Jak zatrudnić słonie do replikacji baz PostgreSQL

Sławomir Civic Białek
<civic@bialek.org>

Jesień Linuksowa 2007, 22 września

Slony-I — replikacja baz PostgreSQL

O projekcie...

- system replikacji danych dla PostgreSQL
- rozwijany od 2004 roku
- Open Source
- Licencja BSD
- Jan Wieck@Afilias ... i inni
- aktualna seria 1.2.x
- dla PostgreSQL 7.4 do 8.3 (PostgreSQL 7.3 - tylko slony 1.1.x)

Slony-I — replikacja baz PostgreSQL

Różne podejścia do replikacji

- synchroniczna
- asynchroniczna
- pojedyncze źródło danych
- wiele źródeł danych
- jeden lub wiele odbiorców
- Slony-I — asynchroniczna, jedno źródło, wielu odbiorców
- Slony-II — synchroniczna, wiele źródeł, wielu odbiorców

Slony-I — replikacja baz PostgreSQL

Różne podejścia do replikacji

- synchroniczna
- asynchroniczna
- pojedyncze źródło danych
- wiele źródeł danych
- jeden lub wiele odbiorców
- Slony-I — asynchroniczna, jedno źródło, wielu odbiorców
- Slony-II — synchroniczna, wiele źródeł, wielu odbiorców

Slony-I — replikacja baz PostgreSQL

Terminologia

- Klaster (Cluster)
- Węzeł (Node)
- Zbiór (Set)
- Źródło (Origin)
- Dostawca (Provider)
- Subskrybent (Subscriber)

Slony-I — replikacja baz PostgreSQL

Ważne cechy

- "prawie" wiele źródeł
- jeden węzeł jednocześnie źródłem i subskrybentem
- odseparowane struktury danych w osobnym schemacie
- separacja — możliwy więcej niż jeden klaster
- różne wersje PostgreSQL w klastrze
- Linux, Windows, AIX, Solaris, *BSD
- tryb log shipping
- replikacja kaskadowa
- praca na wolnych i niepewnych łączach (WAN)
- skalowalność, wydajność

Slony-I — replikacja baz PostgreSQL

Ograniczenia

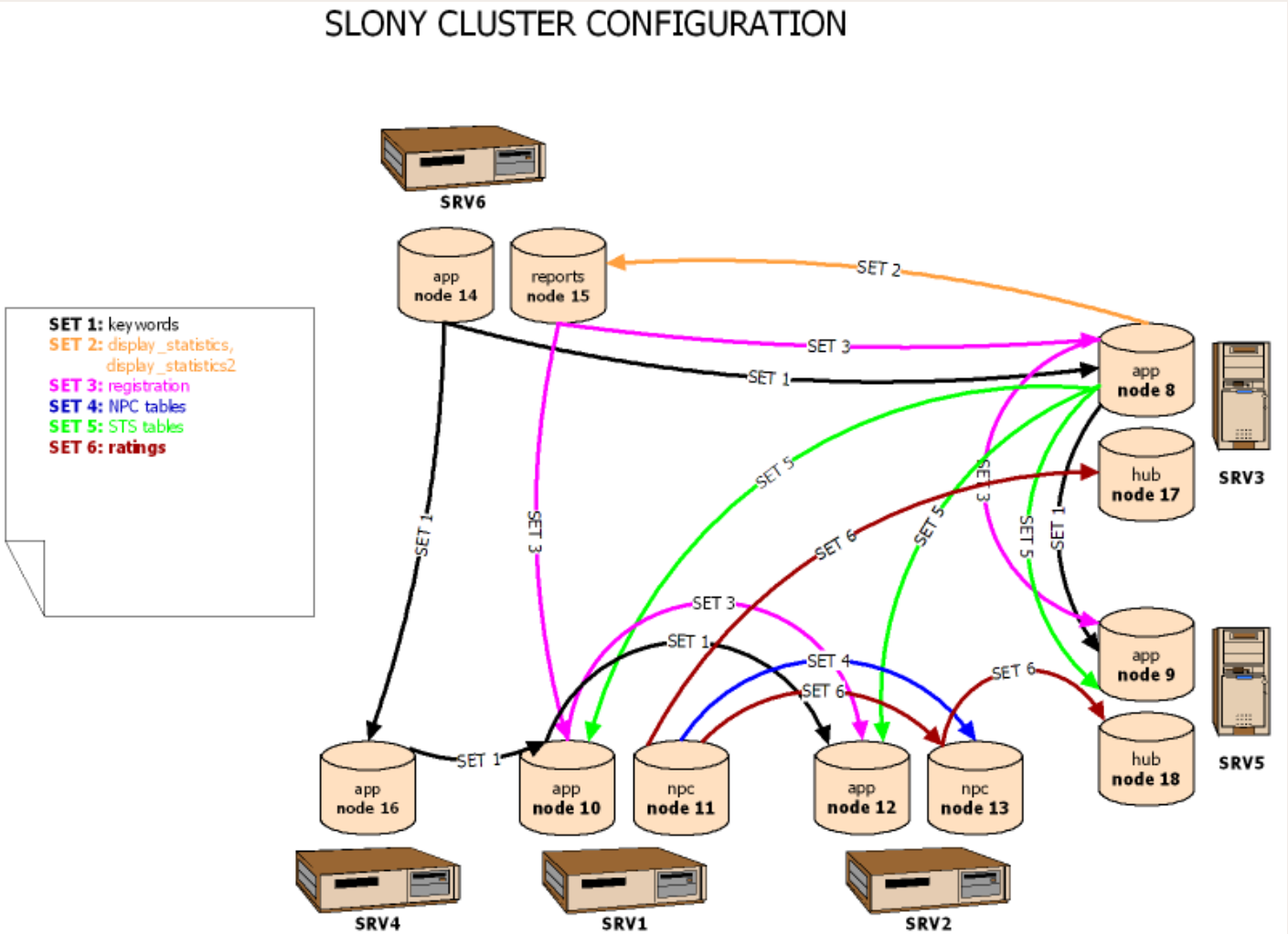
- tylko jedno źródło dla zbioru
- źródło — dostęp RW, reszta — dostęp RO
- brak replikacji DDL
- "siłowa" replikacja sekwencji wydajnosc, praktyczne ograniczenie ilosciowe, ale czy musza byc replikowane
- brak replikacji BLOBów
- ograniczenia ilościowe i wydajnościowe
- brak monitoringu działania silnika/slony
- brak przełączania połączeń klientów

Slony-I — replikacja baz PostgreSQL

Przykłady zastosowań

- grupa serwerów dla rozłożenia obciążenia
- udostępnienie kopii danych w trybie do odczytu
- zapasowe serwery
- tymczasowe przeniesienie pracy bazy na inny serwer
- online'owy backup
- przyrostowe aktualizacje kopii bazy oparte na trybie log shipping

Przykładowy klaster



Slony-I — replikacja baz PostgreSQL

Narzędzia

- slonik
- perl tools
- pgadmin
- monitoring, proxy dla połączeń

Slony-I — replikacja baz PostgreSQL

Jak to działa

- standardowe możliwości PostgreSQL
- procedury w C i PL/PGSQL
- triggery
- struktura danych — tabele, widoki, sekwencje
- program slonik
- daemon slon

Slony-I — replikacja baz PostgreSQL

Instalacja

- bez zakłócania pracy
- konfiguracja (i zmiany) w locie
- kompilacja

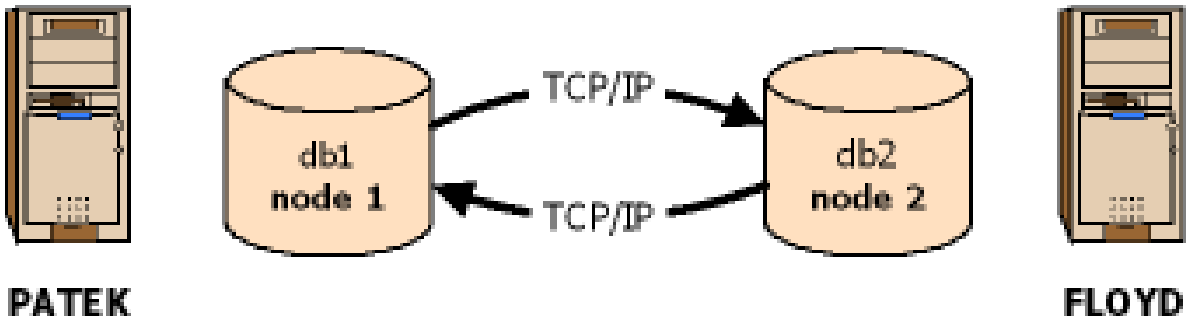
```
./configure --with-perltools --with-docs
LANG=C gmake
gmake install
```

Slony-I — replikacja baz PostgreSQL

Przygotowanie węzłów

- create user slony superuser
- połączenia z PostgreSQL po TCP/IP - postgresql.conf, pg_hba.conf, firewall
- create language plpgsql
- przenieść schematy replikowanych struktur danych (pg_dump -s)

Najprostsza konfiguracja



Struktura baz db1/db2:

Tabela tab1 (id=1)
Tabela tab2 (id=2)
Sekwencja tab1_id_seq (id=1)
Sekwencja tab2_id_seq (id=2)

Plik z definicjami - projekt1.slonik

```
# nazwa klastra
cluster name = projekt1;
# nazwy symboliczne - identyfikatory węzłów
define patek_db1 1;
define floyd_db2 2;
# połączenia
define patek_db1_conn
    'dbname=db1 host=patek user=slony';
define floyd_db2_conn
    'dbname=db2 host=floyd user=slony';
node @patek_db1 admin conninfo =
        @patek_db1_conn;
node @floyd_db2 admin conninfo =
        @floyd_db2_conn;
```

Utworzenie klastra

```
slonik <<_EOF_
  include <projekt1.slonik>
  init cluster ( id = @patek_db1,
      comment = 'wezel1' );
  store node   ( id = @floyd_db2,
      comment = 'wezel2' );

  store path (client = @patek_db1, server =
      @floyd_db2, conninfo = @floyd_db2_conn);
  store path (client = @floyd_db2, server =
      @patek_db1, conninfo = @patek_db1_conn);
_EOF_
```

Uruchomienie klastra

- slonik utworzył schematy _projekt1
- slon dokończy replikując konfigurację
- uruchamiamy procesy:

```
slon -d1 projekt1 \
    'host=patek dbname=db1 user=slony'
slon -d1 projekt1 \
    'host=floyd dbname=db2 user=slony'
```

- usunięcie struktur:

```
drop schema _project1 cascade;
```

Slony-I — replikacja baz PostgreSQL

Tworzenie zestawu

```
slonik <<_EOF_
  include <projekt1.slonik>
  create set (id = 1, origin = @patek_db1,
      comment = 'zbior1');
  set add table (id = 1, set id = 1,
       origin = @patek_db1,
       fully qualified name = 'public.tab1',
       comment = '');
  set add table (id = 2, set id = 1,
       origin = @patek_db1,
       fully qualified name = 'public.tab2',
       comment = '');
  set add sequence (id = 1, set id = 1,
       origin = @patek_db1, fully qualified
       name = 'public.tab1_id_seq',
       comment = '');
```

Tworzenie zestawu

- czy się udało:

```
SELECT * from _projekt1.sl_set;
(...)
db1=# \d tab1
(...)
Triggers:
  _projekt1_logtrigger_1 AFTER INSERT (...)
```

Subskrypcja

```
slonik <<_EOF_
  include <projekt1.slonik>
  subscribe set (id = 1, provider =
     @patek_db1, receiver = @floyd_db2,
     forward = no);
  wait for event (origin = @floyd_db2,
     confirmed = @patek_db1);
  sync (id = @patek_db1);
  wait for event (origin = @patek_db1,
     confirmed = @floyd_db2);
_EOF_
```

Zamiana ról — Switchover

```
slonik <<_EOF_
  include <projekt1.slonik>
  lock set (id = 1, origin = @patek_db1);
  wait for event (origin = @patek_db1,
     confirmed = @floyd_db2);
  move set (id = 1, old origin = @patek_db1,
     new origin = @floyd_db2);
  wait for event (origin = @patek_db1,
     confirmed = @floyd_db2);
_EOF_
```

I wracamy...

```
slonik <<_EOF_
  include <projekt1.slonik>
  lock set (id = 1, origin = @floyd_db2);
  wait for event (origin = @floyd_db2,
     confirmed = @patek_db1);
  move set (id = 1, old origin = @floyd_db2,
     new origin = @patek_db1);
  wait for event (origin = @floyd_db2,
     confirmed = @patek_db1);
_EOF_
```

Slony-I — replikacja baz PostgreSQL

Awaria — failover

```
slonik <<_EOF_
  include <projekt1.slonik>
  failover (id = 1,
     backup node = @floyd_db2);
  drop node (id = @patek_db1,
     event node = @floyd_db2);
_EOF_
```

Inne przydatne polecenia slonika

- MERGE SET
- STORE TRIGGER
- EXECUTE SCRIPT
- DROP ...

Dziękuję za uwagę...