УДК 004.4

Кутковий Богдан Євгенович
магістрант
Національного університету «Львівська політехніка»
Кутковой Богдан Евгеньевич
магистрант
Национального университета «Львовская политехника»
Kutkovy Bogdan
Graduate student of the
Lviv Polytechnic National University

Павич Наталія Ярославівна
кандидат технічних наук,
доцент кафедри програмного забезпечення
Національний університет «Львівська політехніка»
Павыч Наталья Ярославовна
кандидат технических наук,
доцент кафедры программного обеспечения
Национальный университет «Львовская политехника»
Pavych Natalia
Candidate of Technic Sciences,
Associated Professor of Software Department
Lviv Polytechnic National University

ОПТИМІЗАЦІЯ API ЗАПИТІВ ДО ХМАРНИХ СИСТЕМ УПРАВЛІННЯ БАЗАМИ ДАНИХ

ОПТИМИЗАЦИЯ API ВЫЗОВОВ К ОБЛАЧНЫМ СИСТЕМАМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

API–CALLS OPTIMIZATION FOR CLOUD DATABASE-MANAGEMENT SYSTEMS

Анотація. *Висвітлено порівняльний аналіз сучасного та розробленого підходів до використання API сервісів хмарних систем управління базами даних. Наведено графічне та табличне порівняння швидкодії підходів до використання API сервісів.*

Ключові слова: *API, хмарні системи управління базами даних, сховища даних, швидкодія, gem-бібліотека, операції CRUD, Ruby on Rails framework, ORM, ActiveRecord.*

Аннотация. *Освещен сравнительный анализ современного и разработанного подходов использования API сервисов облачных систем управления базами данных. Приведены графическое и табличное сравнение быстродействие подходов использования API сервисов.*

Ключевые слова: *API, облачные системы управления базами данных, хранилища данных, быстродействие, gem-библиотека, операции CRUD, Ruby on Rails framework, ORM, ActiveRecord.*

Summary. *Covering comparative analysis of modern and developed approaches of using cloud database-management systems API services. Demonstrated graphical and table comparison of performance using API services approaches.*

Key words: *API, cloud database-management systems, data store, performance, gem-library, CRUD operations, Ruby on Rails framework, ORM ActiveRecord.*

Вступ. Хмарні бази даних — це бази даних, які запускаються на платформах хмарних обчислень, таких як Amazon EC2, GoGrid і Rackspace. Існують дві найпоширеніші моделі розгортання: користувачі можуть придбати безпосередньо послугу доступу до баз даних, які обслуговує постачальником хмарного сервісу, або ж запустити бази даних в хмарі незалежно, використовуючи образ віртуальної машини. Серед хмарних баз даних присутні як SQL-орієнтовані, так і ті, що використовують модель даних NoSQL.

Сервіси баз даних складаються з компоненту-менеджера БД, що керує основними екземплярами БД використовуючи API цього сервіса. API сервіса розкриті для кінцевого користувача, та дозволяє користувачам виконувати обслуговування та операції зміни розміру на їх екземплярах БД. Наприклад, Amazon Relational Database Service's сервісне API дозволяє створювати екземпляр БД, проводити модифікацію ресурсів доступних для екземпляра БД, видаляти екземпляр БД, створювати снапшоти (схоже до бекапів) баз даних, та відновлювати БД зі снапшоту.

Сучасний підхід до використання API сервісу хмарних систем управління базами даних є малоефективним. Проблема полягає в тому, що API виклики можуть тривати на протязі досить довгих періодів часу. Це дуже згубно позначається на UX кінцевих користувачів.

Виклад основного матеріалу. Порівняння роботи та швидкодії використання традиційного та розробленого підходів API запитів до хмарних систем управління базами даних.

Для порівняльного аналізу було використано одну з найпопулярніших хмарних систем управління базами даних — Salesforce. В якості клієнту використовується веб-застосунок розроблений на мові Ruby, з використанням Ruby on Rails фреймворку.

Принцип використання API сервісів хмарних систем управління базами даних складається з декількох етапів:

- Формування тіла запиту на боці клієнта;
- Виконання запиту використовуючи API сервіса;
- Отримання та опрацювання відповіді з боку API сервіса.

Очевидно, що вузьким місцем такого підходу є залежність від часу передачі тіла запиту, часу опрацювання запиту та часу отримання відповіді.

Пропонується підхід, за допомогою якого можна здійснювати різнобічну інтеграцію даних з хмарними системами управління базами даних в асинхронному режимі роботі. Для практичності використання програмний засіб має бути представлений у вигляді gem'у (пакет з бібліотекою).

Основними функціями, які повинні бути реалізовані в програмному продукті є:

- Створення фонового каналу інтеграції даних між системою управління хмарними базами даних додатком;
- Можливість задання різних видів стратегій синхронізації (постійна, пасивна, асоціативна);
- Можливість синхронізації окремих полів та асоціативних зв'язків в базі даних.

На основі конфігураційного файлу, з чітко визначеними координатами кінцевих точок, система повинна створювати фоновий канал зв'язку між локальною базою даних та хмарною базою даних.

Користувач створює локальну базу даних з ідентичними назвами таблиць, полів та їх типів до існуючих таблиць в хмарній базі даних з метою їх подальшої синхронізації.

Користувач спеціальною командою запускає скрипт створення каналу синхронізації даних. Система аналізує моделі, що підлягають синхронізації та порівнює вміст кожної бази даних. На основі аналізу, відбувається повна синхронізація даних між ними.

Користувач описує логіку синхронізації між локальною базою даних та хмарною базою даних за допомогою DSL в реляційних моделях. Система аналізує опис та здійснює синхронізацію за вказаною стратегією.

Описуючи логіку синхронізації в реляційних моделях, користувач має можливість вказати окремі поля в таблиці для синхронізації, а також вид асоціативного зв'язку з іншими таблицями в базі даних. Система аналізує опис, та здійснює синхронізацію асоціативних таблиць та окремих полів.

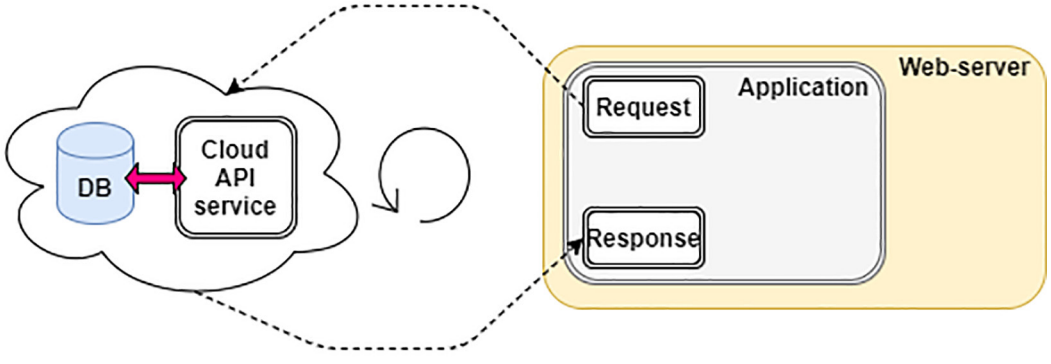

Рис. 1. Схематичне зображення використання API сервісу

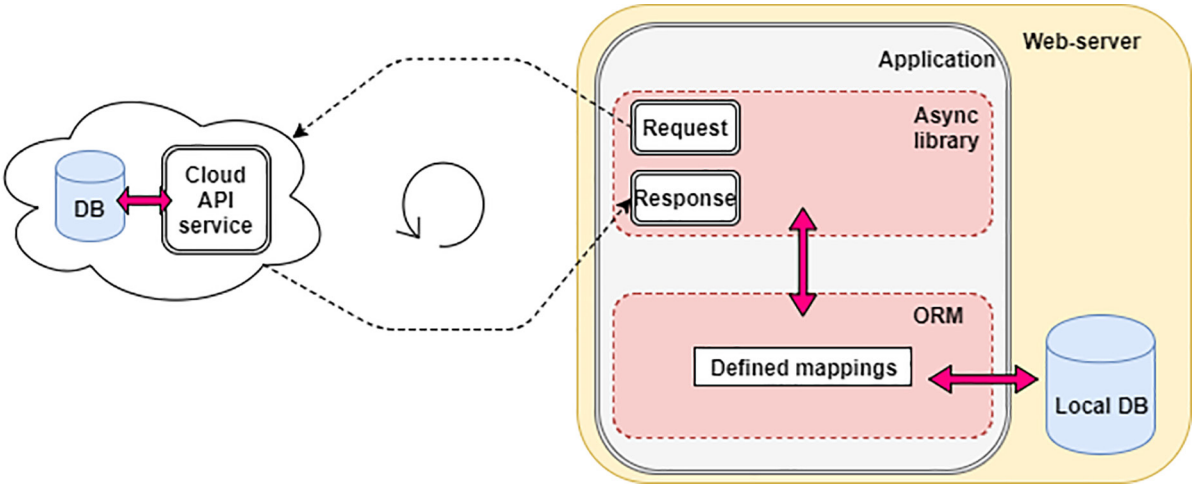

Рис. 2. Схематичне зображення використання API сервісу, використовуючи запропонований підхід

Для зручного доступу до даних використовується ActiveRecord, т.я. він реалізує популярний підхід об'єктно-орієнтованого проекціювання.

Запропонований підхід схематично відображений на рис. 2.

Висновок. З результатів можна констатувати усунення залежності від часу передачі тіла запиту, часу опрацювання запиту та часу очікування відповіді з віддаленого хмарного API сервісу управління базами даних. Залежність вдалось усунути завдяки використання локальної бази даних та асинхронного опрацювання складання та виклику API запитів.

Опис отриманих результатів. Після успішного запуску бібліотеки, використовуючи демонстраційний застосунок було проведено порівняння швидкодії звичайних API викликів та асинхронних API викликів.

Таблиця 1

Порівняльна таблиця типових операцій для різних типів API викликів

Оператор	Звичайний API виклик (sec)	Асинхронний API виклик(sec)
SELECT	1.52	0.02
UPDATE	3.57	0.039
INSERT	2.21	0.032
DELETE	2.71	0.0211

Таблиця 2

Порівняльна таблиця пришвидшення та ефективності різних типів API викликів

SELECT		
API type	Sync	Async
1	1,5123	0,012392
2	1,32342	0,074335
3	1,74356	0,020029
4	2,00016	0,087364
5	1,7722	0,024306
6	1,86553	0,03413
7	1,78625	0,041127
8	1,7971	0,027842
9	1,24168	0,017419
10	1,54275	0,064045
Min time, s	1,24168	0,012392
Speedup	1	100,2001
Efficiency	1	100,2001

UPDATE		
API type	Sync	Async
1	2353	0,06832
2	2,6438	0,03715
3	2,3843	0,07452
4	2,0123	0,05621
5	1,9982	0,09013
6	2743	0,05232
7	2,6285	0,06016
8	2,8478	0,01901
9	2348	0,07727
10	2,4395	0,02872
Min time, s	1,9982	0,01901
Speedup	1	105,113
Efficiency	1	105,113

INSERT		
API type	**Sync**	**Async**
1	2,78079	0,04799
2	2,28276	0,02063
3	2,71386	0,01444
4	2,24497	0,02318
5	2,67617	0,03801
6	2,40364	0,06328
7	2,89672	0,01639
8	2,29987	0,03267
9	2,50797	0,02989
10	2,81776	0,03140
Min time, s	2,24497	0,01444
Speedup	1	155,427
Efficiency	1	155,427

DELETE		
API type	**Sync**	**Async**
1	2,29231	0,02165
2	2,13412	0,02257
3	2,20325	0,02239
4	2,24966	0,02613
5	2,19296	0,03841
6	2,11411	0,03855
7	2,12011	0,03754
8	2,13678	0,03628
9	2,27608	0,03497
10	2,17742	0,03919
Min time, s	2,11411	0,02165
Speedup	1	97,6246
Efficiency	1	97,624600

В цих порівняннях було використано типові операції що виконуються в будь-якій хмарній або ж локальній базах даних.

Порівняння швидкодії було проведено для CRUD операцій:

- Select — оператор для реалізації вибірки даних.
- Update — оператор для реалізації модифікації даних.
- Insert — оператор для реалізації вставки даних.
- Delete — оператор для видалення даних.

Для кожної операції було виконано десять викликів та приведено до усередненого значення.

Висновок. Отже, провівши порівняльний аналіз швидкодії, пришвидшення та ефективності роботи синхронних та асинхронних типів API викликів зрозуміло, що асинхронний тип викликів, який створюється шляхом реалізації алгоритму синхронізації даних між локальною та віддаленою базами даних, є значно ліпшим.

Незважаючи на значне поліпшення показників швидкодії, пришвидшення та ефективності, алгоритм асинхронної синхронізації також має недоліки:

- Якщо є ризик перевищення лімітів API викликів цей підхід може бути непідходящим
- Можуть виникати race-conditions в тому випадку якщо обидві бази будуть оновлені під час одного й того ж пуллінг інтервалу.

Література

1. «What is Object / Relational Mapping?» Hibernate Overview. JBOSS Hibernate. — 2011.
2. «Measuring API Usability» Dr. Dobb's Retrieved 29 July 2016.