Greenplum 助力科学计算

马丽丽 2017.8.23

Outline

Greenplum Architecture

Greenplum supports Data Science

Data Science Bundle for Python & R

When PL meets container

Q&A

About Pivotal

Founded April 2013

2000+ Employees | 1000+ Customers

Spun out from EMC & VMware

Big Data

Pivotal
Big Data Suite

- Data Warehouse, SQL-on-Hadoop and In-Memory Data Grid

Cloud

Pivotal
Cloud Foundry

- Platform-as-a-Service (PaaS) software with multi-cloud support

Agile Development

Pivotal Labs

- World-class application development services

Pivotal Big Data Suite (BDS)

Pivotal Big Data Suite
Open Source data management portfolio

PIVOTAL GREENPLUM DATABASE
Data warehouse based on open source Greenplum Database

PIVOTAL HDB
Advanced analytic SQL database for Hadoop, based on open source Apache HAWQ

PIVOTAL GEMFIRE
High-performance in-memory data grid based on Apache Geode

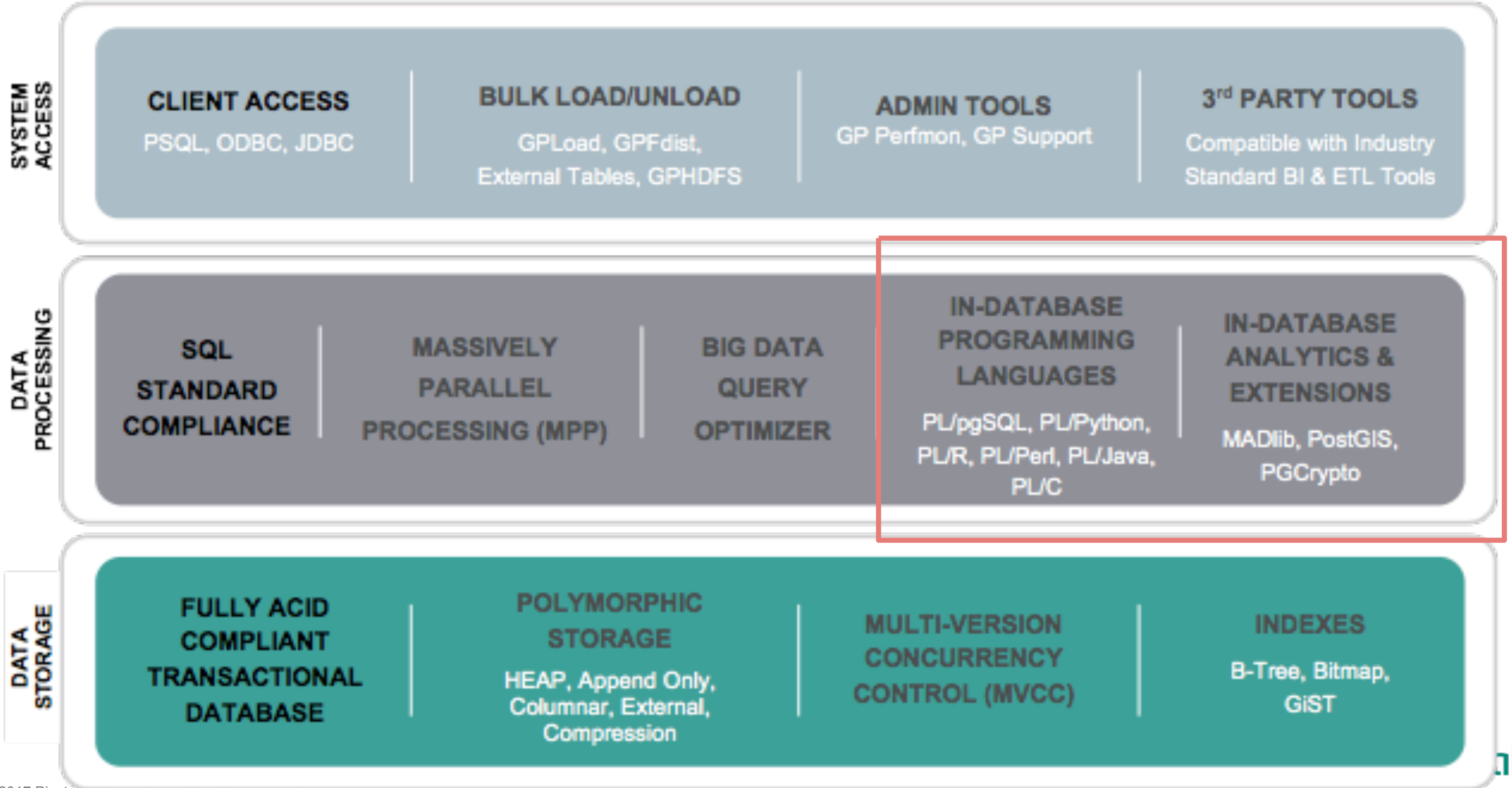
✓ Complete Data platform

✓ Based on open source

✓ Flexible licensing

✓ Advanced data services

Greenplum Architecture



SYSTEM ACCESS

- **CLIENT ACCESS** PSQL, ODBC, JDBC
- **BULK LOAD/UNLOAD** GPLoad, GPFdist, External Tables, GPHDFS
- **ADMIN TOOLS** GP Perfmon, GP Support
- **3rd PARTY TOOLS** Compatible with Industry Standard BI & ETL Tools

DATA PROCESSING

- **SQL STANDARD COMPLIANCE**
- **MASSIVELY PARALLEL PROCESSING (MPP)**
- **BIG DATA QUERY OPTIMIZER**
- **IN-DATABASE PROGRAMMING LANGUAGES** PL/pgSQL, PL/Python, PL/R, PL/Perl, PL/Java, PL/C
- **IN-DATABASE ANALYTICS & EXTENSIONS** MADlib, PostGIS, PGCrypto

DATA STORAGE

- **FULLY ACID COMPLIANT TRANSACTIONAL DATABASE**
- **POLYMORPHIC STORAGE** HEAP, Append Only, Columnar, External, Compression
- **MULTI-VERSION CONCURRENCY CONTROL (MVCC)**
- **INDEXES** B-Tree, Bitmap, GiST

Outline

Greenplum Architecture

Greenplum supports Data Science

Data Science Bundle for Python & R

When PL meets container

Q&A

Greenplum Support on Data Science

- Apache^{TM} MADlib® (incubating)
- GPText
- PL/Python
- PL/R

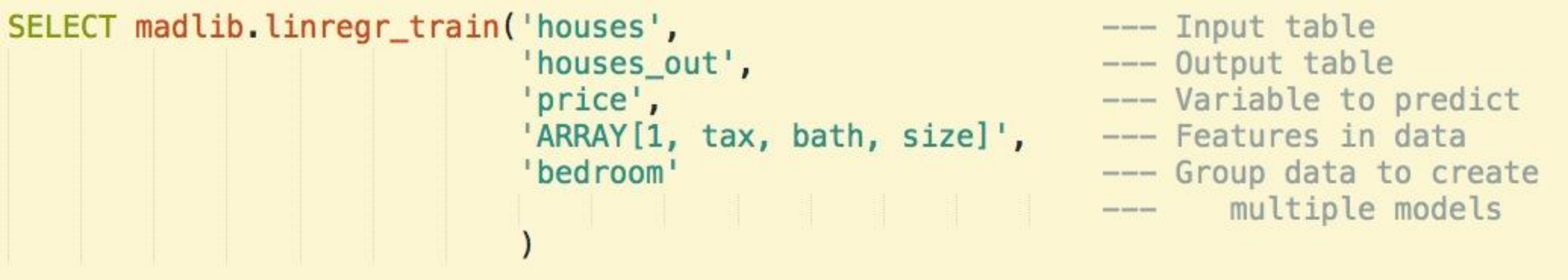
Apache MADlib: In-Database Machine Learning

- **Apache™ MADlib® (incubating)** is an open-source library for scalable in-database analytics
- Provides parallel implementations of mathematical, statistical and machine learning methods for structured and unstructured data
- Supports Apache HAWQ, Greenplum Database and Postgres
- Analytics on all data in-database, without sampling (produces more accurate results, less effort)

http://madlib.incubator.apache.org

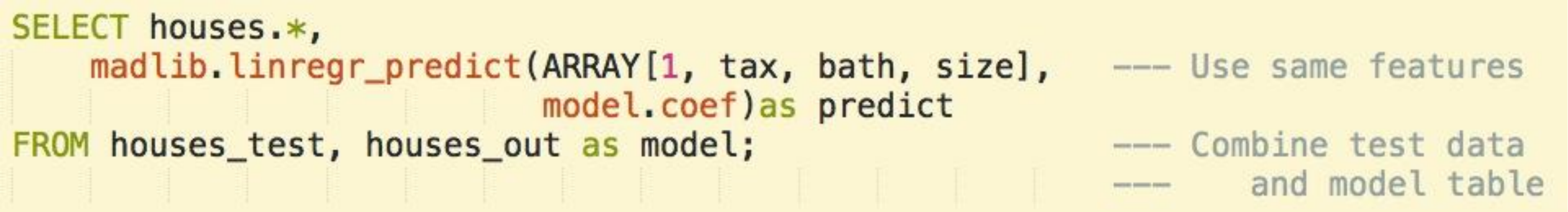
MADlib: SQL-Based Machine Learning

Train a model



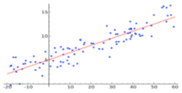
```
SELECT madlib.linregr_train('houses',                              --- Input table
                            'houses_out',                          --- Output table
                            'price',                               --- Variable to predict
                            'ARRAY[1, tax, bath, size]',           --- Features in data
                            'bedroom'                              --- Group data to create
                                                                   ---    multiple models
                            )
```

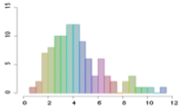
Predict for new data

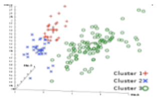


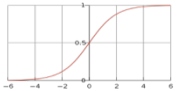
```
SELECT houses.*,
    madlib.linregr_predict(ARRAY[1, tax, bath, size],       --- Use same features
                           model.coef)as predict
FROM houses_test, houses_out as model;                      --- Combine test data
                                                            ---    and model table
```

MADlib Functions









Generalized Linear Models

- Linear Regression
- Logistic Regression
- Multinomial Logistic Regression
- Ordinal 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

Path Functions

- Operations on Pattern Matches

Descriptive Statistics

Sketch-Based Estimators

- CountMin (Cormode-Muth.)
- FM (Flajolet-Martin)
- MFV (Most Frequent Values)

Correlation and Covariance

Summary

Inferential Statistics

Hypothesis Tests

Utility Modules

Array and Matrix Operations

Sparse Vectors

Random Sampling

Probability Functions

Data Preparation

PMML Export

Conjugate Gradient

Stemming

GPText

- Combine with Solr
- Provide solid text analysis and index function
- Computing distributed in segment, can be run simultaneously
- Combine SQL and text analysis together

Procedural Language: PL/Python

-
  ```
  CREATE TABLE sales (id int, year int, qtr int, day int, region text)  DISTRIBUTED BY (id) ;
  INSERT INTO sales VALUES
   (1, 2014, 1,1, 'usa'),
   (2, 2002, 2,2, 'europe'),
   (3, 2014, 3,3, 'asia'),
   (4, 2014, 4,4, 'usa'),
   (5, 2014, 1,5, 'europe'),
   (6, 2014, 2,6, 'asia'),
   (7, 2002, 3,7, 'usa') ;
  ```

-
  ```
  CREATE OR REPLACE FUNCTION mypytest(a integer)
    RETURNS text
  AS $$
    rv = plpy.execute("SELECT * FROM sales ORDER BY id", 5)
    region = rv[a]["region"]
    return region
  $$ language plpythonu;
  ```

-
  ```
  SELECT mypytest(2) ;
  ```

Procedural Language: PL/R

-
  ```
  CREATE OR REPLACE FUNCTION r_norm(n integer, mean float8,
    std_dev float8) RETURNS float8[ ] AS
  $$
    x<-rnorm(n,mean,std_dev)
    return(x)
  $$
  LANGUAGE 'plr';
  ```

-
  ```
  CREATE TABLE test_norm_var
    AS SELECT id, r_norm(10,0,1) as x
    FROM (SELECT generate_series(1,30:: bigint) AS ID) foo
    DISTRIBUTED BY (id);
  ```

Outline

Greenplum Architecture

Greenplum supports Data Science

Data Science Bundle for Python & R

When PL meets container

Q&A

Procedural Language -- Pain Point

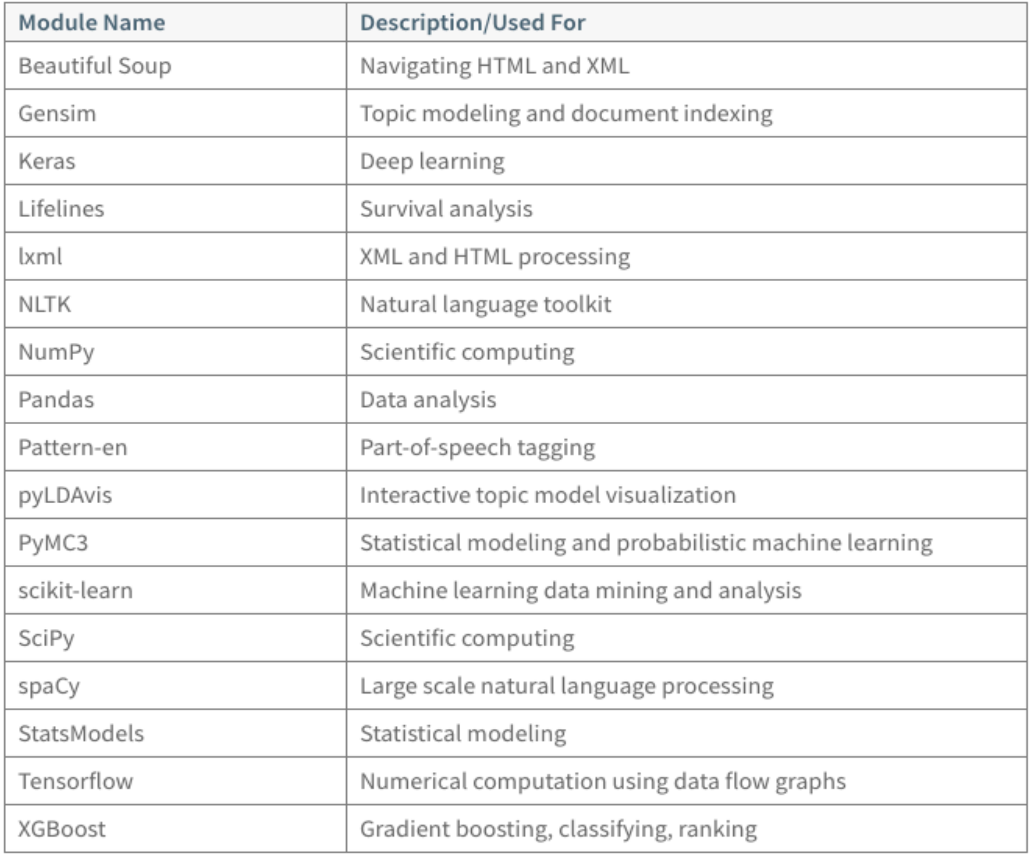
- **Need install third-party Python/R binaries before using**

- Unsecure Execution Environment for Python and R

  Normal user does not have ability to create function in untrusted language

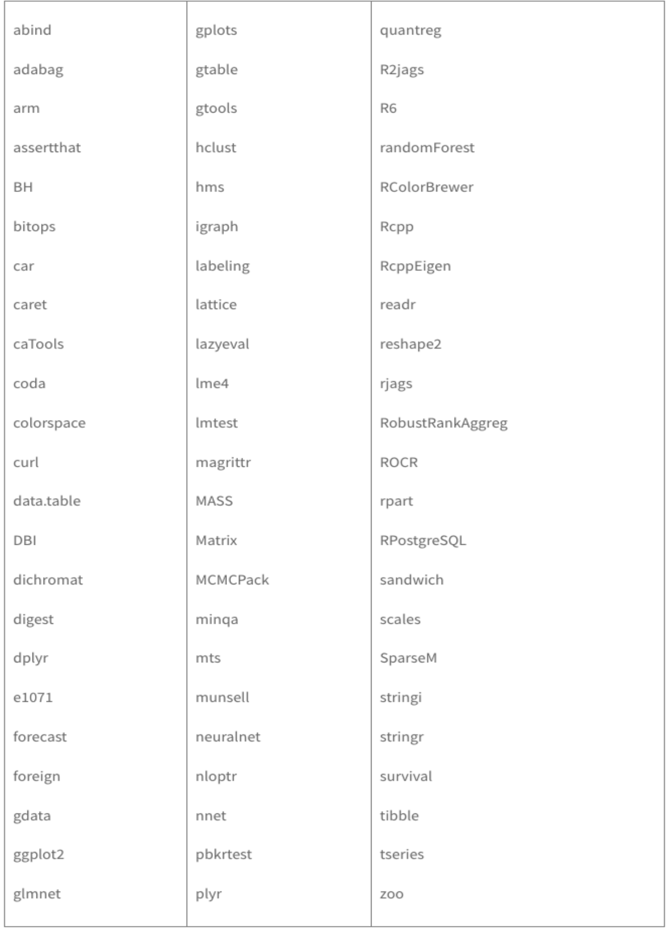
  Function failure may cause postgres process restart

Data Science Bundle for Python



Module Name	Description/Used For
Beautiful Soup	Navigating HTML and XML
Gensim	Topic modeling and document indexing
Keras	Deep learning
Lifelines	Survival analysis
lxml	XML and HTML processing
NLTK	Natural language toolkit
NumPy	Scientific computing
Pandas	Data analysis
Pattern-en	Part-of-speech tagging
pyLDAvis	Interactive topic model visualization
PyMC3	Statistical modeling and probabilistic machine learning
scikit-learn	Machine learning data mining and analysis
SciPy	Scientific computing
spaCy	Large scale natural language processing
StatsModels	Statistical modeling
Tensorflow	Numerical computation using data flow graphs
XGBoost	Gradient boosting, classifying, ranking

Data Science Bundle for R



- abind
- adabag
- arm
- assertthat
- BH
- bitops
- car
- caret
- caTools
- coda
- colorspace
- curl
- data.table
- DBI
- dichromat
- digest
- dplyr
- e1071
- forecast
- foreign
- gdata
- ggplot2
- glmnet
- gplots
- gtable
- gtools
- hclust
- hms
- igraph
- labeling
- lattice
- lazyeval
- lme4
- lmtest
- magrittr
- MASS
- Matrix
- MCMCPack
- minqa
- mts
- munsell
- neuralnet
- nloptr
- nnet
- pbkrtest
- plyr
- quantreg
- R2jags
- R6
- randomForest
- RColorBrewer
- Rcpp
- RcppEigen
- readr
- reshape2
- rjags
- RobustRankAggreg
- ROCR
- rpart
- RPostgreSQL
- sandwich
- scales
- SparseM
- stringi
- stringr
- survival
- tibble
- tseries
- zoo

Case: GP + Tensorflow for Linear Regression

- Table:

T

Two columns: col1 & col2

Linear dependency: col2 = w*col1 + b

- We want to infer the relationship between the two columns

Select tfTrain(agg_train(col1),

agg_train(col2)) from test;

col1	col2
1	0.4
2	0.5
5	0.8
...	...

UDA Part

```
create function sfunc_train(state float[], a float)
returns float[] as
$$
state.append(a)
return state
$$ language plpythonu;

create aggregate agg_train(float)
(
sfunc=sfunc_train,
stype=float[],
initcond=‘{}’
)
```

TF Part

```
create function tfTrain(x_data float[], y_data float[])
returns numeric[] as
$$
import tensorflow as tf
import numpy as np

W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))

y = W * x_data + b

loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)
for step in range(201):
    sess.run(train)
return np.append(sess.run(W)[0], sess.run(b)[0])
$$ language plpythonu;
```

Outline

Greenplum Architecture

Greenplum supports Data Science

Data Science Bundle for Python & R

When PL meets container

Q&A

Motivation

- Simply the process of developing functions for python/R
- Secure environment for PL/Python and PL/R
  - Code in python or R should not be able to modify data files on local disk, including database file, configuration file, or directory.
  - Code in python or R should not be able to connect to local database using gpadmin from localhost.
- Isolation. Independent execution
  - Failure in PL does not affect running QE process and postmaster process.
  - PL running does not change share memory of QE process.
- Flexibility. Users have the flexibility to configure their own running environment, for example, python version
- Performance. Performance should not be impacted so much

Goal

- Implement a secure execution environment, normal user can create their own Python/R function
- Function run on same host as QE, or dedicated computing environment
- Function running failure does not affect other processes on segment postgres
- Performance controlled in 2X times slow-down compared with untrusted language
- Container lifecycle consistent with QE
- Basic debug information can be gathered from container

Container Benefits

- Independent Namespace
- Isolated Execution Environment
- Controllable Resource Occupation
- Easy to scale

Usage

- Install & Configure
  - Embedded in GPDB new binaries
  - Supporting Docker Image: Pivotal provided & User self-defined
  - Single Script for configuring
    - User just need have docker environment in GPDB cluster
    - One simple script including following functions:
      - plcontainer image --install $imageFile $hostFile installing images
      - plcontainer image --configure $ImageName $ImageFile configuring images to Language recognizable format

Usage - cont.

- Create language

```
Create LANGUAGE plcontainer;
```

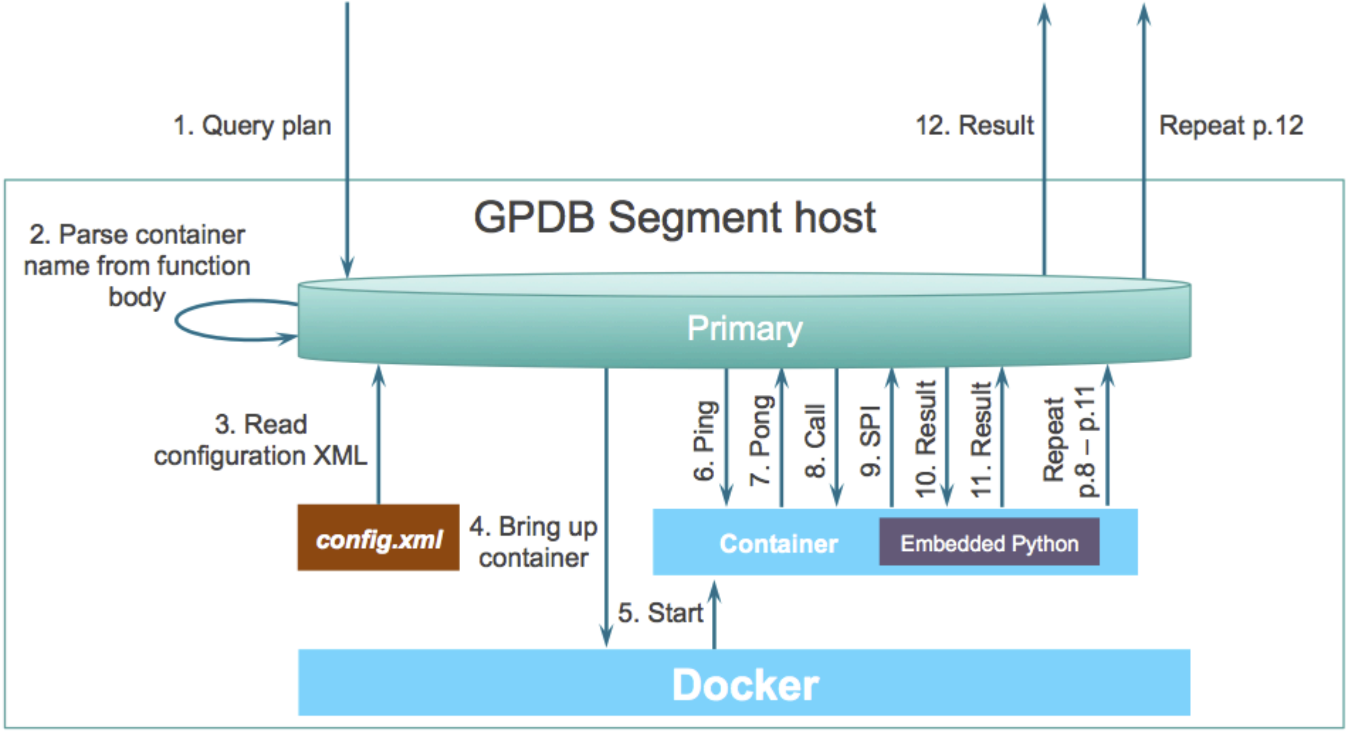
- Create function

```
CREATE OR REPLACE FUNCTION pylog100() RETURNS double precision AS $$
# container: plc_python
import math
return math.log10(100)
$$ LANGUAGE plcontainer;
```

- Execute Function

```
Select pylog(100);
```

Architecture & Flow



Future

- Function run not bundled with QE
- More mechanism for secure environment support, i.g, Garden, separate process
- Contribute back to Postgres community

We’re Hiring

- We’re hiring Product Manager
- Please contact sgao@pivotal.io

Q&A

Pivotal®