AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI
INSTYTUT INFORMATYKI

Projekt semestralny

Studium przypadku technologii Vitess

Autor: Szymon Twardosz, Dominik Pilipczuk, Krzysztof Tranquilo Dziechciarz, Wiktor Gut

Kraków, 2025

Spis treści

Rozdział 1

Cel projektu

1.1. Wprowadzenie

Celem projektu jest przedstawienie studium przypadku systemu **Vitess** – open-sourcowej platformy służącej do skalowania baz danych MySQL w środowiskach chmurowych i rozproszonych. Vitess został stworzony przez YouTube jako odpowiedź na rosnące potrzeby skalowalności i dostępności danych w dużych systemach produkcyjnych, gdzie tradycyjne podejście do baz danych relacyjnych okazywało się niewystarczające.

Vitess łączy w sobie zalety tradycyjnych baz danych (jak ACID i SQL) z elastycznością architektur opartych na mikrousługach i kontenerach. Umożliwia m.in. sharding, replikację, przełączanie awaryjne oraz zarządzanie schematem w sposób spójny i zautomatyzowany. Dzięki integracji z Kubernetesem i innymi narzędziami cloud-native, Vitess idealnie wpisuje się w potrzeby nowoczesnych, skalowalnych aplikacji.

Przykładowe firmy i usługi które korzystają z technologii Vitess to:

- **YouTube** – gdzie projekt się narodził, jako rozwiązanie problemów skalowalności bazy danych,
- **Slack** - dla obsługi ogromnej ilości wiadomości i użytkowników w czasie rzeczywistym,
- **GitHub** - do obsługi skomplikowanej infrastruktury danych przy zachowaniu wysokiej dostępności i wydajności.

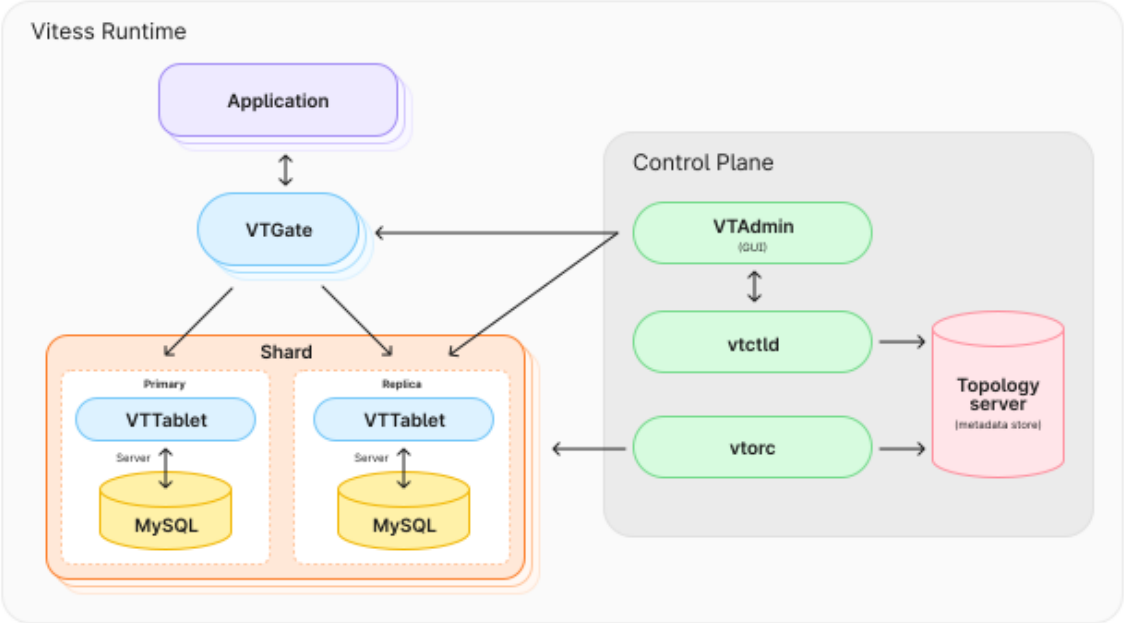
1.2. Wstęp teoretyczny

Vitess składa się z kilku kluczowych komponentów, które współpracują, aby zapewnić wydajność, skalowalność oraz wysoką dostępność baz danych. Oto najważniejsze z nich:

- Vtorc - odpowiada za automatyczne zarządzanie topologią replikacji MySQL oraz wykrywanie i reagowanie na awarie w klastrze bazodanowym.
- VTGate - brama (proxy) łącząca aplikacje z systemem Vitess. Odpowiada za przyjmowanie zapytań SQL od klientów i ich kierowanie do odpowiednich shardów i replik.
- VTTablet - serwis działający obok instancji MySQL w każdej replice. Odpowiada między innymi za zarządzanie instancją MySQL oraz odpowiadanie na zapytania przychodzące z VTGate

- vtctld - interfejs administracyjny służący do zarządzania klastrem Vitess. Umożliwia między innymi monitorowanie stanu klastra czy tworzenie nowych shardów oraz replik.
- Shardy - instancję baz danych, do których kierowane są zapytania. Każdy z nich posiada własną instancję VTTablet, która jest pośrednikiem w komunikacji między instancją bazy danych a VTGate.

  Architektura tej technologi wygląda następująco (obrazek wzięty z oficjalnej dokumentacji Vitess):



Rysunek 1.1: Architektura Vitess

1.3. Opis koncepcji

W ramach projektu opartego o system Vitess, zaprojektowano i przeanalizowano trzy scenariusze ilustrujące różne aspekty działania i odporności systemu bazodanowego opartego o sharding oraz replikację. Poniżej przedstawiono szczegółowy opis każdego scenariusza wraz z jego logicznymi krokami.

1.3.1. Sharding horyzontalny – instancje baz danych posiadają swoje tabele

Sharding horyzontalny (ang. *horizontal sharding*) polega na podziale danych tej samej tabeli pomiędzy różne instancje baz danych, zwykle na podstawie zakresu wartości klucza głównego (np. `user_id`). Każda instancja przechowuje ten sam schemat, lecz inny podzbiór danych.

Kroki scenariusza:

1. Użytkownik wykonuje zapytanie, np. `SELECT * FROM users WHERE user_id = 1350`.

2. Komponent **VTGate** odbiera zapytanie i na podstawie klucza dzielącego kieruje je do odpowiedniego sharda.
3. VTGate przekazuje zapytanie do właściwego **VTTablet**, obsługującego shard zawierający dane użytkownika.
4. VTTablet komunikuje się z backendem MySQL i otrzymuje wynik zapytania.
5. VTGate, w razie potrzeby, agreguje wyniki i zwraca je użytkownikowi.

1.3.2. Sharding wertykalny – repliki całej bazy danych

Sharding wertykalny (ang. *vertical sharding*) polega na logicznym podziale schematu bazy danych, gdzie różne tabele są rozmieszczone na różnych instancjach. Każda instancja może posiadać swoje repliki, co umożliwia równoważenie obciążenia zapytań odczytujących.

Kroki scenariusza:

1. Użytkownik wykonuje zapytanie, np. `SELECT * FROM users`.
2. VTGate identyfikuje, że tabela `users` znajduje się w bazie A.
3. Zapytanie jest kierowane do odpowiedniego **VTTablet**.
4. W przypadku zapytania typu `SELECT`, VTGate może skierować je do jednej z replik (nie do instancji głównej).
5. Replika MySQL odpowiada danymi.
6. VTGate przekazuje wynik użytkownikowi.
7. Jeśli zapytanie wymaga wielu tabel (np. `JOIN`), dane są zbierane z odpowiednich shardów i scala je VTGate.

1.3.3. Błąd głównego węzła – failover na replikę

Scenariusz ten ilustruje odporność systemu w przypadku awarii głównego węzła bazy danych. Vitess zapewnia mechanizm automatycznego przełączenia (ang. *failover*) na replikę, która przejmuje rolę nowego głównego węzła.

Kroki scenariusza:

1. Główna instancja (primary) w shardzie przestaje odpowiadać.
2. System monitorujący (np. `vtctld` lub `orchestrator`) wykrywa awarię.
3. Jedna z replik zostaje wybrana jako nowy **primary**.
4. Pozostałe repliki zostają przekonfigurowane do replikowania z nowego primary.
5. VTGate aktualizuje informacje routingu, kierując zapytania zapisujące do nowej instancji głównej.
6. System kontynuuje pracę z minimalnym przestojem.

1.4. Architektura rozwiązania

1.5. Konfiguracja środowiska

1.6. Metody instalacji

1.7. How to reproduce - step by step

Czym różnią się te dwa punkty ?

1.8. Demo deployment steps

Czym różnią się te dwa punkty ?

1.9. Użycie AI w projekcie

1.10. Wnioski