Apache MADlib (Incubating)

Distributed In-Database Machine Learning for Fun and Profit

Frank McQuillan
Jan 31, 2016
Machine learning and distributed systems are just plain **FUN!!!**
Every large commercial enterprise $$$ uses relational databases
Topics

- Journey to Apache
- In-database machine learning
- Making R scalable
Journey to Apache Software Foundation
Journey to Apache

Michael Stonebraker develops Postgres at UCB

Open Source PostgreSQL

Postgres adds support for SQL

PostgreSQL 7.0 released

PostgreSQL 8.0 released

Greenplum forks PostgreSQL

Greenplum open sourced

Hadoop 2.0 Released

Hadoop 1.0 Released

HAWQ & MADlib go Apache

HAWQ launched

MADlib launched

MADlib
History

MADlib project was initiated in 2011 by EMC/Greenplum architects and Joe Hellerstein from Univ. of California, Berkeley.

UrbanDictionary.com:
*mad* (adj.): an adjective used to enhance a noun.
1. dude, you got skills.
2. dude, you got mad skills.
Why Apache?

- Because the ASF is a great place to be!
- Collaborate on software in open and productive ways
- Need strong community for innovation
Pivotal is Committed to Open Source

- Pivotal GemFire → Apache Geode (April 2015)
- Pivotal HDB → Apache HAWQ (Sept 2015)
- MADlib OSS (BSD License) → Apache MADlib (Sept 2015)
- Pivotal Greenplum → Greenplum Database (Oct 2015) (Apache 2 License)
- Pivotal Query Optimizer → gporca, part of Greenplum Database (Jan 2016) (Apache 2 License)
Apache MADlib Overview
## Scalable, In-Database Machine Learning

### Big Data Machine Learning in SQL for Data Scientists

| Open Source, Apache (incubating) | Supports Postgres, Pivotal Greenplum Database, and Pivotal HAWQ | Powerful analytics for Big Data |

- Open Source [https://github.com/apache/incubator-madlib](https://github.com/apache/incubator-madlib)
- Supports Greenplum DB, Apache HAWQ/HDB and PostgreSQL
## Predictive Modeling Library

### Generalized Linear Models
- Linear Regression
- Logistic Regression
- Multinomial Logistic Regression
- Cox Proportional Hazards Regression
- Elastic Net Regularization
- Robust Variance (Huber-White), Clustered Variance, Marginal Effects

### Matrix Factorization
- Singular Value Decomposition (SVD)
- Low Rank

### Linear Systems
- Sparse and Dense Solvers
- Linear Algebra

## Other Machine Learning Algorithms

### Generalized Linear Models
- Linear Regression
- Logistic Regression
- Multinomial Logistic Regression
- Cox Proportional Hazards Regression
- Elastic Net Regularization
- Robust Variance (Huber-White), Clustered Variance, Marginal Effects

### Matrix Factorization
- Singular Value Decomposition (SVD)
- Low Rank

### Linear Systems
- Sparse and Dense Solvers
- Linear Algebra

### Other Machine Learning Algorithms
- Principal Component Analysis (PCA)
- Association Rules (Apriori)
- Topic Modeling (Parallel LDA)
- Decision Trees
- Random Forest
- Support Vector Machines
- Conditional Random Field (CRF)
- Clustering (K-means)
- Cross Validation
- Naïve Bayes
- Support Vector Machines (SVM)

## Time Series
- ARIMA

## Descriptive Statistics

### Sketch-Based Estimators
- CountMin (Cormode-Muth.)
- FM (Flajolet-Martin)
- MFV (Most Frequent Values)

### Correlation and Covariance Summary

## Inferential Statistics

### Hypothesis Tests

## Support Modules

### Array and Matrix Operations
- Sparse Vectors
- Random Sampling
- Probability Functions

### Data Preparation
- PMML Export
- Conjugate Gradient
- Path Functions

---

Jan 2016
MADlib Features

- Better parallelism
  - Algorithms designed to leverage MPP and Hadoop architecture

- Better scalability
  - Algorithms scale as your data set scales

- Better predictive accuracy
  - Can use all data, not a sample

- ASF open source (incubating)
  - Available for customization and optimization
Supported Platforms

Scale-out machine learning on open source, MPP execution engines.

PHD
HDP
Other ODPi distros

GPDB

PostgreSQL

Now open source!
Now open source!
Has always been open source.
Linear Regression on 10 Million Rows in Seconds

Figure 5: Linear regression execution times using MADlib v0.3 on Greenplum Database 4.2.0, 10 million rows

Performance tests are run on a Pivotal Data Computing Appliance (DCA) half-rack for GPDB 4.2.7.1 and a DCA half-rack for HAWQ 1.2.1.0 with 8 nodes and 6 segments per node.
Logistic Regression Scalability

Performance tests are run on a Pivotal Data Computing Appliance (DCA) half-rack for GPDB 4.2.7.1 and a DCA half-rack for HAWQ 1.2.1.0 with 8 nodes and 6 segments per node.
Example Usage

Train a model

```sql
SELECT madlib.linregr_train('houses', 'houses_out', 'price', 'ARRAY[1, tax, bath, size]', 'bedroom')
```

--- Input table
--- Output table
--- Variable to predict
--- Features in data
--- Group data to create
--- multiple models

Predict for new data

```sql
SELECT houses.*, madlib.linregr_predict(ARRAY[1, tax, bath, size], model.coef) as predict
FROM houses_test, houses_out as model;
```

--- Use same features
--- Combine test data
--- and model table
Architecture
Architecture

User Interface

High-Level Iteration Layer (iteration controller)

Functions for Inner Loops (implements ML logic)

Low-level Abstraction Layer (array operations, C++ to DB type-bridge, …)

C API (Greenplum, PostgreSQL, HAWQ)

RDBMS Built-in Functions

SQL

Python

C++
How to Implement Scalability
Example: Linear Regression

- Finding linear dependencies between variables
  \[ y \approx c_0 + c_1 \cdot x_1 + c_2 \cdot x_2 \]
  *i.e., want to find* \( c_1, c_2 \)
Solve Using Ordinary Least Squares

\[ \hat{c} = (X^T X)^{-1} X^T y \]
OLS for Parallel Computation

\[ \hat{c} = (X^T X)^{-1} X^T y \]

\[
\begin{bmatrix}
  a & b \\
  c & d \\
\end{bmatrix}
\]

Segment 1
Segment 2

Pivotal
OLS for Parallel Computation

\[ \hat{c} = (X^T X)^{-1} X^T y \]

Segment 1

Segment 2
OLS for Parallel Computation

\[
\begin{pmatrix}
X^T \\
\end{pmatrix}
\begin{pmatrix}
\begin{array}{cc}
a & c \\
b & d \\
\end{array}
\end{pmatrix}
\begin{pmatrix}
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\end{pmatrix}
\]

\[
\hat{c} = \left( X^T X \right)^{-1} X^T y
\]

Operating across segments increases network traffic
OLS for Parallel Computation

\[
\hat{c} = (X^T X)^{-1} X^T y
\]

Looking at algebra, this is decomposable
OLS for Parallel Computation

\[
\hat{c} = (X^T X)^{-1} X^T y
\]

User outer product for less network traffic
OLS for Parallel Computation

\[ \hat{c} = (X^T X)^{-1} X^T y \]
Do in Single Table Scan

\[
\hat{c} = (X^T X)^{-1} X^T y
\]
Basic Building Block: User-Defined Aggregate

Aggregation phase 1 on each node:
1. Initialize: \((A, b) = (0, 0)\)
2. Transition for all rows:
   \[(A, b) = (A, b) + (x \cdot x^T, x \cdot y)\]
3. Send \((A, b)\)

Aggregation phase 2 on master node:
1. Merge: \((\bar{A}, \bar{b}) = (\bar{A}, \bar{b}) + (A, b)\)
2. Finalize: \(\hat{\beta} = \text{solve}(\bar{A}, \bar{b}) = \bar{A}^{-1} \cdot \bar{b}\)
But not all data scientists speak SQL ...

Making R Scalable
"The preponderance of R and Python usage is more surprising ... two most commonly used individual tools, even above Excel. R and Python are likely popular because they are easily accessible and effective open source tools."

O'Reilly: Strata 2013 Data Science Salary Survey
PivotalR: Bringing MADlib and HAWQ to a Familiar R Interface

- Harness the familiarity of R’s interface and the performance & scalability benefits of in-DB analytics

```r
# Pivotal R

d <- db.data.frame("houses")
houses_linregr <-
  madlib.lm(price ~ tax
  + bath
  + size
  , data=d)

# SQL Code

SELECT madlib.linregr_train( 'houses',
  'houses_linregr',
  'price',
  'ARRAY[1, tax, bath, size]');
```
PivotalR Design Overview

- Call MADlib’s in-DB machine learning functions directly from R
- Syntax is analogous to native R function

1. R \(\rightarrow\) SQL

2. SQL to execute

3. Computation results

- Data doesn’t need to leave the database
- All heavy lifting, including model estimation & computation, are done in the database
- Only strings of SQL and model output transferred across DBI

No data here

Data lives here
What’s Coming Up?
Upcoming Release (1.9)

**Predictive Models**
- Support vector machines including non-linear kernel (Gaussian, polynomial)

**Utilities**
- Matrix operations (phase 2)
- Path functions (phase 1)
- Stemming

**Descriptive Stats**
- Covariance matrix
Potential Future Features*

**Predictive Models**
- Mixed effects models
- Time series models
- Parameter weights
- Graph models
- Connected components
- Linkage operations

**Utilities**
- Path functions (phase 2)
- Pivoting
- Anonymization
- Sessionization
- Prediction metrics
- URI tools
- Stratified sampling

**Usability**
- Refresh interface for 2.0
- Python API

* Subject to community interest
Please Join Us!

- Web sites
  - http://madlib.incubator.apache.org/
  - https://cwiki.apache.org/confluence/display
  - https://cran.r-project.org/web/packages/PivotalR/index.html

- Github
  - https://github.com/apache/incubator-madlib
  - https://github.com/pivotalsoftware/PivotalR

- Mailing lists
  - dev@madlib.incubator.apache.org
  - user@madlib.incubator.apache.org