УДК 004.65

А.В. Миргородський[1]
О.В. Романюк[1]

СТРАТЕГІЇ МАСШТАБУВАННЯ РОЗПОДІЛЕНИХ NOSQL СКБД

[1]Вінницький національний технічний університет

Анотація

Проаналізовано стратегії масштабування розподілених NoSQL систем керування базами даних; досліджено їх архітектурні та технічні особливості, проаналізовано зв'язок характеристик СКБД з механізмами масштабування та реплікації.

Ключові слова: СКБД, NoSQL, розподілена БД, масштабування розподілених систем, реплікація даних, фрагментація даних.

Abstract

The paper analyses strategies for scaling distributed NoSQL database management systems; investigates their architectural and technical features; and analyses the relationship between DBMS characteristics and scaling and replication mechanisms.

Keywords: DBMS, NoSQL, distributed database, distributed system scaling, data replication, data fragmentation.

Вступ

NoSQL бази даних сьогодні є невід'ємною частиною цифрових інформаційних систем. Хоча реляційні системи керування базами даних досі займають лідируючі позиції, популярність NoSQL СКБД продовжує зростати [1]. Більш того, NoSQL системи стали гнучким інструментом, що спрямовані на доповнення реляційних систем та дозволяють покрити більше різноманітних сценаріїв використання. Згідно статистики за 2023 рік, близько 49% розробників використовують одночасно і NoSQL, і реляційні СКБД [2].

Відсутність обмежень реляційних систем дозволяє використовувати більш комплексні стратегії масштабування та реплікації даних. Це в свою чергу робить можливим побудову більш ефективних розподілених інформаційних систем, що можуть показати кращу швидкодію в порівнянні з аналогічними реляційними СКБД [3]. Вивчення стратегій реплікації даних та масштабування NoSQL систем керування базами даних дозволить сформувати рекомендації та подальші напрями дослідження для розробки нових методів масштабування розподілених СКБД.

Метою дослідження є вивчення та порівняння стратегій масштабування розподілених NoSQL систем керування базами даних.

Об'єктом дослідження є процес масштабування розподілених NoSQL систем керування базами даних.

Предмет дослідження – стратегії масштабування розподілених NoSQL систем керування базами даних.

Основна частина

Основною відмінністю NoSQL баз даних від класичних реляційних є використання довільних структур даних, що не використовують реляційну алгебру в своїй основі. Термін NoSQL може включати в себе широкий спектр різних систем керування базами даних, що можуть бути як доволі універсальними, так і більш спеціалізованими під конкретні сценарії використання. До найбільш популярних типів NoSQL СКБД можна віднести: документо-орієнтовані, ключ-значення, графові, об'єктні та стовпчикові [4]. Варто зазначити, що деякі NoSQL системи є мультимодальними – тобто одночасно можуть працювати з даними в різних форматах, як-от ключ-значення та документо-орієнтованими.

Далеко не всі NoSQL бази даних є розподіленими – велика кількість БД типу ключ-значення використовуються в якості локального кешу, де швидкості доступу до даних надається пріоритет. Проте, розподілені СКБД також можуть відрізнятись за своїм характером використання. Класичне розуміння розподіленої системи передбачає запуск СКБД на декількох географічно віддалених серверах, що зазвичай або мінімізує ризик непрацездатності, або дозволяє зменшити затримки для локально-близьких клієнтів. В сучасних сценаріях використання розподілена база даних може також розгортатись в межах одного дата центра або ж хмарного провайдера заради покращення швидкодії в цілому.

Відповідно, в процесі дослідження стратегій масштабування розподілених NoSQL СКБД варто звернути увагу на системи різних типів – як за структурою даних, так і за механізмами глобальної реплікації даних. Тому, для аналізу було обрано популярні СКБД Apache Cassandra, MongoDB та Amazon DynamoDB.

Apache Cassandra є децентралізованою СКБД зі сховищем з широкими стовпчиками. Це підтип NoSQL бази даних типу ключ-значення, що передбачає подібну до реляційних систем структуру даних з використанням таблиць, але з довільними атрибутами для кожного запису. Децентралізованість Apache Cassandra означає, що всі вузли є рівноправними – зазвичай такі системи мають більші ліміти масштабування за рахунок відсутності єдиного лідера кластера [5].

При масштабуванні ця СКБД використовує фрагментацію даних – метод поділу єдиної таблиці або іншої структури даних на менші частини, що можуть бути розміщені на різних серверах розподіленої системи. Cassandra застосовує фрагментацію на основі хешування – ключ кожного запису хешується і отримане значення використовується для фактичного розміщення даних на вузлах кластера [5]. Аналогічно до деяких інших NoSQL СКБД, присутня можливість створення декількох реплік одного й того самого фрагмента даних, що покращує стійкість розподіленої системи до збоїв.

Окрім цих базових механізмів, що присутні в більшості розподілених систем керування базами даних, Apache Cassandra має окремі інструменти масштабування та відновлення. По-перше, для уникнення конфліктів в децентралізованій системі використовується розподіл обов’язків – так як дані розподілені на основі захешованих ключів та мають по декілька реплік, то можна визначити окремі вузли-координатори, що будуть керувати процесом реплікації при масштабуванні системи. Це дозволяє реалізувати механізм Hinted Handoff – вузол-координатор може автоматично оновити застарілі сервери-репліки або ж «полагодити» інший вузол після виникнення мережевих проблем та розсинхронізації реплікації [5]. Для більш комплексних сценаріїв застосування також підтримується режим групування серверів-реплік в окремі дата центри в залежності від географічного розташування, що дозволяє застосовувати більш складні політики та стратегії розміщення записів і реплікації.

MongoDB – це документо-орієнтована NoSQL база даних, що також використовує фрагментацію даних при масштабуванні розподіленої системи. Ця СКБД також підтримує розподіл записів на основі діапазону значень, а не тільки на основі захешованого значення ключа, але найбільш важливою відмінністю є загальна архітектура системи – розподілений кластер MongoDB працює в централізованому режимі [6]. Тобто, присутній лише один вузол, що керує усіма операціями запису та відслідковує їх послідовність – комбінація цих двох механізмів дозволяє досягти узгодженості даних на всіх вузлах системи. Для покращення швидкодії використовується асинхронна реплікація та є можливість обрати необхідний режим узгодженості для кожної операції окремо.

Для великих розподілених систем з географічно віддаленими серверами MongoDB також дозволяє виконати зональну фрагментацію, що включає в себе автоматичний розподіл записів на основі додаткових атрибутів та автоматичний роутинг клієнтських запитів [6]. Варто відмітити, що через централізовану архітектуру швидкодія виконання окремих запитів може значно відрізнятись в залежності від розміщення поточного лідера кластера, особливо при збільшенні масштабів системи.

Amazon DynamoDB – це пропрієтарна розподілена NoSQL база даних, що доступна в складі хмарного провайдера AWS. Однією з особливостей цієї СКБД є значний ступінь автоматизації процесу масштабування. Аналогічно до попередніх розглянутих систем, тут застосовується фрагментація даних для балансування навантаження та спрощення реплікації. Тип фрагментації співпадає з Apache Cassandra – це розподіл на основі захешованого ключа запису [7].

Автоматичне масштабування та реплікація даних в DynamoDB покладаються на архітектуру самого хмарного провайдера AWS. Amazon гарантує підтримку мінімум трьох реплік в окремих дата центрах в межах регіону, що передбачає стійкість до апаратних проблем та стихійних лих. Так як СКБД є пропрієтарною, то всі деталі архітектури не розкриваються, але відомо, що кластер DynamoDB

аналогічно до MongoDB має лідера та другорядні сервери-репліки. Також підтримуються різні моделі узгодженості, але механізм вирішення конфліктів при реплікації є більш простим – остання операція запиту завжди вважається більш коректною, тобто такою, що буде реплікуватись на інші сервери. Цей механізм називається Last Writer Wins (LWW) – аналогічний також використовується і в Apache Cassandra. Сам процес масштабування базується на відслідковуванні кількості клієнтських запитів.

Важливо також виділити глобальний режим масштабування Global Tables, що в DynamoDB відрізняється від Cassandra та MongoDB. AWS автоматично створює нові копії обраної таблиці в інших регіонах, але на відміну від звичайних реплік використовується Active-Active система – всі копії на рівні регіонів є активними й доступними для операцій і читання, і запису [8]. Така стратегія масштабування дозволяє досягти вищої швидкодії, проте механізм LWW може не підходити для певних типів інформаційних систем.

Підсумуємо всі описані та деякі додаткові характеристики розглянутих розподілених СКБД і їх стратегій масштабування в єдине порівняння, що наведене в таблиці 1.

Таблиця 1 – Порівняння розподілених NoSQL СКБД за стратегіями масштабування

Характеристика	**Apache Cassandra**	**MongoDB**	**Amazon DynamoDB**
Тип NoSQL БД	Сховище з широкими стовпчиками	Документо-орієнтована	Ключ-значення
Архітектура	Децентралізована	Лідер-репліка (Master-Slave)	Пропрієтарна, частково лідер-репліка
Механізми масштабування	Вузли-координатори, Hinted Handoff	Централізоване керування масштабуванням та реплікацією	Автоматичне масштабування на основі навантаження
Додаткові механізми глобального масштабування	Групування реплік за дата центром	Групування реплік за дата центром + гнучка фрагментація	Global Tables (Active-Active реплікація з LWW)
Фрагментація даних	Підтримується, на основі хешування	Підтримується, на основі хешування або діапазону значень	Підтримується, на основі хешування
Узгодженість даних	Контрольована, за запитом користувача		
Вирішення конфліктів узгодженості даних	LWW, обмежена підтримка транзакцій	Відслідковування порядку операцій, підтримка транзакцій	LWW, обмежена підтримка транзакцій

Порівняння дозволяє виділити подібні та відмінні особливості розглянутих NoSQL СКБД та їх стратегій масштабування. По-перше, всі вони використовують фрагментацію даних, що спрощує сам процес реплікації при масштабуванні та покращує стійкість СКБД до збоїв. Конкретний тип розміщення даних не впливає на сам механізм масштабування в жодному з розглянутих прикладів, проте він потенційно може впливати на швидкодію при обробці запитів. Аналогічно, використовувана структура даних, тобто тип NoSQL БД, також не впливає на інші особливості системи масштабування: Cassandra використовує повністю децентралізовану архітектуру з вузлами координаторами, в той час як MongoDB та DynamoDB мають різні варіації моделі лідер-репліка. Додаткові механізми глобального масштабування також значно відрізняються – Cassandra та MongoDB вміють групувати репліки, надаючи додаткову гнучкість в налаштуваннях, а DynamoDB використовує механізм Global Tables – потенційно швидший, але обмежений в функціональності. При цьому, всі три СКБД та їх моделі масштабування дозволяють за запитом користувача пожертвувати швидкодією заради кращих гарантій узгодженості даних.

Висновок

Було досліджено особливості роботи популярних NoSQL СКБД Apache Cassandra, MongoDB та Amazon DynamoDB. Розглянуто їх механізми роботи з даними при реплікації та масштабуванні.

Досліджено особливості глобального масштабування для великих розподілених систем на основі обраних СКБД. Проаналізовано спільні та відмінні характеристики розподілених систем.

Дослідження показало, що важливим елементом розглянутих СКБД та їх стратегій масштабування є реплікація даних із застосуванням фрагментації. При цьому, конкретна структура даних та тип їх розміщення не мають значного впливу на інші архітектурні аспекти – серед популярних розподілених систем використовується як більш класична модель лідер-репліка, так і повністю децентралізована модель. При цьому, швидкодія СКБД у великих кластерах може контролюватись додатковими механізмами глобального масштабування, різними моделями узгодженості даних та вирішення конфліктів. З урахуванням цього, для розробки нових та покращення існуючих стратегій масштабування розподілених СКБД варто розглянути інші особливості методів фрагментації даних. Також, для досягнення кращої потенційної швидкодії варто детальніше дослідити моделі підтримки узгодженості даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Krstić M. NoSQL Databases - Analysis and Directions of Further Development / Marija Krstić, Lazar Krstić, V. Stanković // 10th International Conference on Applied Information and Internet Technologies, Zrenjanin, 16 October 2020. – [S. l.]. – P. 203–208.
2. Ayub K. 2024 NoSQL Database Trend Report [Electronic resource] / Kamran Ayub // RavenDB NoSQL Database. – Mode of access: https://ravendb.net/whitepapers/2024-nosql-database-trend-report (date of access: 09.06.2025). – Title from screen.
3. Performance Comparison between NoSQL (RethinkDB) and MySQL Database Replication from Master to Slave in Big Data [Electronic resource] / Dildar Hussain [et al.] // Jurnal Kejuruteraan. – 2021. – Si4, no. 2. – P. 65–69. – Mode of access: https://doi.org/10.17576/jkukm-2021-si4(2)-10 (date of access: 09.06.2025). – Title from screen.
4. ElDahshan K. A. Data in the time of COVID-19: a general methodology to select and secure a NoSQL DBMS for medical data [Electronic resource] / Kamal A. ElDahshan, AbdAllah A. AlHabshy, Gaber E. Abutaleb // PeerJ Computer Science. – 2020. – Vol. 6. – P. e297. – Mode of access: https://doi.org/10.7717/peerj-cs.297 (date of access: 09.06.2025). – Title from screen.
5. Shirolkar A. A Comprehensive Guide to Apache Cassandra Architecture [Electronic resource] / Anup Shirolkar // Instaclustr. – Mode of access: https://www.instaclustr.com/blog/cassandra-architecture/ (date of access: 09.06.2025). – Title from screen.
6. Katz E. MongoDB Replica Set: A Developer's Tutorial to MongoDB Replication [Electronic resource] / Eyal Katz // Spectral. – Mode of access: https://spectralops.io/blog/mongodb-replica-set-a-developers-tutorial-to-mongodb-replication/ (date of access: 09.06.2025). – Title from screen.
7. A Deep Dive into Amazon DynamoDB Architecture [Electronic resource] // ByteByteGo Newsletter | Alex Xu | Substack. – Mode of access: https://blog.bytebytego.com/p/a-deep-dive-into-amazon-dynamodb (date of access: 09.06.2025). – Title from screen.
8. Awad J. W. DynamoDB Global Tables [Electronic resource] / Joud W. Awad // Medium. – Mode of access: https://medium.com/@joudwawad/dynamodb-global-tables-e62f2dce5f76 (date of access: 09.06.2025). – Title from screen..

Миргородський Андрій Вікторович – аспірант групи 121-24а, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: mirgorodskijav@gmail.com

Романюк Оксана Володимирівна – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: romaniukoksanav@gmail.com

Myrhorodskyi Andrii – graduate student of group 121-24a, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: mirgorodskijav@gmail.com

Oksana Romaniuk – Candidate of Technical Sciences, Associate Professor of the Software Chair, Vinnytsia NationalTechnical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com