Opis Rozwiązania - Sharding

Projekt – RSO MongoDB

Autorzy:

* Tomasz Adamiec
* Piotr Cebulski
* Marek Kowalski
* Mateusz Rosiewicz
* Paweł Sokołowski
* Marcin Wnuk

Sharding

Podział bazy na części – shardy został zrealizowany na wzór shardingu w MongoDB. Zachowana została koncepcja shardów, serwera konfiguracyjnego oraz procesu mongos, odpowiedzialnego za równoważenie obciążenia shardów. Ze względu na znaczne uproszczenie funkcji poszczególnych elementów, niektóre aspekty zostały zmienione w stosunku do MongoDB.

* 1. Serwer Konfiguracyjny

Serwer konfiguracyjny jest centralnym elementem w bazie. W nim przechowywana jest informacja o aktualnym stanie wszystkich shardów. W implementacji serwera został wykorzystany interfejs Java RMI.

W pakiecie configserver znajdują się cztery klasy:

* Rem
* RemClient
* RemServer
* RemImpl.

Pierwsza z nich zawiera jedynie prototypy funkcji i służy do tworzenia stub’a, czyli obiektu widocznego dla klientów serwera przez rejestr RMI. Obiekt Rem pozwala klientom na wywoływanie metod: registerToConfigServer(), updateShardInfo(), registerBalancer(), getShards(). Dwie pierwsze są zarezerwowane dla shardów, a dwie ostatnie dla balancera.

Klasa RemClient jest wykorzystywana przez shardy w celu połączenia się do serwera konfiguracyjnego. Po zarejestrowaniu się w serwerze wątek klienta usypia aż do momentu wybudzenia go przez ShardMonitor. Wtedy następuje aktualizacja stanu shardu w serwerze konfiguracyjnym.

RemServer tworzy rejestr RMI z nasłuchiwaniem na domyślnym porcie 1099 i nadaje nazwę usługi widoczną przez sieć: //<serverIP>/Rem.

W klasie RemImpl znajduje się implementacja funkcji, które są zdalnie wykonywane przez shardy i balancer. Zawiera ona również informacje o shardach, w postaci mapy, której kluczami są adresy IP, a wartościami obiekty balancer.ShardInfo:

HashMap<InetAddress, ShardInfo> shards;

Dodatkowo, klasa ta korzysta z blokady java.util.concurrent.locks.ReentrantLock w celu uchronienia przed jednoczesnym zapisem i odczytem informacji o shardzie.

* 1. Shardy

Każdy shard spełnia podstawową funkcjonalność bazy – nasłuchuje na porcie 27017, tworzy wątki dla wszystkich podłączonych klientów mongo i realizuje ich zapytania. Ponadto, wykonuje on dodatkowy wątek monitorujący stan bazy. Służy do tego wspomniana wcześniej klasa ShardMonitor. Monitor skanuje zawartość folderu z bazą i zapisuje informacje o wszystkich plikach we wszystkich kolekcjach w obiekcie ShardInfo. Wylicza on przy tym obciążenie sharda, czyli łączną wielkość wszystkich plików. Kolejną funkcją ShardMonitora jest wznawianie wątku RemClient’a, w celu wysyłania zaktualizowanej informacji o stanie shardu do serwera konfiguracyjnego.

Po podłączeniu sharda do bazy rozproszonej uruchamiany jest dodatkowo wątek MigrationListener’a. Klasa MigrationListener pozwala na odbieranie żądań przeniesienia części dokumentów do innego sharda bazy. Żądania są wysyłane przez balancer tylko do sharda źródłowego. MigrationListener wysyła również odpowiednie pliki do sharda o mniejszym obciążeniu (na podstawię żądania balancera) oraz odbiera pliki z innych shardów.

Reasumując shard spałnia następujące funkcje:

* Realizowanie zapytań od klientów bazy
* Monitorowanie stanu sharda
* Przesyłanie informacji o stanie sharda do serwera konfiguracyjnego
* Odbieranie informacji o migracji danych
* Przenoszenie danych między shardami
  1. Balancer

Podstawowym zadaniem balancera jest obliczanie polityki równoważenia. W klasie balancer.Balancer następuje wyszukanie usługi Rem serwera konfiguracyjnego i podłączenie do niego za pomocą metody registerBalancer(). Następnie rozpoczyna się wykonywanie rund balancera. W każdej rundzie pobierana jest najbardziej aktualna informacja o stanie bazy za pomocą funkcji getShards().

Gdy balancer otrzyma informacje o shardach tworzony jest obiekt BalancerPolicy. BalancerPolicy na podstawie danych zawartych w HashMap<InetAddress, ShardInfo> wyznacza politykę równoważenia i zapisuje ją w postaci obiekty MigrateInfo.

Algorytm wyznaczania polityki równoważenia przebiega w dwóch etapach. Najpierw, znajdywane są shardy o najmniejszym i największym obciążeniu. Jeżeli różnica między nimi jest większa niż zakładany próg, to następuje drugi etap – wyznaczanie dokumentów do przeniesienia. W tym etapie znajdywane są dokumenty, których zsumowana wielkość będzie niemniejsza niż połowa różnicy obciążenia shardów. Wybierane są dokumenty o wielkości najbardziej zbliżonej do średniej wielkości dokumenty w mocno obciążonym shardzie.

Po wyznaczeniu polityki przez BalancerPolicy, w klasie Balancer następuje sprawdzenie czy należy coś przenieść – czy MigrateInfo nie jest puste. Jeśli nie, to znaczy, że zawiera ono adres sharda mocno obciążonego i mniej obciążonego i nazwy dokumentów, które należy między nimi przenieść. Na końcu rundy, informacje zawarte w MigrateInfo są przesyłane do sharda mocno obciążonego na port 28017, na którym nasłuchuje MigrationListener.

Balancer’a nie trzeba dołączać do bazy, nie jest on niezbędny do jej funkcjonowania.

Sharding:

1. Shard rejestruje się w serwerze konfiguracyjnym.
2. ShardMonitor aktualizuje informacje o shardzie.
3. RemClient wysyła informacje do serwera konfiguracyjnego.
4. Balancer pobiera informacje o wszystkich shardach.
5. BalancerPolicy wyznacza politykę i tworzy MigrateInfo.
6. MigrateInfo wysyłane jest do sharda.
7. Shard wysyła dokumenty do innego sharda.

Warszawa, 2013