Pivotal

Petabyte Scale Data Warehousing Greenplum

PLContainer --Trusting Untrusted Languages

Postgres Conf 2018

Marshall Presser
Craig Sylvester
Andreas Scherbaum
17 April 2018

Agenda

- The Problem
- What is PL/Container
- How to use PL/Container
- PL/Container Internals
- Future Work
- Q+A

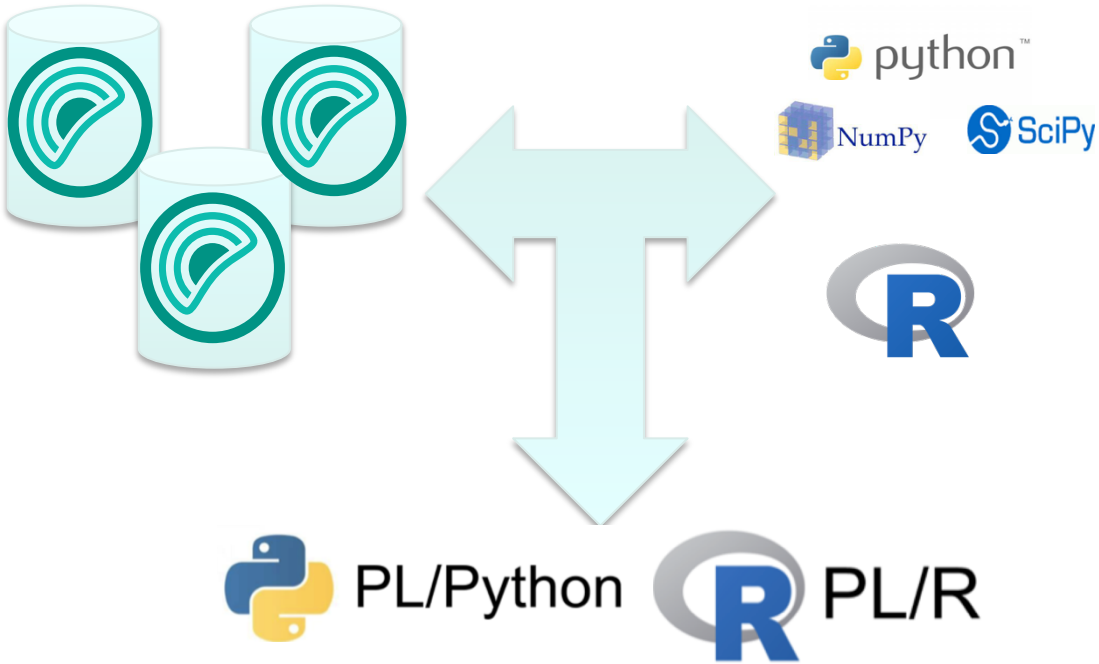
Pivotal
Greenplum Database

PL/Python

PL/R

PL/Container

We want to analyse data IN Database



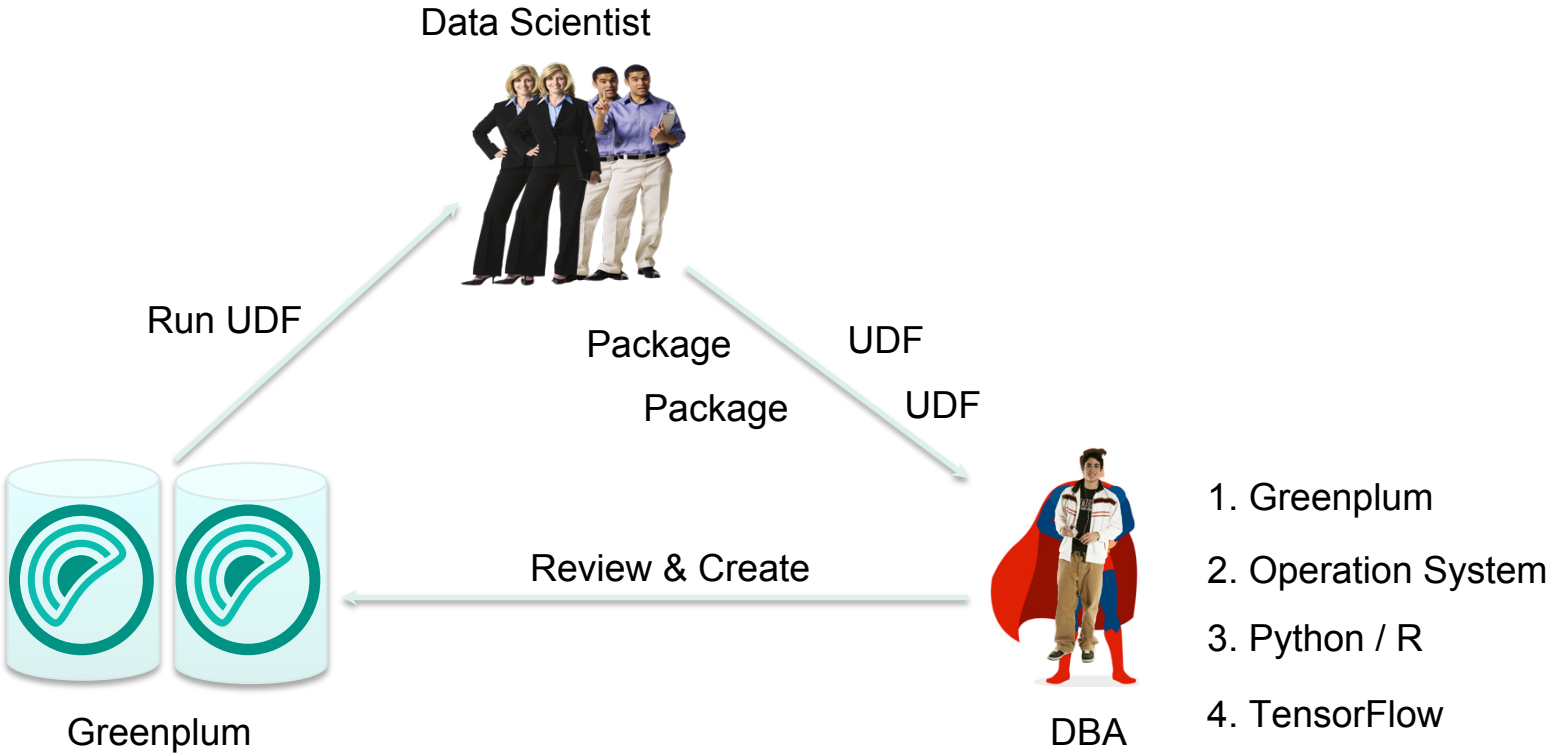
BUT …

PL/Python and PL/R are UNTRUSTED Languages

Only Superuser can Create UDF in Untrusted Languages

System(“rm -rf /data”)

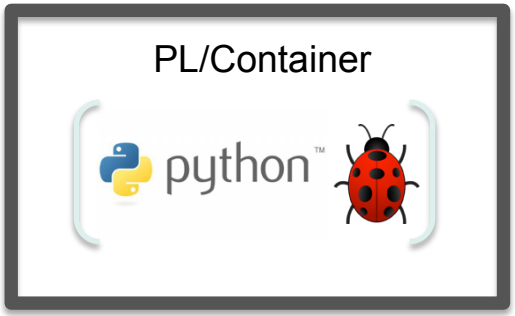
The Problem: Triangle Dependency

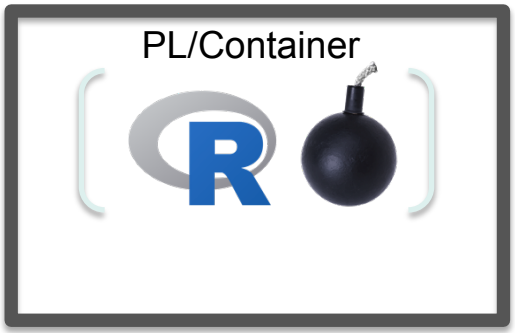


What is PL/Container?

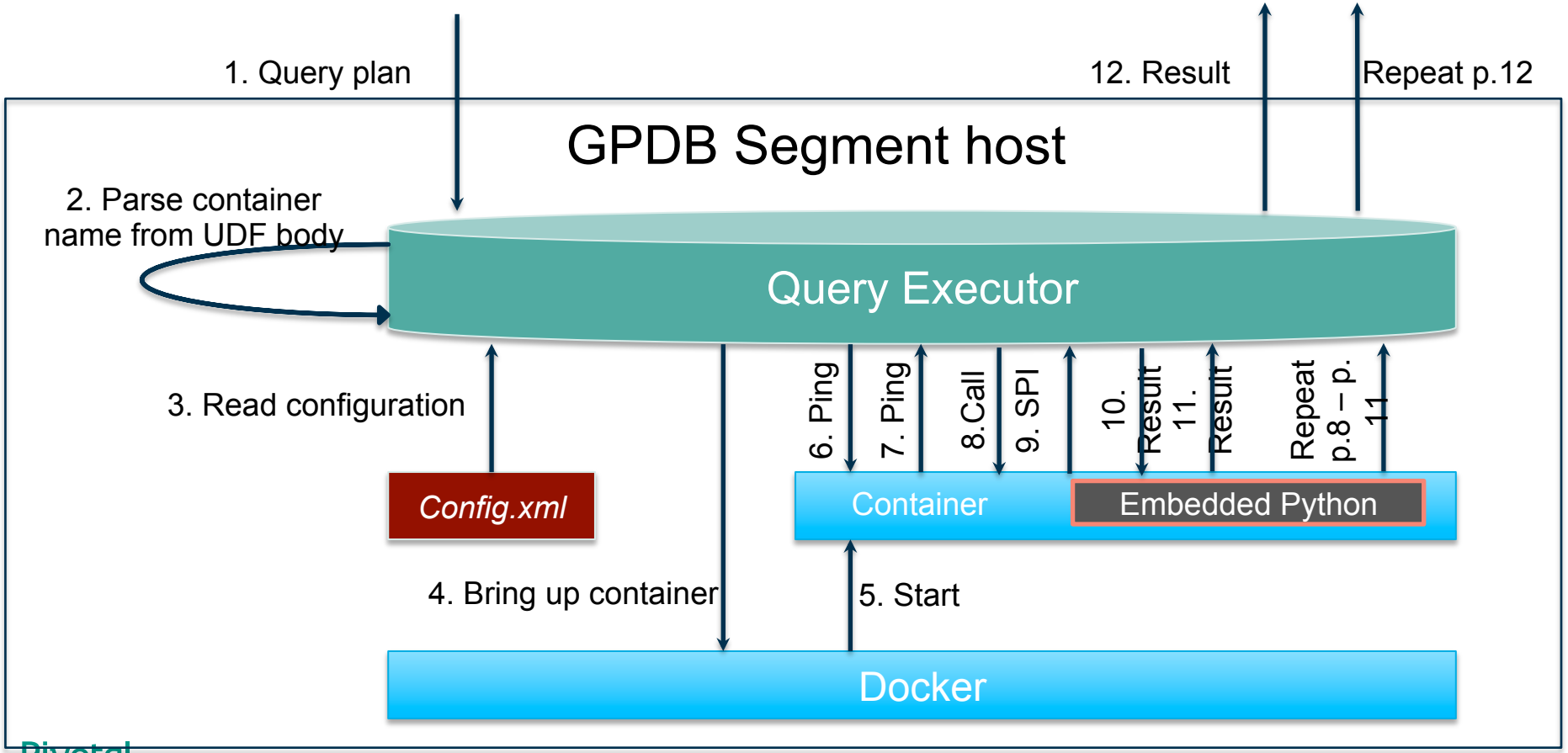
PL/Container is a customizable, secure runtime for Greenplum Database Procedural Languages.

- Greenplum Database Extension
- Stateless
- Based on Docker Container
- Customizable
- Secure
- Isolated





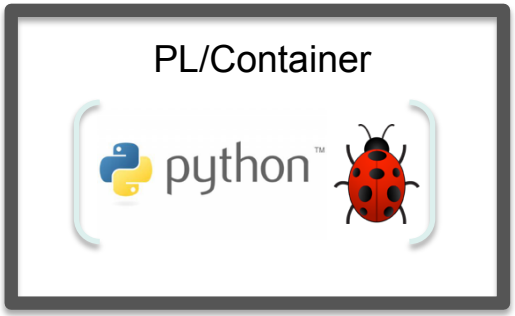
PL/Container Architecture

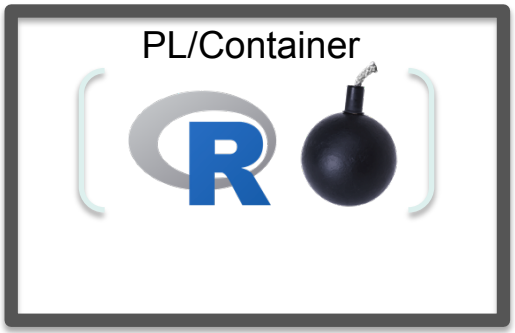


How UDF run in PL/Container?

PL/Container is a customizable, secure runtime for Greenplum Database Procedural Languages.

a. PL/Container Extension starts a docker container (only in 1st call)
b. Transfer UDF and data to docker container
c. Run the UDF in docker container
d. Contact the docker container to get the results





Install PL/Container on Greenplum

Install from Source Code

- source $GPHOME/greenplum_path.sh
- make install

Install from GPPKG

- gppkg –i plcontianer-1.1.0-rhel7-x86_64.gppkg
- no additional dependencies

Prerequisites

Platform

Centos 6.6+ or 7.x

Database

Greenplum 5.2+

Docker

Docker 17.05+ on Centos7

Docker 1.7+ on Centos6

Build Custom Docker Image (optional)

Minimum Requirement:

- Python or R environment
- Add location of libpython.so and libR.so to LD_LIBRARY_PATH

```
FROM continuumio/anaconda

ENV LD_LIBRARY_PATH "/opt/conda/lib:$LD_LIBRARY_PATH”
```

Customize Your image:

- Install specific packages

```
FROM continuumio/anaconda3

RUN conda install -c conda-forge -y tensorflow
ENV LD_LIBRARY_PATH "/opt/conda/lib:/usr/local/lib:$LD_LIBRARY_PATH"
```

Configure PL/Container

XML

runtime

- **<id>**
- **<image>**
- **<command>**
- **<shared directory>**
- **<setting memory_mb>**
- **<setting cpu_share>**

container cgroup node
memory.memsw.limit_in_bytes
cpu.shares

Image

image add

image delete

image list

Runtime

runtime add
runtime delete

runtime backup
runtime restore

runtime edit
runtime show

Run PL/Container

Running a simple plpython UDF to calculate $\log_{10}$

```
postgres=# CREATE LANGUAGE plpythonu; plcontainer;

postgres=# CREATE OR REPLACE FUNCTION pylog10(input double precision) RETURNS double

precision AS $$

import math

return math.log10(input)

$$ LANGUAGE plpythonu;

postgres=# SELECT pylog10(100);
  pylog10
----------
         2
(1 row)
```

Run PL/Container

Running a simple PL/Container UDF to calculate $\log_{10}$

```
postgres=# CREATE EXTENSION plcontainer;

postgres=# CREATE OR REPLACE FUNCTION pylog10(Input double precision) RETURNS double
precision AS $$
# container: plc_python_shared
import math
return math.log10(input)
$$ LANGUAGE plcontainer;

postgres=# SELECT pylog10(100);
  pylog10
----------
         2
(1 row)
```

Pivotal

Performance

Test Environment

- Hardware: 6 virtual machines, each with 19G memory and 5 processors. (Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz)
- Software: Centos7, GPDB 5.2 with 30 segments.

Workloads

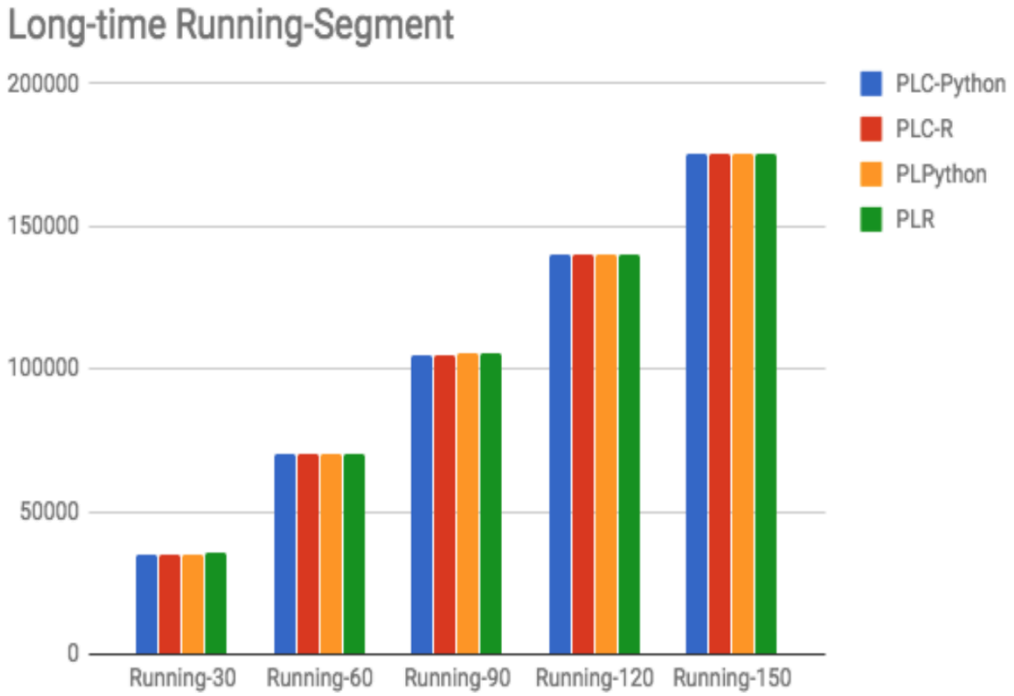
- Long-running function
- Large input array function
- Large output array function

Performance

Long-running function

```
CREATE OR REPLACE FUNCTION pysleep(i int
RETURNS void AS $$
# contaziner: plc_python_shared
import time
time.sleep(i)
$$ LANGUAGE plcontainer;

SELECT count(pysleep(1)) FROM tbl;
```



no performance downgrade

Transforming How The World Builds Software