Słonie pracują w stadzie

Adam Buraczewski
aburacze@gmail.com

GNU/Politechnika, 13.01.2007 r.

Plan wykładu

- Wprowadzenie – po co łączyć serwery bazodanowe?
- Transakcje rozproszone: 2PC, dblink, JTA
- Równoważenie obciążenia serwerów: pgpool, Sequoia, PostgreSQL Relay
- Replikacja asynchroniczna: Slony-I
- Replikacja synchroniczna: PGCluster, PostgreSQL-R

Najważniejsze pojęcia

- Transakcja – pojedyncza logiczna operacja na bazie danych, mogąca się składać z wielu poleceń. Cechy:
  - *Atomicity* – jest zrealizowana w całości albo wcale
  - *Consistency* – po jej zakończeniu baza jest spójna
  - *Isolation* – inni użytkownicy bazy danych nie widzą pojedynczych operacji składających się na transakcję
  - *Durability* – zatwierdzona transakcja nie może być cofnięta, jej efekt jest utrwalony w bazie

Najważniejsze pojęcia

- Transakcja rozproszona:
  - Obejmuje kilka serwerów bazodanowych połączonych za pomocą sieci
  - Zapewnia cechy *ACID* ale w obrębie tych kilku serwerów na raz
  - Realizacja techniczna: za pomocą protokołu wielofazowego zatwierdzania (*Two-Phase Commit*)

Najważniejsze pojęcia

- Replikacja – automatyczne przekazywanie zmian w bazie danych do innych serwerów w sieci
  - Master-slave – zmiany dozwolone są tylko na głównym serwerze w sieci i są propagowane do serwerów podrzędnych
  - Multi-master – zmiany mogą być nanoszone na dowolnym z serwerów i są propagowane do pozostałych

Najważniejsze pojęcia

- Równoważenie obciążenia (*Load balancing*):
  - Przekierowanie poleceń/zapytań bazodanowych do różnych serwerów w sieci, w celu ich optymalnego wykorzystania
  - Stosowane zwykle razem z replikacją
- *Failover* – reakcja systemu na awarię, przekierowanie poleceń/zapytań do innego serwera w sieci
- *Failback* – przywrócenie poprzedniej konfiguracji po zlikwidowaniu awarii

Najważniejsze pojęcia

- Partycjonowanie danych – podzielenie jednej logicznej bazy danych pomiędzy kilka serwerów:
  - Poziome – każdy serwer ma te same tabele, ale różne zakresy wierszy (np. w zależności od daty wpisanej do jednej z kolumn)
  - Pionowe – każdy serwer ma inny zestaw tabel
  - Mieszane
- Umożliwia lepsze rozłożenie obciążenia, ułatwia określenie polityki dostępu (np. dane osobowe na osobnym serwerze) itp.

Zalety łączenia serwerów

- Połączenie wielu serwerów za pomocą sieci pozwala na:
  - Zwiększenie wydajności poprzez rozłożenie obciążenia
  - Zwiększenie dostępności (*availability*) systemu
  - Partycjonowanie danych ze względów np. prawnych lub innych

Wady łączenia serwerów

- Połączenie wielu serwerów za pomocą sieci jest kłopotliwe ze względu na:
  - Konieczność wymiany informacji o pracy serwerów (blokady, transakcje, spójność danych, awarie, wybór nowych głównych serwerów)
  - Techniczną realizację rozłożenia obciążenia pomiędzy serwerami, autoryzacji użytkowników itp.
  - Techniczną realizację replikacji danych pomiędzy serwerami
  - Konieczność monitoringu, oprogramowania

Moduł dblink

- Dostęp do baz PostgreSQL z poziomu SQL
- Dostępny od wersji 7.3, wykorzystuje SRF
- Sposób użycia:
  - SELECT dblink_connect('host=192.168.0.1 dbname=baza');
  - SELECT * FROM dblink('SELECT c1, c2 FROM tabela') AS (c1 INTEGER, c2 INTEGER);
  - SELECT dblink_exec('UPDATE tabela SET c2 = 4 WHERE c1 = 2');
  - SELECT dblink_close();

Moduł dblink

- Możliwość stosowania w perspektywach i procedurach składowanych, triggerach itp. Przykład:
  - CREATE VIEW zdalna_tabela AS SELECT * FROM dblink('SELECT c1, c2 FROM tabela') AS (c1 INTEGER, c2 INTEGER);
    SELECT * FROM zdalna_tabela;
- API do pracy z kilkoma połączeniami naraz
- API do dynamicznego budowania poleceń SQL, odczytu schematu zdalnej bazy danych itp.

Moduł dblink

- Problem z transakcjami:
  - BEGIN;
  - SELECT dblink_exec('BEGIN');
  - SELECT dblink_exec('INSERT ...');
  - SELECT dblink_exec('COMMIT');
  - ROLLBACK;
- Rozwiązanie: transakcje rozproszone, realizowane za pomocą 2PC

Alternatywy dla dblink

- Inne rozwiązania, podobne do dblink:
  - *dblinkora, dblink-TDS* – stosowanie identyczne jak *dblink*
  - *Dblink-DBI* – wykorzystuje język pl/Perl i sterowniki DBI przeznaczone dla Perla
    - Skrypt analizujący strukturę wskazanej bazy danych i tworzący lokalnie komplet perspektyw do korzystania z tabel zdalnej bazy jak z tabel lokalnych

Transakcje rozproszone - 2PC

- Mechanizm *Two-Phase Commit* – protokół dwufazowego zatwierdzania transakcji
- Standardowe rozwiązanie dla wielu serwerów baz danych
- Przystosowane do działania z managerami transakcji (np. standard JTA dla Javy – odpowiednie wsparcie w sterowniku JDBC).

Transakcje rozproszone - 2PC

- Polecenia (transakcję rozpoczyna BEGIN):
  - PREPARE TRANSACTION 'nazwa' – przygotowuje transakcję; pomyślne zakończenie gwarantuje w 100% możliwość późniejszego wykonania COMMIT PREPARED, nawet po np. awarii zasilania
  - COMMIT PREPARED 'nazwa' – ostatecznie zatwierdza transakcję
  - ROLLBACK PREPARED 'nazwa' – wycofuje
- PREPARE TRANSACTION pozostawia założone blokady na tabele i wiersze

Transakcje rozproszone - 2PC

- Możliwość sprawdzenia aktualnie przygotowanych transakcji za pomocą: SELECT * FROM pg_prepared_xacts;
- Zatwierdzić/wycofać przygotowaną transakcję może użytkownik, który ją przygotował lub administrator

Dblink i 2PC

- Stosowanie mechanizmu 2PC z modułem *dblink* do wiarygodnego zatwierdzania transakcji:
  - BEGIN;
  - SELECT dblink_exec('BEGIN');
  - SELECT dblink_exec('INSERT ...');
  - SELECT dblink_exec('PREPARE TRANSACTION ''t123'';');
  - COMMIT;
  - SELECT dblink_exec('COMMIT PREPARED ''t123'';');

JTA

- *Java Transaction API*- standardowy mechanizm języka Java do obsługi transakcji rozproszonych
- Obsługuje wiele różnych źródeł danych o zachowaniu transakcyjnym (np. Hibernate, obiekty EJB itp.)
- Wymaga *Java Transaction Service* – dostępne w J2EE, Jboss, oraz jako niezależne implementacje (np. JOTM z rozwiązań Open Source)

JTA

- Wykorzystanie:
  - UserTransaction ut = (UserTransaction) context.lookup(“java:comp/UserTransaction”) ;
  - ut.begin();
  - // dostęp do wielu źródeł danych
  - ut.commit();
- Zatwierdzenie transakcji odbywa się z użyciem mechanizmów 2PC, o ile dane źródło danych obsługuje transakcje rozproszone (PostgreSQL jak najbardziej do nich należy)

Sequoia

- Dawniej: Clustered-JDBC
- Produkt Open Source, niezależny od PostgreSQL, ale dobrze z nim współpracuje
- Działa z językiem Java, ale jest API dla C++
- Organizuje cały klaster złożony z wielu baz danych i udostępnia go za pomocą pojedynczego połączenia JDBC
- Realizuje replikację i partycjonowanie danych, równoważenie obciążenia itp.

Sequoia

- Serwery bazodanowe (w tym PostgreSQL) służą jedynie składowaniu danych, nie można z nimi współpracować bezpośrednio
- Sequoia wykorzystuje własny mechanizm wieloetapowego zatwierdzania transakcji, nie korzysta z 2PC (nie współpracuje też z JTA), sama realizuje synchronizację baz po awarii
- Wygodna administracja: graficzne “wizardy”, narzędzia do administracji i backupu całego klastra

pgpool

- Dojrzały system do rozkładania obciążenia pomiędzy kilka serwerów
- Utrzymuje pulę otwartych połączeń do serwera PostgreSQL, skracając czas potrzebny na np. autoryzację
- Wykorzystuje natywny protokół komunikacji klient-serwer systemu PostgreSQL, współpracuje więc z każdym programem, biblioteką czy sterownikiem
- Doskonale sprawdza się w serwisach WWW

pgpool

- Pgpool I:
  - Obsługa tylko 2 serwerów PostgreSQL
  - Polecenia SELECT kierowane tylko do jednego z serwerów (można określić jak ma być rozłożone obciążenie)
  - Polecenia INSERT/UPDATE/DELETE kierowane do obydwóch serwerów, dzięki czemu bazy danych pozostają spójne
  - *Failover* – w przypadku awarii wszystkie połączenia przekierowane do drugiego serwera
  - Nie potrafi korzystać z mechanizmu 2PC

Pgpool

- Pgpool II (wydany w 2006 r.):
  - Dowolnie wiele serwerów składających się na klaster
  - Tryb “parallel” - zapytania rozkładane są na podzapytania, wykonywane niezależnie na różnych serwerach, a wynik łączony (przezroczyste dla klienta)
  - Nowy, wygodny panel administracyjny – pgpooladmin (napisany w PHP)
  - Lepsza współpraca z systemem Slony-I

Pgpool

- Bardziej elastyczny niż Sequoia, lepiej dostosowany do PostgreSQLa
- Wymaga, aby przed uruchomieniem bazy danych były zsynchronizowane np. za pomocą systemu Slony
- Możliwość podawania “podpowiedzi” w kodzie SQL, dotyczących realizacji zapytania w klastrze
- Warto stosować nawet dla pojedynczego serwera obsługującego np. serwis WWW (dużo krótkich sesji bazodanowych).

PostgreSQL Relay

- Integruje wiele rozproszonych serwerów i baz danych na jednym serwerze
- Wygodne rozwiązanie gdy przeniesiono pojedynczą bazę danych lub cały serwer pod inną lokalizację, a nie chcemy zmieniać konfiguracji programów
- Możliwość użycia tcpwrappers do kontroli dostępu
- Prosty plik konfiguracyjny

SQL Relay

- Realizuje *connection pooling* dla wielu różnych serwerów bazodanowych i języków programowania
- Ma własny protokół komunikacji klient-serwer
- Nie potrafi synchronizować baz danych – należy korzystać z replikacji osobnym mechanizmem lub korzystać z baz w trybie “tylko do odczytu”
- Wygodne narzędzia administracyjne (graficzny do konfiguracji, tekstowy

SQL Relay

- Elementy składowe:
  - sqlr-connection – łączy się do bazy danych
  - sqlr-listener – udostępnia pulę połączeń aplikacjom
  - sqlr-scaler – uruchamia dodatkowe połączenia w razie potrzeby
- Udostępnia połączenia standardowymi mechanizmami (DBI, JDBC itp.) oraz za pomocą własnych bibliotek (własna wersja biblioteki libpq, pozwala na korzystanie z SQL Relay bez rekompilacji programu przeznaczonego dla PG)

Slony-I

- Najbardziej dojrzałe rozwiązanie Open Source do replikacji baz danych dla PostgreSQL
- Opracowany w 2004 r. przez Jana Wiecka dla firmy Afilias
- Replikacja typu master-slave (jeden serwer główny i dowolnie wiele podrzędnych, zorganizowanych w drzewo)
- Działa dla PostgreSQL 7.3, nie wymaga żadnych zmian w kodzie serwera

Slony-I

- Wykorzystuje mechanizmy LISTEN/NOTIFY, własne procedury PL/pgSQL oraz triggery
- Nie replikuje zmian w strukturze bazy danych (tylko w samych danych w tabelach)
- Replikuje pojedyncze bazy danych lub ich fragmenty (operuje na zestawach tabel – *set*).
- Wymaga, aby węzły klastra były nawzajem widoczne; przeznaczony dla

Slony-I

- Działanie:
  - Modyfikacja bazy głównej powoduje umieszczenie informacji o zmianie w prywatnych tabelach Slony'ego (razem dla całej transakcji)
  - Bazy podrzędne mają zablokowaną możliwość zmian w replikowanych tabelach
  - Proces “slon” odczytuje te zmiany i przenosi do serwerów podrzędnych
  - Pomyślne naniesienie zmian we wszystkich bazach podrzędnych powoduje wykasowanie logów z serwera

Slony-I

- Konfiguracja za pomocą programu “slonik”, wyposażonego w specjalny język skryptowy
- Przed uruchomieniem replikacji należy przenieść samą strukturę tabel BEZ danych (pg_dump -s | pg_restore)
- Etap I – określenie węzła nadrzędnego i podrzędnych, następnie uruchomienie procesów “slon”
- Etap II – określenie zestawów tabel do replikacji

Slony-I

- Etap I:

```
cluster name = moj_klaster;
node 1 admin conninfo = 'dbname=nadrzedny
host=192.168.0.1';
node 2 admin conninfo = 'dbname=podrzedny
host=192.168.0.2';
init cluster (id = 1);
store node (id = 2);
store path (server = 1, client = 2,
    conninfo = ''dbname=nadrzedny host=192.168.0.1');
store path (server = 2, client = 1,
    conninfo = ''dbname=podrzedny host=192.168.0.2');
store listen (origin = 1, provider = 1, receiver = 2);
store listen (origin = 2, provider = 2, receiver = 1);
```

Slony-I

- Etap II:

```
cluster name = moj_klaster;
node 1 admin conninfo = 'dbname=nadrzedny
host=192.168.0.1';
node 2 admin conninfo = 'dbname=podrzedny
host=192.168.0.2';
create set (id = 1, origin = 1);
set add sequence (set id = 1, origin = 1, id = 1,
    fully qualified name = 'public.sekwencja');
set add table (set id = 1, origin = 1, id = 1,
    fully qualified name = 'public.tabela1');
set add table (set id = 1, origin = 1, id = 2,
    fully qualified name = 'public.tabela2');
set add table (set id = 1, origin = 1, id = 3,
    fully qualified name = 'public.tabela3');
subscribe set (id = 1, provider = 1, receiver = 2, forward
```

Slony-I

- *Failover* z wyborem węzła najbardziej aktualnego (wykrywanie awarii musi być oprogramowane przez administratora)
- Możliwość przekazywania danych z określonym opóźnieniem
- Możliwość przekazywania danych poprzez pliki
- Bogactwo języka skryptowego “slonik”

PGCluster, PostgreSQL-R

- Tworzone, ale już całkiem stabilne rozwiązania replikacji multi-master
- Ingerują w kod źródłowy PostgreSQL
- Replikacja całej instalacji PostgreSQLa na raz (bez możliwości wybrania baz danych)
- Wszystkie węzły klastra są równorzędne
- Odczyty baz danych są realizowane lokalnie, zapis i blokady muszą być synchronizowane między węzłami klastra (wymaga szybkiej sieci)

PostgreSQL-R

- Dawniej PGReplicator
- Dostępna wersja binarna – obraz ISO płyty z Linuksem i skonfigurowanym PostgreSQL-R
- Do komunikacji między węzłami wykorzystany jest mechanizm Spread
- Wymaga stosowania zewnętrznego programu do rozkładania obciążenia (np. pgpool)

PGCluster

- Dostępny w postaci łaty na PostgreSQL (nawet na wersję 8.2.1!)
- Ma własny program do rozkładania obciążenia, ora dodatkowy demon do monitoringu klastra
- Do wstępnej synchronizacji klastra wykorzystuje program rsync i ssh, na każdym węźle klastra cała instalacja musi być w tym samym katalogu

Adresy

- www.pgfoundry.org - serwis pgFoundry, czyli katalog i repozytorium wiekszosci oprogramowania zwiazanego z PostgreSQL, np. pgpool, PGcluster, PostgreSQL Relay, SkyTools itp.,
- slony.info - strona domowa projektu Slony I,
- www.postgres-r.org - strona domowa projektu Postgres-R,
- jotm.objectweb.org - strona domowa JOTM,
- sequoia.continuent.org – strona projektu Sequoia,
- sqlrelay.sourceforge.net - stronaSQL Relay.