DOI 10.31891/2307-5732-2023-321-3-37-42
УДК 637.5.02

БЕЛОУС РОМАН
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"
ORCID ID: 0000-0002-7588-941X
e-mail: belous22@ukr.net
КРИЛОВ ЄВГЕН
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"
ORCID ID: 0000-0003-4313-938X
e-mail: ekrylov1964@gmail.com

ОПТИМІЗАЦІЯ ЧАСУ ПРОЦЕСУ УЗГОДЖЕНОСТІ ДАНИХ В NOSQL

Наголошено, що продуктивність розподіленої інформаційної системи залежить від багатьох факторів, включаючи розмір і структуру бази даних, використовуване обладнання, кількість реплік та їх географічне розташування. Зроблено висновок про перспективність концепції нереляційних баз даних NoSQL та зроблено висновок, що була запропонована для ефективного зберігання та забезпечення швидкого доступу до великих обсягів інформації, т. зв. Big Data, що неможливо досягти використовуючи традиційні системи управління реляційними базами даних. Розкрито сутність поняття «база даних NoSQL» та названо існуючі на даний час труднощі у користуванні великих систем.

В базах даних NoSQL для забезпечення високої відмовостійкості використовується багаторазова реплікація (копіювання) запису. Але для них існує суттєвий недолік, адже в цих системах не підтримується режим ведення транзакцій і блокувань, тому виникає проблема узгодження даних. Наголошено, що у базах даних NoSQL в основному використовуються два способи розміщення і оновлення часу процесу узгодженості даних в NoSQL: за принципом «головний-підлеглий» (master-slave) і «по кільцю» (ring).

Описана модель сильного узгодження даних в NoSQL та розглянуто приклад оптимізації часу для онлайн журналу «eClass». Встановлено, що навіть при великих значеннях інтенсивності надходження вимог на читання λ час очікування не перевищує 3 мс. Однак при виконанні аналітичних запитів цей час може зрости, так при великих N і великої інтенсивності вхідних вимог на читання цей час може досягати 7 мс.

Ключові слова: Бази даних, NoSQL, час очікування, Big Data, масив даних, реляційна база.

BELOUS ROMAN, KRYLOV IEVGEN
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

TIME OPTIMIZATION OF PROCESS OF DATA CONSISTENCY IN NOSQL

It is emphasized that the performance of a distributed information system depends on many factors, including the size and structure of the database, the hardware used, the number of replicas and their geographical location. A conclusion was made about the perspective of the concept of non-relational NoSQL databases and it was concluded that it was proposed for efficient storage and provision of quick access to large volumes of information, the so-called Big Data, which cannot be achieved using traditional relational database management systems. The essence of the concept of "NoSQL database" is revealed and the currently existing difficulties in using large systems are named.

In NoSQL databases, multiple replication (copying) of a record is used to ensure high fault tolerance. But there is a significant drawback for them, because these systems do not support the mode of conducting transactions and blocking, so there is a problem of data reconciliation. It is emphasized that NoSQL databases mainly use two methods of placing and updating the time of the data consistency process in NoSQL: according to the principle of "master-slave" and "ring". A model of strong data reconciliation in NoSQL is described and an example of time optimization for the healthcare system is considered. It has been established that even with large values of the intensity of receiving requests for reading λ, the waiting time does not exceed 3 ms. However, when performing analytical queries, this time can increase, so with large N and high intensity of incoming read requests, this time can reach 7 ms.

Keywords: databases, NoSQL, waiting time, Big Data, data array, relational database.

Постановка проблеми

Існуючі на даний час комп’ютерні системи щоденно оперують величезними обсягами даних в мережі і значна їх частина опрацьовується системами управління реляційними базами даних (RDBMS). Функціонування таких баз даних полегшує моделювання даних та прикладне програмування за рахунок чіткої математичної основи [5].

Концепція нереляційних баз даних (БД) NoSQL була запропонована для ефективного зберігання та забезпечення швидкого доступу до великих обсягів інформації, т. зв. Big Data, що неможливо досягти використовуючи традиційні системи управління реляційними базами даних. Технології Big Data є однією з найбільш швидкозростаючих сфер інформаційних технологій, адже загальний обсяг оброблюваних даних подвоюється кожні 1-2 роки [1]. Відповідно до прогнозів і тенденцій ринку до 2024 року, обсяг ринку Big Data зросте в 2.4 рази щодо 2014 року і складе 68.7 млрд $, отже, потреба в базах даних NoSQL буде зростати.

Поряд із значним прогресивним розвитком великих систем розробники стали відчувати такі труднощі як: 1) ускладнилася процедура агрегування даних, тому що такий процес включає прочитку записів з великою кількістю пов'язаних таблиць (виникла проблема втрати відповідності), 2) виникає протиріччя між необхідністю зберігання великих обсягів неструктурованих даних і необхідністю їх структурувати за допомогою розробки схеми бази даних, 3) для зберігання великих обсягів інформації потрібно купувати спеціалізовані програмні комплекси паралельних систем баз даних (Teradata, Sun Oracle

database Machine), 4) за наявності великої кількості вузлів виникає проблема забезпечення необхідної відмовостійкості системи. Як спроба вирішити накопичені проблеми реляційних баз даних з'явилися альтернативні засоби зберігання і обробки даних, що отримали назву «бази даних NoSQL» [10].

Аналіз останніх джерел

Оптимізація часу процесу узгодженості даних в NoSQL є актуальною темою для багатьох дослідників та розробників баз даних. В останні роки було опубліковано багато статей та досліджень на цю тему. Нижче наведено аналіз останніх джерел публікацій на цю тему.

1. "Optimizing Data Consistency and Availability in NoSQL Databases" (2020) авторів Akhil Arora та інших. У цій статті автори розглядають питання забезпечення консистентності та доступності даних в NoSQL базах даних. Вони пропонують підхід, який дозволяє забезпечити високу консистентність даних без втрати доступності. Автори проводять експерименти на різних NoSQL базах даних та демонструють ефективність свого підходу.

2. "Efficient Data Consistency for NoSQL Databases" (2021) авторів M. M. Ahsan та інших. У цій статті автори розглядають питання забезпечення ефективної консистентності даних в NoSQL базах даних. Вони пропонують новий підхід до вирішення цього питання, який забезпечує швидку консистентність даних та зменшує навантаження на систему. Автори проводять експерименти на різних NoSQL базах даних та демонструють ефективність свого підходу.

3. "Optimizing the Consistency and Latency Tradeoff in NoSQL Databases" (2022) авторів M. A. Shaikh та інших. У цій статті автори розглядають питання забезпечення оптимального балансу між консистентністю та затримкою в NoSQL базах даних. Вони пропонують новий підхід до вирішення цього питання, який дозволяє зменшити затримки та зберігати високу консистентність даних.

Метою роботи є аналіз шляхів оптимізації часу процесу узгодженості даних в NOSQL.

Виклад основного матеріалу

В даний час під поняття «база даних NoSQL» потрапляє дуже широкий спектр систем зберігання і обробки даних: сховища типу Riak, Redis, DynamoDB, Project Voldemort, сховища сімейств стовпців (HBase, Apache Cassandra, HyperTable), документо-орієнтовані (CouchDB, MongoDB, eXist), бази даних на основі графів (Neo4j, OrientDB, InfiniteGraph). Бази даних на основі графів в даному дослідженні ми не розглядали, тому що вони не є розподіленими (реплікується весь граф). Крім того, вплив стійкості до втрати зв'язності ми не розглядали, адже вважається, що NoSQL-система повинна володіти цією властивістю [6].

В БД NoSQL для забезпечення високої відмовостійкості використовується багаторазова реплікація (копіювання) запису. Але для них існує суттєвий недолік, адже в цих системах не підтримується режим ведення транзакцій і блокувань, тому виникає проблема узгодження даних. Узгодженість даних означає, що всі екземпляри програми представлені з однаковим набором значень даних весь час. Підтримка необхідного рівня узгодженості для кожної конкретної предметної області може регулюватися параметрами (N, W, R) [3]. Однак, як випливає з теореми Брювера [4], зміна параметрів з метою забезпечення гарантій узгодженості впливатиме на доступність до системи. Оцінка та порівняння оптимізації часу процесу узгодженості даних в NoSQL є актуальним напрямом наукових та практичних досліджень. З цією метою більшість з наявних робіт за цим напрямком використовують засоби навантажувального випробування (наприклад, Yahoo! Cloud Serving Benchmark, YCSB).

Важливими показниками для оптимізації часу процесу узгодженості даних в NoSQL є ймовірність читання застарілого запису за час поширення оновлень по вузлах системи, час очікування початку читання запису з оновленого кворуму серверів, число версій запису в базі даних NoSQL і час їх обробки, ймовірність відмови в доступі до запису БД. Це дозволяє уникнути ручного підбору значень необхідних параметрів для великого числа типів записів БД на етапі налагодження системи і необхідності натурного моделювання екстремального навантаження на систему [2].

У базах даних NoSQL в основному використовуються два способи розміщення і оновлення часу процесу узгодженості даних в NoSQL: за принципом «головний-підлеглий» (master-slave) і «по кільцю» (ring). Перший спосіб має на увазі зберігання даних на master-вузлі і їх реплікація на slave-вузли. Всі зміни виконуються на master-вузлі, і вони зберігаються в пам'яті цього вузла. Slave-вузли періодично опитують master вузол, читають накопичені зміни і зберігають їх у своїй пам'яті. Прикладами таких баз даних є Mongodb, HBase, Neo4j. При розміщенні даних «по кільцю» необхідно визначити, скільки в ньому буде секцій (v-вузлів). Ці секції розподіляються по серверах (по кільцю).

Ще одним методом вирішення конфліктів є ведення версій записів, реалізована за допомогою вектору часу (Vector Clock - VC) [11]. Вектор часу – це послідовність пар, яка описує порядок поновлення цього запису. На рис. 1 показаний приклад ведення вектора часу (в дужках показаний вектор часу запису). Запис D (наприклад, якийсь документ) оновлюється користувачами A1, A2, A3.

Перші два поновлення виконуються користувачем A1 послідовно. Далі користувачі A2, A3 одночасно читають і оновлюють цей запис (випадковий збіг). У базі даних зберігаються дві версії запису: D1 і D2. При читанні документа D користувачеві A1 повертаються дві версії запису з одним і тим же ключем (D1 і D2). Він вносить зміни, наприклад, поєднує в собі оновлення, виконані користувачами A2, A3. У базі зберігається одна узгоджена версія запису з вектором часу, що включає ідентифікатори трьох користувачів. Перевагами вектору часу є відсутність єдиної точки відмови системи, тому що при використанні тимчасових міток в записах необхідно виконувати точну синхронізацію часу з одним еталоном. Недоліками вектору

часу є відсутність можливості автоматично вирішувати конфлікти, а також збільшення довжини вектору часу при багаторазовому оновленні запису. Однак в NoSQL існують механізми урізання вектору часу. Наприклад, в системі Riak можна задавати частоту обрізання вектору на рівні сегмента, а також максимальний розмір (довжину) вектора часу [8].

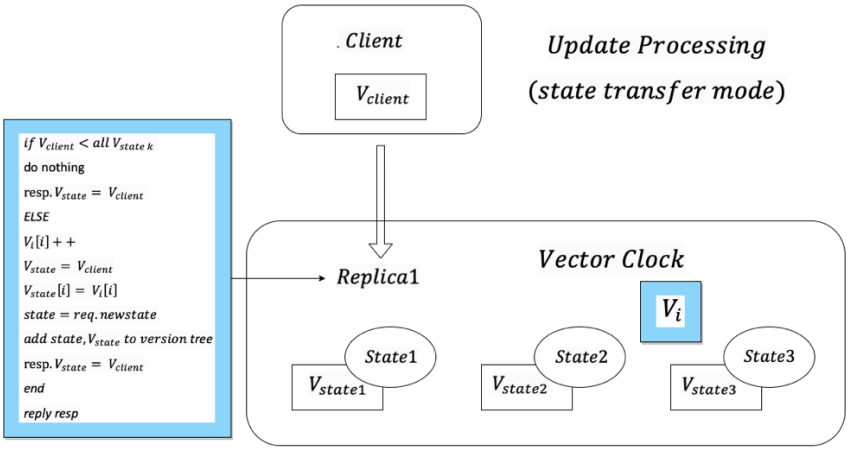

Рис. 1. Схематичне зображення способу реалізації методу ведення вектору часу

Наведемо приклади неузгодженості даних, що призводять до появи кількох версій запису, і способи їх усунення: кілька співробітників одночасно коригують примірник одного і того ж документу (записи) – наприклад, пропозиція щодо розвитку підприємства; дозвіл неузгодженості – об'єднання (вибір) коригувань керівником групи; кілька експертів одночасно оцінюють примірник одного і того ж документа – наприклад, blind-оцінка статті, поданої на конференцію; дозвіл неузгодженості – вибір оцінки головою комісії; кілька співробітників одночасно працюють з одним примірником документа, причому один співробітник може бачити результати роботи іншого співробітника – наприклад, кілька авторів працюють над однією статтею; може з'явитися кілька версій записів, неузгодженість усуває керівник групи авторів; користувачі обговорюють будь-яку одну подію в блозі.

З наведених вище прикладів видно, що забезпечення узгодженості часу для бази даних дуже важливим для стабільної і правильної роботи системи. Одночасна робота призводить до конфліктів, які можуть бути вирішені, наприклад, керівником групи на основі вектору часу в процесі узгодженості даних. Знання оцінки вікна неузгодженості сприяло б у цій ситуації запобігання конфліктів [9].

У випадку багатофазної обробки неузгодженості бути не повинно, тому що вихідні дані однієї фази використовуються як вхідні дані інший. Знання затримки реакції системи при забезпеченні суворої узгодженості дозволяє виконати точну оцінку часу виконання всього багатофазного завдання. При багатофазній обробці запиту за технологією MapReduce вихідні дані однієї фази є вхідними даними іншої фази, наприклад, при виконанні операції Join при доступі до структурованих даних або при виконанні складних запитів в пошукових системах (репліки вхідних даних наступної фази можуть бути неузгоджені); цю неузгодженість усуває NoSQL (узгодженість, заснована на потенційному причинно-наслідковому зв'язку).

При сильній (суворій) узгодженості даних в NoSQL в кожен момент часу гарантується читання останніх оновлень (досягається за рахунок вибору параметрів $W = R = N/2 + 1$). При слабкій узгодженості гарантується, що всі дані будуть ідентичні в кінцевому рахунку (параметри $W = R = 1$). Гарантії сильної узгодженості при оптимізації часу очікування досягаються шляхом вибору параметрів W і R таким чином, щоб їх сума перевищувала N. У цьому випадку множина реплік, з яких необхідно виконати читання запису БД для завершення операції, і безліч реплік, на яких необхідно виконати оновлення запису, щоб вважати цю операцію завершеною, завжди матимуть непорожню множину в своєму перетині. Отже, гарантується читання актуальної записи хоча б з однієї репліки, оскільки координатор не буде читати записи з реплік в цьому перетині, поки не завершиться їх оновлення. В даному дослідженні розглянемо випадок $W = R = N / 2 + 1$.

На рис. 2 представлена модель узгодження даних для розглянутого випадку. Вхідний потік вимог на читання з однієї репліки приймається пуасонівським з параметром λ. На рис. 2 оновлювані репліки позначаються 1, 2,..., $N/2+1$, буквою K позначений координатор; $\Psi_i(s)$ – перетворення Лапласа-Стилтьєса функції розподілу ймовірностей часу оновлення i-ї репліки

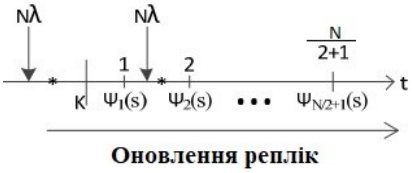

Рис. 2. Схематичне зображення моделі сильної узгодженості реплік ($W = R = N/2 + 1$)

Вимоги на читання надходять з сумарною інтенсивністю $N\lambda$. Вимоги можуть надійти як в процесі оновлення W реплік, так і до початку оновлення. Завдання полягає в тому, щоб оцінити час очікування на читання можливого закінчення оновлення ($N/2 + 1$)-ї репліки після читання $N/2$ реплік. Читання $N / 2$ реплік може завершитися у випадковий момент часу. Для вирішення поставленого завдання спочатку пропонується отримати виробленою функцією (ВФ) Wq (z) кількості тих вимог на читання запису БД з N реплік, які надійдуть за час оновлення ($N/2 + 1$) реплік. Використовуючи такий підхід можна показати, що необхідна ВФ кількість вимог на читання з N реплік, що надійшли за час оновлення i-ї репліки, рівна $\Psi_i(\lambda N(1 - z))$. Так як вимоги на читання запису БД з кожної репліки надходять незалежно один від одного, ВФ кількості вимог на читання з N реплік, що надійшли за час оновлення ($N/2 + 1$) реплік, буде дорівнювати [12]:

$$W_q(z) = \prod_{i=1}^{N/2+1} \Psi_i(\lambda N(1 - z)), \tag{1}$$

Тоді ймовірність того, що за час оновлення ($N/2 + 1$) реплік не надійде жодної вимоги на читання, дорівнює $Wq(0)$. Далі, якщо за деякий часовий інтервал надійшло $n \geq 1$ вимог, то моменти надходження цих вимог всередині даного інтервалу незалежні і розподілені за рівномірним законом. Це дозволяє зробити висновок, що будь-яка з вимог має одну і ту ж функцію розподілу ймовірностей часу очікування закінчення оновлення ($N/2 + 1$)-ї репліки. Розглянемо інтервал між початком і закінченням оновлення $W = N/2 + 1$ реплік як інтервал між подіями потоку подій ζ1, ζ2, ζ3... з однаковими функціями розподілів ймовірностей інтервалів між цими подіями. Тоді завдання дослідження зводиться до того, щоб оцінити час $t0$ між довільним моментом (в силу рівномірності надходження вимог всередині інтервалу) і моментом настання наступної події ζi (рис. 3):

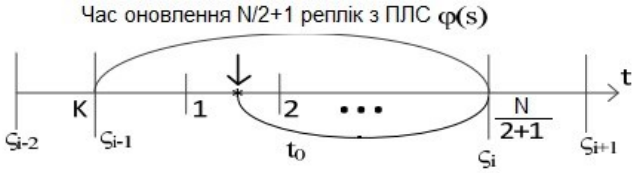

Рис. 3. Схема оцінки часу очікування *t*0

ПЛС Ф (s) часового проміжку t0 можна вивести в такому вигляді:

$$\Phi(s) = \frac{a}{s}(1 - \varphi(s)), \tag{2}$$

де α – середня кількість подій потоку в одиницю часу; φ(s) – перетворення Лапласа-Стілтьєса (ПЛС) функції розподілу ймовірностей часу між подіями потоку. Оскільки α = -1 / φ'(0), формулу (2) можна переписати наступним чином:

$$\Phi(s) = \frac{1}{-s^* \sum_{i=1}^{N/2+1} \Psi_i(0)} \left(1 - \prod_{i=1}^{N/2+1} \Psi_i(0)\right), \tag{3}$$

де $\Psi_i(s)$ – перетворення Лапласа-Стілтьєса розподілу ймовірностей часу оновлення i-ї репліки [2].

ПЛС Ω (s) функції розподілу ймовірностей часу очікування вимогою на читання закінчення оновлення ($N/2 + 1$) - ї репліки (після читання $N / 2$ реплік) має наступний вигляд:

$$\Omega(s) = W_q(0) + (1 - W_q(0))^* \Phi(s), \tag{4}$$

де Wq (0) і Ф (s) визначаються виразами (3) та (4) і їх можна трактувати так: з ймовірністю Wq (0) того, що за час оновлення ($N/2 + 1$) реплік не надійде жодної вимоги на читання з n реплік, ПЛС дорівнює одиниці (час очікування дорівнює нулю); з ймовірністю (1-Wq (0)) того, що за час оновлення ($N/2 + 1$) реплік надійде хоча б одна вимога на читання, ПЛС часу очікування будь-яким з них закінчення оновлення реплік дорівнює ПЛС, який розраховується за формулою (4).

Проведемо оцінку середнього часу очікування в процесі узгодженості даних для ($N / 2 + 1$) реплік. Диференціюючи вираз по s в нулі, отримаємо математичне очікування часу очікування ($N/2 + 1$) репліки. Але Ф(s) і $\Psi_i(s)$ – складні функції, і ручне диференціювання є трудомістким завданням. Для отримання математичного очікування можна використовувати методи чисельного диференціювання і отримані результати представити так:

$$-\Omega(0) \approx \frac{-(\Omega_{\frac{1}{2}}^1 - \frac{1}{2}\Omega_1^2)}{h}, \tag{5}$$

де Ω im – різниця m-го порядку (i – ціле при парному m; i – напівціле при непарному m); h – крок таблиці різниць. Формули для розрахунку Ωim мають вигляд:

$$\Omega_i^m = \sum_{j=0}^{m} (-1)^j C_m^j \Omega_{i+m/2-1} \tag{6}$$

$$\Omega_i = \Omega(\text{ih}) \tag{7}$$

де C_m^j – кількість поєднань з *m* по *j*.

Однак безпосередньо вирази (6) та (7) використовувати не можна, адже при чисельному їх диференціюванні на одній з ітерацій отримуємо невизначеність виду 0/0. Для розкриття невизначеностей використовують правило Лопіталя. Для вираження початкових моментів з врахуванням виразу Ф(1)(0) отримаємо таке математичне очікування шуканої величини:

$$\mathrm{M}_\Omega = -\Omega^{(1)}(0) = -\left(1 - W_q(0)\right) * \frac{\varphi^{(2)}(0)}{2 * \varphi^{(1)}(0)} \tag{8}$$

де (-1)*jφ(j)* (*s*) – *j*-й початковий момент ПЛС φ (*s*). Для отримання початкових моментів функції φ (*s*) можна використовувати методи чисельного диференціювання.

В якості прикладу проведемо аналіз шляхів оптимізації часу процесу узгодженості даних в NoSQL для системи електронного журналу «eClass», для закладів середньої освіти, яка може працювати на території Київської області, середня кількість записів(оцінок), для всіх навчальних закладів середньої освіти можна оцінити як 128.4 млн.

Згідно реєстром суб'єктів освітньої діяльності Київської області всього 770 навчальних закладів середньої освіти, тому це 770 точок входу в систему. Середню інтенсивність поновлення будь-якого запису можна встановити в 0.0012 1 / с (1 раз в 13.5 хвилин). Середній розмір агрегату з повного опису специфікацій розглянутих сегментів дорівнює 3417 байтів. На одному вузлі зберігається 128.4 млн /15 = 8.56 млн записів, середній обсяг записів на одному вузлі - 8.56 млн ≈ 3417 байтів= 27.25 Гбайт.

Продуктивність локальної мережі всередині сегменту мережі приймаємо 100 Мбіт / с; інтенсивність передачі даних по шині локальної мережі μn = 12.5·106 (байт/с). Продуктивність мережі між її сегментами становить 32 Мбіт / сек; інтенсивність передачі даних по шині мережі, що з'єднує підмережі μns = 4·106 (байт/с). Результати оцінки зміни часу процесу узгодженості даних в NOSQL наведено в таблиці 1.

Таблиця 1

Результати оцінки зміни часу процесу узгодженості даних в NOSQL

N =3	N = 4	N=5	N=6	N = 7
0,22	0,33	0,36	0,61	0,77
0,34	0,5	0,55	1,25	1,42
0,41	0,73	0,78	1,73	1,91
0,45	0,91	1,16	2,3	2,44
0,49	1,22	1,42	2,67	2,77

Розглянемо проблеми узгодженості, які можуть виникнути при роботі з системою:

1. Може виникати затримка читання записів з сегменту бази даних «eClass», тому що в період виставлення оцінок в кінці кожного семестру інтенсивність читання і оновлення даних сегменту висока.

2. При високій інтенсивності читання даних з сегменту бази даних «eClass» користувачам може бути повернута неактуальна інформація (застарілі записи), тому що даний сегмент узгоджений в кінцевому рахунку.

Висновки

Таким чином, ми встановили, що навіть при великих значеннях інтенсивності надходження вимог на читання λ час очікування не перевищує 3 мс. Однак при виконанні аналітичних запитів цей час може зрости, так при великих *N* і великої інтенсивності вхідних вимог на читання цей час може досягати 7 мс. Також слід зазначити, що при збільшенні кількості реплік *N* значення часу очікування вимогою на читання все менше залежить від λ, що підтверджує рівномірність розподілу моментів надходження вимог на читання всередині інтервалу оновлення баз даних.

За результатами проведених досліджень був запропонований шлях оптимізації часу процесу узгодженості даних в NOSQL.

Література

1. Борис Т. В. Управління великими масивами даних в якості послуги. Проблеми телекомунікацій. Київ. 22-25 квітня. 2014. С. 243–245.

2. Брацький В. О. Дослідження особливостей застосування реляційних і нереляційних баз даних на прикладі SQL Server та MongoDB. Наукові праці Національного університету харчових технологій. 2016. Т. 22. № 2-5. С. 15–24.

3. Григор'єв Ю. А. Аналіз характеристик узгодження реплік в базах даних NoSQL. Інформатика та системи управління. 2014. № 3. С. 3–11.

4. Крашівський А. І. Розробка веб-системи з використанням NODE.JS та MONGODB на прикладі

системи автоматизації HR-процесів. Комп'ютерні інформаційні технології. Тернопіль : ЗУНУ, 2020. С. 23–24.

5. Спасітєлєва С. О. Комплексний захист гетерогенних корпоративних сховищ даних. Сучасний захист інформації: науково-технічний журнал. 2017. № 1(29). С. 58–65.

6. Цвященко О. В. Аналіз адекватності моделі узгодження реплік в базах даних NosQL. Інформаційні технології. 2015. Т. 21. № 11. С. 840–848.

7. Joshi S. Balanced Load in Distributed System with NoSQL Middleware. International Journal of Emerging Technologies and Innovative Research. 2019. № 6(5), pp. 133–137.

8. Gyorodi C., Gyorody R., Andrada I., Livia B. A Comparative Study Between the Capabilities of MySQl Vs. MongoDB as a Back-End for an Online Platform. IJACSA. 2016. №1. С. 73–38.

9. Gorbenko A., Romanovsky A. Interplaying Cassandra NoSQL consistency and performance: A benchmarking approach. Communications in Computer and Information Science, Vol. 1279 / Editors: S. Bernardi, et al. Springer Nature. Berlin, 2020. pp. 168–184.

10. Karniavoura F. and Magoutis K. A measurement-based approach to performance prediction in NoSQL systems. 25th IEEE Int. Symposium on the Modeling, Analysis, and Simulation of Computer and Telecom. Systems, Banff, Canada, 2017. pp. 255–262.

11. Klein J., Gorton I., Ernst N., Donohoe P. Pham Performance Evaluation of NoSQL Databases: A Case Study. Proceedings of the 1st ACM/SPEC Int. Workshop on Performance Analysis of Big Data Systems, Austin, USA, 2015. pp. 5–10.

12. Kumar S. Comparison of NoSQL Database and Traditional Database-An emphatic analysis. JOIV: International Journal on Informatics Visualization. 2018. № 2. P. 51–55.

13. Nayak A., Poriya A., Poojary D. Type of NOSQL Databases and its Comparison with Relational Databases. International Journal of Applied Information Systems. № 5 (4). 2013. С. 16–19.

14. NoSQL Database Classification: New Era of Databases for Big Data. International Journal of Knowledge-Based Organizations (IJKBO). 2019. № 9. P. 50–65.

15. Rao T. R. The big data system, components, tools, and technologies: a survey. Knowledge and Information Systems. 2018. P. 71–81.

16. Banothu N., Bhukya S. and Sharma K. (2016). Big-data: Acid versus base for database transactions. 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). URL: https://ieeexplore.ieee.org/document/7755401.

References

1. Borys T. V. Upravlinnia velykymy masyvamy danykh v yakosti posluhy. Problemy telekomunikatsii. Kyiv. 22-25 kvitnia. 2014. S. 243–245.

2. Bratskyi V. O. Doslidzhennia osoblyvostei zastosuvannia reliatsiinykh i nereliatsiinykh baz danykh na prykladi SQL Server ta MongoDB. Naukovi pratsi Natsionalnoho universytetu kharchovykh tekhnolohii. 2016. T. 22. № 2-5. C. 15–24.

3. Hryhoriev Yu. A. Analiz kharakterystyk uzghodzhennia replik v bazakh danykh NoSQL. Informatyka ta systemy upravlinnia. 2014. № 3. S. 3–11.

4. Krashivskyi A. I. Rozrobka veb-systemy z vykorystanniam NODE.JS ta MONGODB na prykladi systemy avtomatyzatsii HR-protsesiv. Kompiuterni informatsiini tekhnolohii. Ternopil : ZUNU, 2020. S. 23–24.

5. Spasitielieva S. O. Kompleksnyi zakhyst heterohennykh korporatyvnykh skhovyshch danykh. Suchasnyi zakhyst informatsii: naukovo-tekhnichnyi zhurnal. 2017. № 1(29). S. 58–65.

6. Tsviashchenko O. V. Analiz adekvatnosti modeli uzghodzhennia replik v bazakh danykh NosQL. Informatsiini tekhnolohii. 2015. T. 21. № 11. S. 840–848.

7. Joshi S. Balanced Load in Distributed System with NoSQL Middleware. International Journal of Emerging Technologies and Innovative Research. 2019. № 6(5), pp. 133–137.

8. Gyorodi C., Gyorody R., Andrada I., Livia B. A Comparative Study Between the Capabilities of MySQl Vs. MongoDB as a Back-End for an Online Platform. IJACSA. 2016. №1. S. 73–38.

9. Gorbenko A., Romanovsky A. Interplaying Cassandra NoSQL consistency and performance: A benchmarking approach. Communications in Computer and Information Science, Vol. 1279 / Editors: S. Bernardi, et al. Springer Nature. Berlin, 2020. pp. 168–184.

10. Karniavoura F. and Magoutis K. A measurement-based approach to performance prediction in NoSQL systems. 25th IEEE Int. Symposium on the Modeling, Analysis, and Simulation of Computer and Telecom. Systems, Banff, Canada, 2017. pp. 255–262.

11. Klein J., Gorton I., Ernst N., Donohoe P. Pham Performance Evaluation of NoSQL Databases: A Case Study. Proceedings of the 1st ACM/SPEC Int. Workshop on Performance Analysis of Big Data Systems, Austin, USA, 2015. pp. 5–10.

12. Kumar S. Comparison of NoSQL Database and Traditional Database-An emphatic analysis. JOIV: International Journal on Informatics Visualization. 2018. № 2. R. 51–55.

13. Nayak A., Poriya A., Poojary D. Type of NOSQL Databases and its Comparison with Relational Databases. International Journal of Applied Information Systems. № 5 (4). 2013. S. 16–19.

14. NoSQL Database Classification: New Era of Databases for Big Data. International Journal of Knowledge-Based Organizations (IJKBO). 2019. № 9. R. 50–65.

15. Rao T. R. The big data system, components, tools, and technologies: a survey. Knowledge and Information Systems. 2018. R. 71–81.

16. Banothu N., Bhukya S. and Sharma K. (2016). Big-data: Acid versus base for database transactions. 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). URL: https://ieeexplore.ieee.org/document/7755401.