index
  - A key
- Pre-qualify data to be processed in Map Reduce
- Maximize parallelism
- Minimize programmer complexity
- In Realtime
1) Shared Nothing Architecture, every node identical

2) No hotspots – DHT with RIPEMD160

3) Single row ACID – synch replication within cluster

4) Real-time prioritization of transactions + long running tasks

5) Smart Cluster – Zero touch auto fail-over, rebalancing, rolling upgrades..

6) Smart Client - 1 hop to data, no load balancers
Queries + User Defined Functions = Real-time Analytics

User Defined Functions (UDFs)
for real-time analytics and aggregations

STREAM AGGREGATIONS
(INdexed MAP-REDUCE)

Pipe Query results through UDFs
- Filter, Transform, Aggregate... Map, Reduce
Conceptual Stream Processing

- Output of a query is a **Stream**
- Stream flows through
  - Filter
  - Mapper
  - Aggregator
  - Reducer
Hot Analytics Scenario – Airline Late Flights

Data
- Airline flights in the USA January 2012
- 1,050,000 flight records

Task
- On a specific date
  - Which Airline had late flights?
  - How many flights?
  - How many were late?
  - Percentage late flights?

Performance Requirements
- Results in < 1 Sec
- No impact on production transaction performance (300K TPS)

GitHub Repo - https://github.com/aerospike/airline-late-flights-analytics
Solution

- Index the flight records by Date
- Aggregate (Map) late flight data on node
- Reduce flight data from each node in the “client”

**Indexed Map Reduce**

- User Defined Functions (UDFs) written in Lua
- Registered with the Aerospike Cluster
- Invoked as part of a secondary index query
Prepare and execute a Query

```java
/*
 * build the query
 */
Statement stmt = new Statement();
stmt.setNamespace(this.namespace);
stmt.setSetName("FLIGHTS");
stmt.setFilters(Filter.range("FL_DATE_BIN", startTimeStamp, endTimeStamp));
log.info("built query");

// Execute the query
ResultSet rs = client.queryAggregate(null, stmt,
   "simple_aggregation", "late_flights_by_airline");
```
Aggregation Function (Map) function

local function add_values(airlineMap, nextFlight)
  local carrier = nextFlight["CARRIER"]
  local airline = airlineMap[carrier]
  if airline == null then
    airline = map {flights = 0, late = 0}
  end
  airline.flights = airline.flights + 1
  -- if this flight is late, increment the late count in airline
  if toMinutes(nextFlight.ELAPSED_TIME) >
  (toMinutes(nextFlight.ARR_TIME) - toMinutes(nextFlight.DEP_TIME))
  then
    airline.late = airline.late + 1
  end
  -- put the airline into the airlineMap
  airlineMap[carrier] = airline
return airlineMap
end
local function reduce_values(a, b)
    return map.merge(a, b, flightsMerge)
    --return a
end
Stream Function (StreamUDF)

```plaintext
function late_flights_by_airline(stream)

    return stream : aggregate(map(), add_values)
        : reduce(reduce_values)

end
```
Operations (300k TPS) + Analytics (Indexed Map/Reduce)

- Java App calculates % of late flights by Airline
- 300k TPS Operations + Process 1 Million records
  - Indexed Map/Reduce
  - Aggregations
  - Distributed Queries + UDF
- Runs in 0.5 seconds
Operational + Analytics + Adding servers and Re-balancing

- 300k TPS Operations + Process 1 Million records
- Runs in .5 seconds

- Add servers, auto-rebalance while running query

Cluster
- 3 Nodes
  - Hex core 3.4Ghz
  - RAM 24GB
  - 2x Micron p420m SSDs
  - 10GB network
Software

- Aerospike

- Tools
  - Lua Plugin - http://www.eclipse.org/koneksi/ldt/
  - Aerospike Plugin - https://github.com/aerospike/eclipse-tools

- Example
  - Fligtt Analytics - https://github.com/aerospike/flights-analytics
QUESTIONS?

info@aerospike.com

www.aerospike.com