БД Cassandra

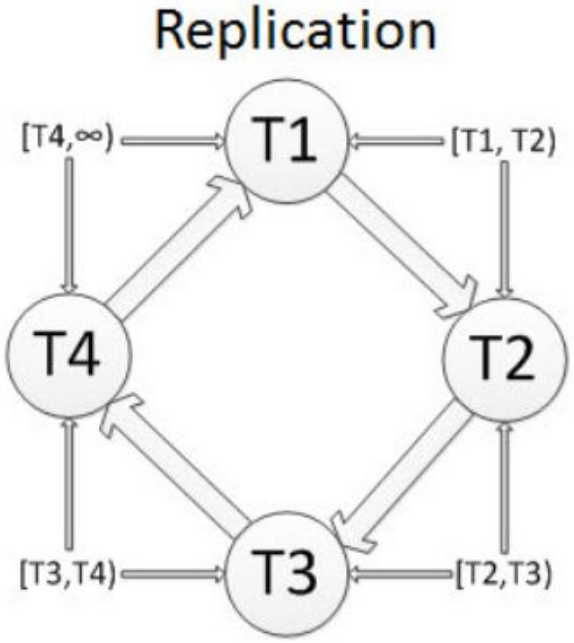
ПІДГОТУВАЛИ СТУДЕНТИ ПЗПІ-18-8

СОРОКІН ВОЛОДИМИР

ШПОРТА АРТЕМ

Архітектура Cassandra

- Основна ідея - це «кільце токенов». У нас є певна кількість серверів в кластері. Наприклад, їх 4, як на картинці. Ми кожного серверу призначимо токен. Це, грубо кажучи, певна кількість. Але спочатку ми визначаємо, які у нас взагалі, в принципі, бувають ключі.
- Припустимо, що у нас ключі 64-бітові. Відповідно, кожного серверу ми призначимо 64-бітний токен. Після цього ми їх збудуємо по колу, і згідно з цим відсортуємо токени. Кожен сервер у нас буде відповідати за якийсь із діапазонів токенов (Token Range).



Як влаштована реплікація?

- Ми вкажемо деякий основний сервер, який відповідає за якийсь діапазон токенов. Наприклад, сервер з токеном T1 відповідає за T1 і T2. Зазначимо, що наступний сервер буде відповідати за той же діапазон токенов. За кожен інтервал даних відповідають відразу два сервера. Приблизно так влаштована реплікація. Якщо у нас один сервер "падає", то дані, в принципі, доступні і зберігаються.
- У Cassandra політика реплікації настроюється.

Запис

- Клієнт не знає нічого про топологію кластера
- Команда на запис відправляється на довільний сервер, який стає координатором даної операції
- Координатор записує дані на потрібні сервера
- Клієнт вибирає критерій успішності записи
- Якщо одна з реплік недоступна, запис буде завершений пізніше

Як відбувається запис в Cassandra?

- Клієнт нічого не знає про те, як Ноди в кластері об'єднані в це кільце токенов. Він знає тільки список нод, які є в кластері, і посилає команду на запис довільного сервера. Можливо, використовується якесь балансування навантаження. Клієнт запам'ятовує, який сервер скільки відповідав, і посилає команду того сервера, який відповідає швидше. Це вирішується на клієнті.

Як відбувається зчитування?

- Приблизно так само, як і запис. Знову відправляється команда безпідставного сервера в нашому кільці, в нашому кластері. Сервер стає координатором даної конкретної операції читання. Він, знаючи про те, де і як розташовані дані, починає опитувати сусідні сервери і віддає дані клієнта.
- Клієнт визначає і критерій успішності читання. Він може сказати, що читання відбулося, коли дані були взяті хоча б з одного сервера реплікації. Або він може сказати, що читання відбулося, коли дані були зібрані з усіх серверів.

Що взято з Dynamo?

- З Dynamo взята структура кільця токенов. Повна распределенность, у нас немає ніякого ведучого вузла. Клієнт "спілкується" з усіма даними. Також реалізований алгоритм читання і запису.

Що взято з BigTable?

- З BigTable розробники рішення Cassandra взяли модель даних. На відміну від Amazon Dynamo, Cassandra не просто розподілений хеш. Там є колонки, ключі і значення у колонок. Взяли локальну структуру даних на серверах. Для Cassandra використовували локальну структуру на серверах, як у BigTable.

Як Cassandra виглядає зовні?

- Є таке поняття, як кластер. Це установка рішення Cassandra, це все наші Ноди. Для цього кластера налаштовуються різні параметри: як розподілені дані, яка у нас є структура і так далі.
- Ключове простір (англ. Keyspace) - це те ж саме, що база даних в термінології MySQL.
- Сімейство колонки (англ. Column family). Це те ж саме, що таблиця в термінах MySQL та інших стандартних баз даних.

Super columns

- На відміну від стандартної таблиці, де у вас є ключі і колонки, в Cassandra є ще один рівень, який називається суперколонкой. Колонки можна групувати.
- Ми зберігаємо відносини користувачів соціальної мережі, які дружать або не дружать. Вони можуть або дружити, або можуть бути підписані на оновлення сторінок один одного. Ми маємо такі суперколонкі: "Друзі" і "Підписники". Усередині них є колонки, в яких ми зберігаємо дані.

• Еще один уровень вложенности

Super columns	Friends		Subscribers	
Columns	Вова	Дима	Вова	Дима
Вова				•
Дима	•			

Які операції підтримуються в Cassandra?

- • Є операція "mutation", яка здійснюється, коли ми вказуємо, що з даного ключа в дану колонку ми повинні отримати дане значення. Чому операція називається "mutation"? Тому що Cassandra "не розрізняє" операції insert, update і інші. У нас є тільки така операція. Немає даних - значить, вони з'являться. Були старі дані - значить, вони перезапишуться.
- • Є операція "get". Ця операція дозволяє отримати значення з даного ключу і по даній колонці.

Які операції підтримуються в Cassandra?

- • Є операція "multi_get". Те ж саме з багатьох ключів.
- • Є операція "get_slice". Те ж саме, тільки можна відфільтрувати колонки, наприклад, по інтервалу або за належністю до якогось безлічі.
- • Є операція "get_range_slices", коли ми отримуємо щось не по конкретному ключу, а по інтервалу ключів.

Як Cassandra працює зсередини

- Команда йде на довільний сервер в кластері, і цей сервер стає координатором даної конкретної операції. Сервер знаходить потрібні репліки в залежності від настройки, яка називається "ReplicationStrategy".
- Якщо у нас "ReplicationFactor" дорівнює одиниці, тобто ми зберігаємо всі на одному сервері. Ми знаємо ключ і знаємо сервер, який відповідає за даний інтервал токенов.

Як Cassandra працює зсередини

- Відразу після того як прийшла команда записи на сервер, дані зберігаються локально в деякій таблиці. Цей сигнал називається "Hinted handoff". Навіщо це потрібно? На випадок, якщо у нас якісь сервери недоступні. Нам необов'язково з помилкою завершувати операцію запису, тому що клієнтові може бути не дуже важливо, щоб дані розклалися по всіх серверів відразу. Тоді сервер збереже дані локально, скаже, що все добре. Коли сервери знову почнуть функціонувати, дані будуть розподілені по всіх серверів реплікації.

Критерії успішності

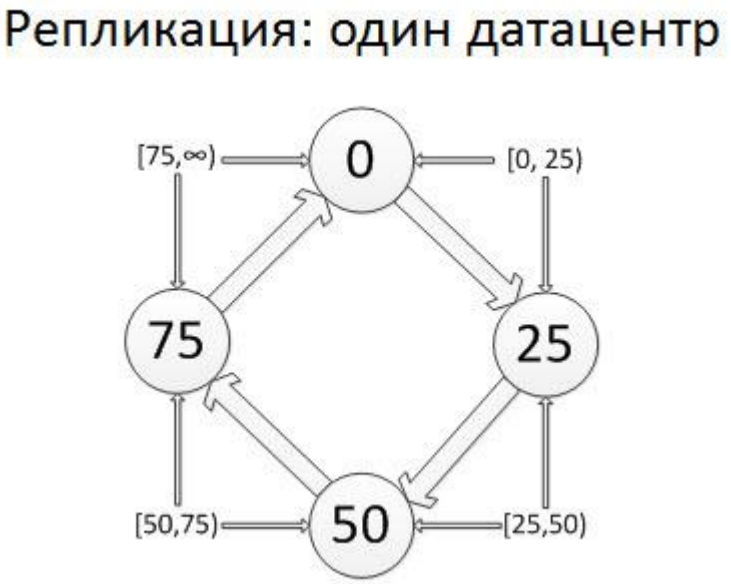
- Критерій успішності запису визначається клієнтом. Таких критеріїв 4.
- Перший критерій називається "ANY". Це означає, що запис визнається успішним, як тільки дані потрапляють на початковий сервер. Неважливо, розподілилися вони по серверам реплікації чи ні. Головне, що вони потрапили на наш довільний сервер, який є координатором.
- Є критерій "ONE". Як тільки дані потрапили хоча б на 1 сервер реплікації, вважається, що все записано успішно.

Критерії успішності

- Є критерій "QUORUM". Це "ReplicationFactor" поділити навпіл + 1.
- Є критерій "ALL". Запис визнається успішною після "розкладання" даних по всіх серверів реплікації, які відповідальні за даний ключ.

Як влаштована реплікація?

- Розглянемо простий випадок. У нас є 4 сервери, кожен відповідає за свій токен. У нас є сервер, який відповідає за нульовою. Припустимо, що у нас ключі від одного до ста. Сервер відповідає за нульовою, 25-й, 50-й, 75-й.
- Кожен сервер відповідає за два інтервалу ключів. Перший сервер відповідає як за інтервал ключів від 0 до 25 (за свій інтервал), так і за інтервал ключів попереднього сервера.

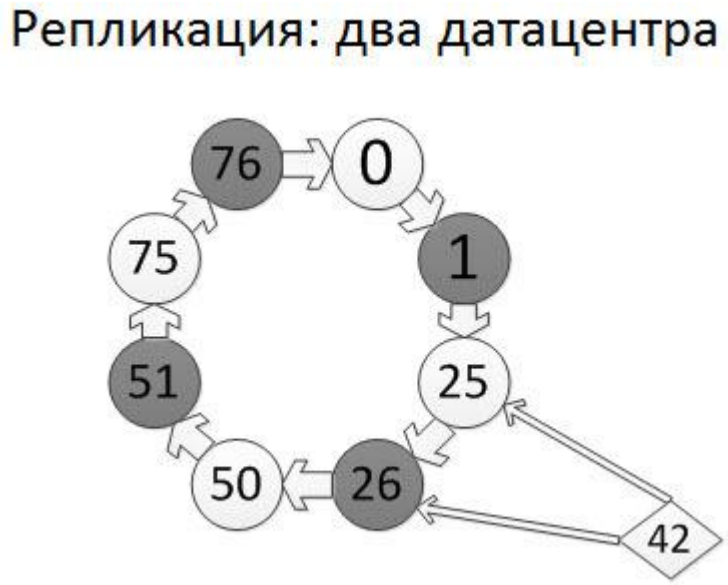


Як влаштована реплікація?

- Все це працює дуже добре, і не тільки всередині одного дата-центру. Одна з відмінних рис Cassandra - то, що це рішення може добре реплікувати дані між дата-центрами. Чому це важливо? Це може стати в нагоді тим, хто використовує Amazon, Esety та інші схожі сервіси. Наприклад, у того ж Amazon є така властивість, що цілий дата-центр може впасти. Якщо ми хочемо, щоб система працювала добре, непогано було б, щоб вона продовжувала працювати з другого дата-центру.

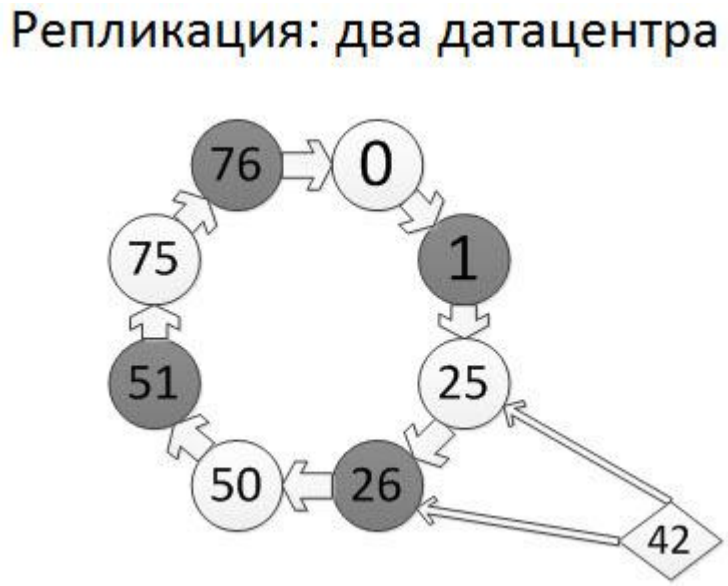
Як влаштована реплікація між дата-центрами?

- Це свого роду хак до структури Cassandra. Ідея така. У Cassandra є така вимога: у кожного сервера повинен бути свій унікальний токен.
- Призначимо серверів такі маркери. Виглядає це досить штучно. Перший сервер отримує 0, другий отримує 1. "Розфарбуємо" ці сервери в різні кольори. Припустимо, "білі" сервери у нас знаходяться в одному дата-центрі, а "сірі" - в іншому.



Як влаштована реплікація між дата-центрами?

- Налаштуємо реплікацію наступним чином: копія потрапляє на якийсь сервер, а наступна копія йде на сервер через один. Наприклад, згідно з таким налаштуванням, ключ із значенням 42 потрапить на 25-й і на 26-й сервер .
- Після цього можна "білі" ноди "відправити" в один дата-центр, "сірі" - в інший. Так буде працювати реплікація. Команда буде йти на довільний сервер. Оскільки копії "знають", де вони знаходяться, вони всі будуть здійснювати реплікацію між собою.

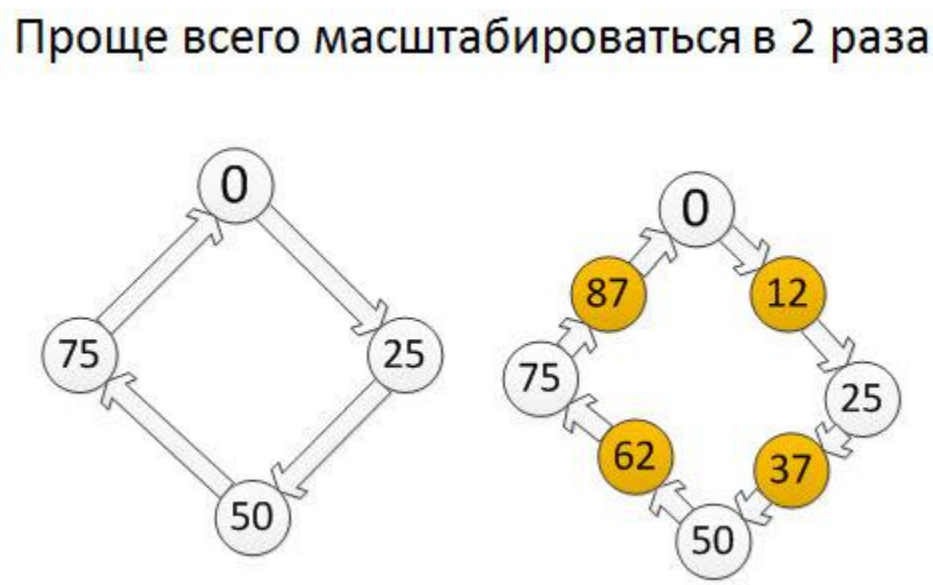


Масштабирование в Cassandra

- Ключова особливість Cassandra - це легке масштабування. У будь-який момент, завжди можна додати ще якусь кількість серверів в кластер. Це підноситься як суттєву перевагу.
- Як було сказано, у кожного сервера є певний токен, за який він відповідає. Є операція "move token". Можна перепризначити токен. Вказати, що цей сервер відповідає тепер за іншою діапазон. Це можна зробити прямо "на ходу". Відповідно, топологія, картинка з розподілом токенов і діапазонів зміниться. При цьому міграція відбуватиметься у фоновому режимі. Для клієнтів це все буде досить прозоро.
- Поки міграція не закінчена, сервер буде відповідати як за старий "шматок" даних, так і за новий. Аналогічно буде відпрацьована ситуація, коли ми додаємо щось в наше кільце токенов.

Масштабування в Cassandra

- Ця картинка ілюструє ідею масштабування. Найпростіше масштабуватися в 2 рази, як і всюди. Наприклад, у нас є 4 сервери, кожен з яких відповідає за токен 0, 25, 50 і 75 (кожен за свій діапазон). Між ними дуже легко вставити сервери, які позначені на зображенні помаранчевим, кожному призначити токен, який знаходиться рівно посередині між сусідами.
- Така операція в Cassandra дійсно відбувається безболісно. Ми додали ці сервери, вказали, що вони відповідають за ці маркери, додали їх в кластер. Все переноситься в фоновому режимі і продовжує працювати, як раніше.



Література

- https://dmkpress.com/files/PDF/978-5-97060-453-3.pdf
- Стаття в вікіпедії - https://ru.wikipedia.org/wiki/Apache_Cassandra
- Керівництво по Cassandra - https://proselyte.net/tutorials/cassandra/introduction/
- Кассандра - Коротке керівництво - **https://coderlessons.com/tutorials/bolshie-dannye-i-analitika/vyuchi-kassandru/kassandra-kratkoe-rukovodstvo**

Дякуємо за увагу!