2ndQuadrant®
PostgreSQL

Desbravando PG Hooks

Rafael Castro

Sobre mim...

- Desenvolvedor (Python, C, …)
- DBA PostgreSQL
- Criador e mantenedor da ferramenta OmniDB

Sumário

- Introdução a Hooks
- Lista de Hooks
- Exemplos Úteis
- Depurador PL/pgSQL com Hooks
- Plus: Background Workers

Como customizar o PG?

Como customizar o PG?

- Funções / Procedimentos

Como customizar o PG?

- Funções / Procedimentos
- Tipos

Como customizar o PG?

- Funções / Procedimentos
- Tipos
- Operadores

Como customizar o PG?

- Funções / Procedimentos
- Tipos
- Operadores
- Agregadores

Como customizar o PG?

- Funções / Procedimentos
- Tipos
- Operadores
- Agregadores
- Extensões

Como customizar o PG?

- Funções / Procedimentos
- Tipos
- Operadores
- Agregadores
- Extensões
- **Ganchos (Hooks)**

Como customizar o PG?

- Funções / Procedimentos
- Tipos
- Operadores
- Agregadores
- Extensões
- **Ganchos (Hooks)**
- **BG Workers**

Hooks - Introdução

Hooks - Introdução

- Extender / Modificar / Interromper operações

Hooks - Introdução

- Extender / Modificar / Interromper operações
- Biblioteca externa escrita em C

Hooks - Introdução

- Extender / Modificar / Interromper operações
- Biblioteca externa escrita em C
- Carregada com *“shared_preload_libraries”*

Hooks - Implementação

Hooks - Implementação

- Ponteiros *NULL* espalhados pelo código-fonte do PG

src/backend/commands/user.c

```
47
48  /* Hook to check passwords in CreateRole() and AlterRole() */
49  check_password_hook_type check_password_hook = NULL;
50
```

Hooks - Implementação

- Bibliotecas apontam ponteiros para funções específicas

```
check_password_hook = custom_check_password;

static void
custom_check_password(char *username,
                      char *password,
                      int password_type,
                      Datum validuntil_time,
                      bool validuntil_null)
{
  ...
}
```

Hooks - Implementação

- Função _PG_init() chamada quando PG inicia

```
void _PG_init(void)
{
  check_password_hook = custom_check_password;
}
```

Hooks - Implementação

- PG_MODULE_MAGIC - Checagem de compatibilidade

```
#include "postgres.h"

#include <limits.h>

#include "access/parallel.h"
#include "commands/explain.h"
#include "executor/instrument.h"
#include "jit/jit.h"
#include "utils/guc.h"

#ifdef PG_MODULE_MAGIC
PG_MODULE_MAGIC;
#endif
```

Hooks - Implementação

- Função _PG_fini() chamada quando PG é desligado

```
void _PG_fini(void)
{
  check_password_hook = NULL;
}
```

Hooks - Boas práticas

- Bibliotecas devem coexistir em harmonia (encadeamento)

```
static check_password_hook_type prev_check_password_hook = NULL;

void _PG_init(void)
{
  prev_check_password_hook = check_password_hook;
  check_password_hook = custom_check_password;
}

void _PG_fini(void)
{
  check_password_hook = prev_check_password_hook;
}
```

Hooks - Boas práticas

- Bibliotecas devem coexistir em harmonia

```
static void
custom_check_password(char *username,
                      char *password,
                      int password_type,
                      Datum validuntil_time,
                      bool validuntil_null)
{
  ...

  if (prev_check_password_hook)
        (*prev_check_password_hook) (username, password,
                                     password_type, validuntil_time,
                                     validuntil_null);
}
```

Hooks - Boas práticas

- Hooks que substituem devem chamar função original

```
static void custom_process_utility(Node *parsetree, const char *queryString,
    ParamListInfo params, DestReceiver *dest,
    char *completionTag, ProcessUtilityContext context)
{
  ...
  if (prev_utility_hook)
        (*prev_utility_hook) (parsetree, queryString,
                              context, params,
                              dest, completionTag);
  else
    standard_ProcessUtility(parsetree, queryString,
                            context, params,
                            dest, completionTag);
}
```

Tipos de Hooks

Tipos de Hooks

- Gerais
  - *emit_log_hook*: Chamado antes de escrever linha de log
  - *shmem_startup_hook*: Chamado após PG criar região de memória compartilhada

Tipos de Hooks

- Segurança
  - *password_check_hook*: Chamado ao executar CREATE ROLE or ALTER ROLE (para mudar senha)
  - *ClientAuthentication_hook*: Chamado após PG realizar autenticação de usuário (e antes de retornar ao cliente)

Tipos de Hooks

- Gerenciamento de funções
  - *fmgr_hook*: Chamado antes e logo depois de uma função ser executada (chamado 2x)

Tipos de Hooks

- Planejador
  - *explain_get_index_name_hook*: Chamado durante explain, para alterar o mecanismo de criação de nomes de índices fictícios

Tipos de Hooks

- Execução
  - *ExecutorStart_hook*: Chamado antes da execução de um planejamento de query
  - *ProcessUtility_hook*: Chamado antes da execução de operações do *Process Utility*, responsável pela execução de DDL

Tipos de Hooks

- PL/pgSQL
  - *func_beg*: Chamado antes da execução de uma função PL/pgSQL
  - *stmt_beg*: Chamado antes da execução de uma linha de comando do corpo de uma função PL/pgSQL

Exemplos

- *Contrib: passwordcheck*

Exemplos

- *Michael Paquier: Superuser restrictions*

Exemplos

- *Contrib: auto explain*

Depurador PL/pgSQL

Depurador PL/pgSQL

- Depurar funções PL/pgSQL utilizando interface gráfica

Depurador PL/pgSQL

- Depurar funções PL/pgSQL utilizando interface gráfica
- Execução passo a passo

Depurador PL/pgSQL

- Depurar funções PL/pgSQL utilizando interface gráfica
- Execução passo a passo
- Inserir *Breakpoints*

Depurador PL/pgSQL

- Depurar funções PL/pgSQL utilizando interface gráfica
- Execução passo a passo
- Inserir *Breakpoints*
- Visualizar variáveis

Depurador PL/pgSQL

- Depurar funções PL/pgSQL utilizando interface gráfica
- Execução passo a passo
- Inserir *Breakpoints*
- Visualizar variáveis
- Estatísticas de execução

Depurador - Implementação

- Utiliza-se Hooks PL/pgSQL
  - *func_setup*
  - *func_beg*
  - *func_end*
  - *stmt_beg*
  - *stmt_end*

Depurador - Implementação

- Cliente 1 (PG backend)

Depurador - Implementação

- Cliente 1 (PG backend)
  - Executa função a ser depurada

Depurador - Implementação

- Cliente 1 (PG backend)
  - Executa função a ser depurada
  - Pausa execução de cada linha com os Hooks

Depurador - Implementação

- Cliente 1 (PG backend)
  - Executa função a ser depurada
  - Pausa execução de cada linha com os Hooks
  - Controla tabela de estatísticas e de variáveis

Depurador - Implementação

- Cliente 1 (PG backend)
  - Executa função a ser depurada
  - Pausa execução de cada linha com os Hooks
  - Controla tabela de estatísticas e de variáveis
- Cliente 2 (Interface Gráfica)

Depurador - Implementação

- Cliente 1 (PG backend)
  - Executa função a ser depurada
  - Pausa execução de cada linha com os Hooks
  - Controla tabela de estatísticas e de variáveis
- Cliente 2 (Interface Gráfica)
  - Se comunica com Cliente 1

Depurador - Implementação

- Cliente 1 (PG backend)
  - Executa função a ser depurada
  - Pausa execução de cada linha com os Hooks
  - Controla tabela de estatísticas e de variáveis
- Cliente 2 (Interface Gráfica)
  - Se comunica com Cliente 1
  - Controla pausa/prosseguimento da execução

Depurador - Implementação

- Cliente 1 (PG backend)
  - Executa função a ser depurada
  - Pausa execução de cada linha com os Hooks
  - Controla tabela de estatísticas e de variáveis
- Cliente 2 (Interface Gráfica)
  - Se comunica com Cliente 1
  - Controla pausa/prosseguimento da execução
- Como se comunicam?

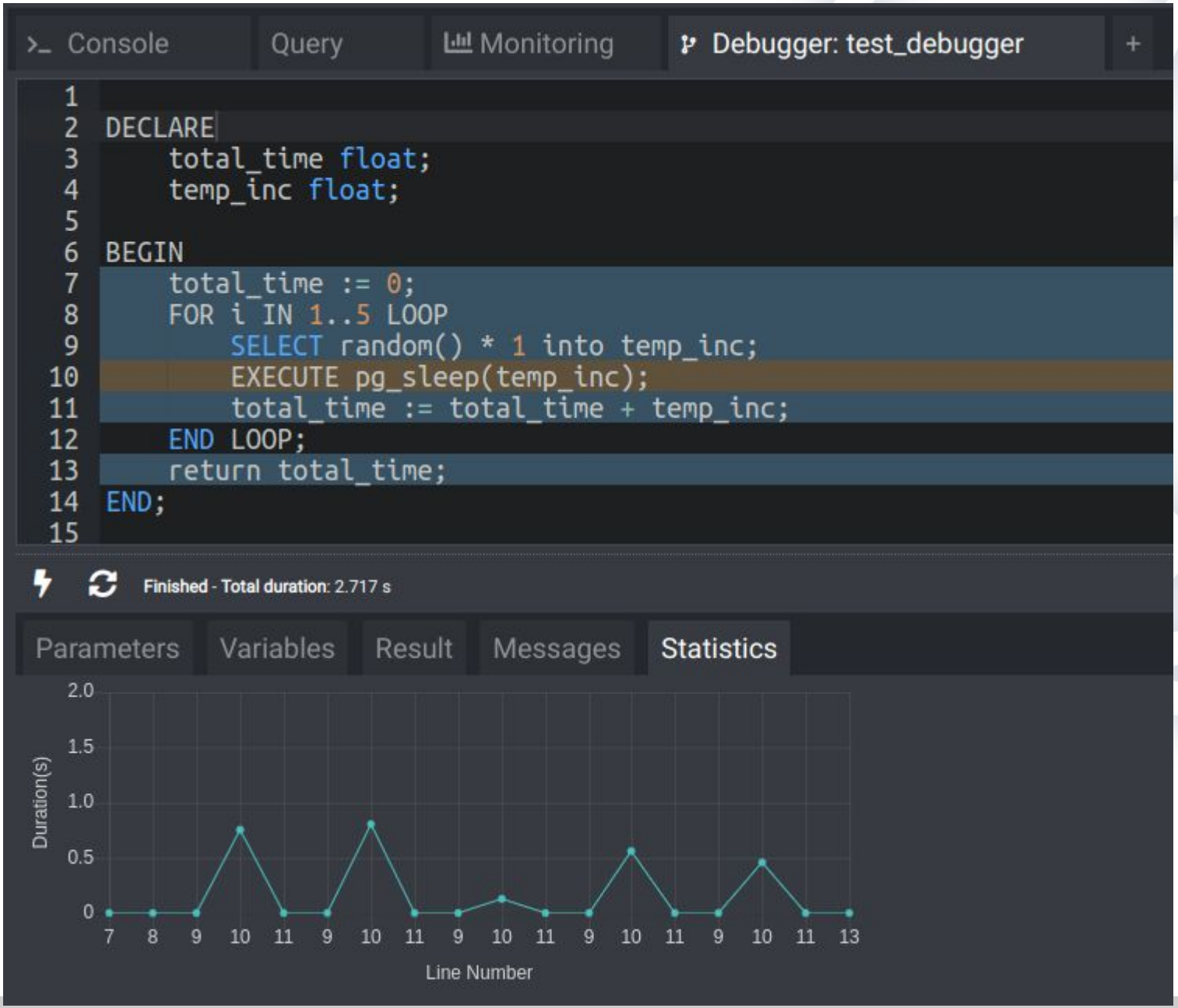
Depurador - Implementação

- pgAdmin
  - Comunicação cliente/servidor entre clientes
- OmniDB
  - *Advisory Locks*
  - Intercalação de aquisição de *locks*

Depurador - Implementação

	id	nome
1	1	nome 1
2	2	nome 2
3	3	nome 3

Cliente 1	Cliente 2
SELECT pg_advisory_lock(1) FROM tabela WHERE id = 1	.
.	SELECT pg_advisory_lock(1) FROM tabela WHERE id = 1
.	x
SELECT pg_advisory_unlock(1) FROM tabela WHERE id = 1	x
SELECT pg_advisory_lock(1) FROM tabela WHERE id = 1	.
x	.


Console
Query
Monitoring
Debugger: test_debugger
1
2 DECLARE
3 total_time float;
4 temp_inc float;
5
6 BEGIN
7 total_time := 0;
8 FOR i IN 1..5 LOOP
9 SELECT random() * 1 into temp_inc;
10 EXECUTE pg_sleep(temp_inc);
11 total_time := total_time + temp_inc;
12 END LOOP;
13 return total_time;
14 END;
15
Finished - Total duration: 2.717 s
Parameters
Variables
Result
Messages
Statistics
2.0
1.5
1.0
0.5
0
Duration(s)
7 8 9 10 11 9 10 11 9 10 11 9 10 11 9 10 11 13
Line Number

Depurador - OmniDB

Demonstração...

Background Worker

Background Worker

- Extender funcionalidades do PG

Background Worker

- Extender funcionalidades do PG
- Processo externo iniciado junto com o PG

Background Worker

- Extender funcionalidades do PG
- Processo externo iniciado junto com o PG
- Biblioteca externa escrita em C

Background Worker

- Extender funcionalidades do PG
- Processo externo iniciado junto com o PG
- Biblioteca externa escrita em C
- Carregada com *“shared_preload_libraries”*

Background Worker

- Extender funcionalidades do PG
- Processo externo iniciado junto com o PG
- Biblioteca externa escrita em C
- Carregada com *“shared_preload_libraries”*
- Acesso à memória compartilhada

Background Worker

- Extender funcionalidades do PG
- Processo externo iniciado junto com o PG
- Biblioteca externa escrita em C
- Carregada com *“shared_preload_libraries”*
- Acesso à memória compartilhada
- Acesso ao banco com interface interna (SPI)

Background Worker

- Extender funcionalidades do PG
- Processo externo iniciado junto com o PG
- Biblioteca externa escrita em C
- Carregada com *“shared_preload_libraries”*
- Acesso à memória compartilhada
- Acesso ao banco com interface interna (SPI)
- *libpq* para instanciar novas conexões e fazer transações

Background Worker

- *src/test/modules/worker_spi/worker_spi.c*

Obrigado!

rafael.castro@2ndquadrant.com