PL/Parrot and PL/Perl6

Parrots and Butterflies in your Database

Jonathan "Duke" Leto

Parrot Virtual Machine

- Process (Application) Virtual Machine
- Register-based
- Continuation Passing Style
- Design Goals
  - Pluggable
  - Interoperable
  - Dynamic
- 2.7.0 "Australian King" just released
- Undergoing lots of optimizations

Rakudo Perl 6

- Most active implementation of Perl 6
- Implements $\sim$ 80% of the spec
- Currently uses Parrot as a backend, but plans to support others
- Stable release series is Rakudo Star, dev release every month
- Getting faster do to optimizations in Parrot

Why Embed Parrot VM in PostgreSQL?

- Procedural/PostgreSQL Languages (PL's) are hard to write and maintain
- PL/Parrot does the hard work, HLLs benefit with much less effort
- Platform independent, fast†, stored procedures

PL/Parrot Timeline

- PostgreSQL came from Ingres, started in the mid 1970's
- Parrot started around 2001
- Empty pgFoundry repo created 2006 by David Fetter/Joshua Tolley
- GitHub repo created Oct 2009 by Duke Leto
- PL/Perl6 started to work June 2010
- What next?

Current Features

- PL/PIR and PL/Perl6
- Pass and return basic datatypes
- Basic security model (Don't do that)
- Growing Test Suite
- Enthusiastic and friendly community
- Coolest Feature: Use Perl 6 grammars in PostgreSQL!

Things that creak

- Documentation - http://pl.parrot.org
- SPI - branch being worked by cxreg++, elog works!
- Triggers
- SETOF - branch with some tests
- Parrot Bugs
  - IMCC Syntax Errors
  - Security API
  - Extend/Embed API

Installing/Testing PL/Parrot

- Install Rakudo or Parrot
- # parrot_config needs to be in your $PATH
- One of:
- git clone git://github.com/leto/plparrot.git
- wget http://icanhaz.com/plparrot0.20
- cd plparrot
- export PGPORT=5555 # if necessary
- make install installcheck # might need sudo
- make test test_plperl6 # PL/PIR + PL/Perl6 tests

PL/PIR Example 1

```
CREATE OR REPLACE FUNCTION
 pir_concat(text,text,float)
RETURNS varchar LANGUAGE plpir AS $$
      .param string s1
      .param string s2
      .param num x
  if x < 0 goto backward
      $S1 = s1 . s2
      goto done
  backward:
      $S1 = s2 . s1
  done:
      .return($S1)
$$;
```

PL/Perl6 Example 1

```
CREATE OR REPLACE FUNCTION fibonacci_sum(int)
RETURNS int LANGUAGE plperl6 AS $$
{
    [+] (1, 1, *+* ... $^limit)
}
$$;
```

PL/Perl6 Example 2

```
CREATE OR REPLACE FUNCTION is_inventory(text) RETURNS integer
LANGUAGE plperl6 AS $q$
($item) {
    # This grammar needs a 'my' because the default
    # is 'our' i.e. package scope
    my grammar Inventory {
        regex product     { \d+ }
        regex quantity    { \d+ }
        regex color       { \S+ }
        regex description { \N* }
        rule TOP { ^^ <product> <quantity>
                   [
                   | <description> '(' \s* <color> \s*  ')'
                   | <color> <description>
                   ]
                   $$
        }
    }
    return ?Inventory.parse($item);
}
$q$;
```

How is PL/Parrot Sausage Made?

Future Goals

- Easy onramps to add new languages to PL/Parrot
- Framework for DSL's
- Allow various PL's to communicate
- Freeze/thaw subtransaction-level states

Get involved!

- Try PL/Parrot on your system and submit detailed bug reports
- Fork on github and hack on stuff!
- Help with GitHub Issues
- #plparrot on freenode
- http://pl.parrot.org
- http://groups.google.com/group/plparrot

Thanks

- PL/Parrot team:
  - David Fetter, David E. Wheeler, Joshua Tolley, Daniel Arbelo Arrocha, Dave Olszewski
  - AKA davidfetter++, theory++, eggyknap++, darbelo++, cxreg++
- Everyone working on Parrot Virtual Machine, Perl 6 and PostgreSQL
- Especially (Moritz Lenz) moritz++, (Peter Lobsinger) plobsing++ and (Stephen O'Rear) sorear++, for great advice and help

Resources

- http://pl.parrot.org
- http://github.com/leto/plparrot
- http://parrot.org
- @parrotvm / !parrot on twitter/identi.ca
- @dukeleto / @leto on twitter/identi.ca