МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ВАЩУК Михайло Федорович

Метод та програмне забезпечення для управління реплікаціями в нереляційних СУБД / Method and software for replication management in non-relational DBMS

спеціальність: 121 - Інженерія програмного забезпечення
освітньо-професійна програма - Інженерія програмного забезпечення

Кваліфікаційна робота

Виконав студент групи ІПЗм-21
М. Ф. Ващук

_____

Науковий керівник:
к.т.н., доцент А. М. Мельник

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2022

ЗМІСТ

ВСТУП 4
РОЗДІЛ 1 ОГЛЯД ВІДОМИХ МЕТОДІВ УПРАВЛІННЯ РЕПЛІКАЦІЄЮ В БАЗАХ ДАНИХ NOSQL 7
1. Реляційні бази даних і зберігання NoSQL 7
1.2. Переваги та недоліки баз даних NoSQL 8
1.3. Класифікація баз даних NoSQL 12
1.4. Паралельні реалізації в базах даних NoSQL 13
1.5. Постановка проблеми дослідження 23
Висновки до першого розділу 25
РОЗДІЛ 2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ УПРАВЛІННЯ РЕПЛІКАЦІЯМИ В СИСТЕМАХ NOSQL 26
2.1. Приклади процесів підготовки дублікатів зображень під час оновлення кожного запису бази даних 26
2.2. Аналіз прийнятних реплік у кінцевому документі 31
2.3. Математична модель збоїв і пошуку документів у відновленні баз даних NoSQL 35
2.4. Дослідження адекватності моделі керування реплікацією на базах даних типу NoSQL 39
Висновок до другого розділу 43
РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КЕРУВАННЯ РЕПЛІКАЦІЄЮ В БАЗАХ ДАНИХ NOSQL 45
3.1. Архітектура системи та продуктивність 45
3.2. Використання програмного забезпечення на етапі проектування системи 50
3.3. Причина вибору технології NoSQL 54
3.4. Аналіз показників ефективності контролю реплікації та відмови в доступі до запису бази даних 55

Висновки до третього розділу 60
ВИСНОВКИ 61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 62

ВСТУП

Актуальність. В останні кілька років у сфері роботи даних домінує реляційна СУБД. У таких системах дані зберігаються на у вигляді таблиць також передбачається існування структури бази даних. але в також були розроблені великі системи (великі дані), що використовують реляційні СУБД , розробники почали відчувати серйозні проблеми:

1) стало складніше процес збору даних, оскільки він вимагає читання записів із великої кількості пов’язаних таблиць (виникає проблема втрати відповідності);

2) встати конфлікт між необхідністю зберігання великих обсягів документів неструктурованість інформації та необхідність її певним чином упорядкувати шляхом розробки системи баз даних ;

3) для підготовки великих обсягів даних необхідно придбати дорогі спеціалізовані апаратно - програмні комплекси паралельних систем баз даних (Teradata, Sun Oracle). Web Engine тощо) ;

4) виникає при великій кількості вузлів проблема забезпечення того, щоб система була здатна до необхідної відмовостійкості.

Як спроба вирішити накопичені проблеми в реляційних базах даних виникли інші методи зберігання та обробки отриманих даних. під назвою «бази даних NoSQL». Двоє є піонерами в цій роботі компаній: Google і Amazon. У базі даних NoSQL для забезпечення чистоти толерантність до помилок використовує кілька ітерацій (кодування). але є проблема з базами даних NoSQL: у цих системах немає вони підтримують природу торгівлі та цензури, так буває проблема координації реплікації .

Основні характеристики синхронізації зображень у системах баз даних NoSQL існує ймовірність того, що застарілий запис буде прочитано під час розповсюдження оновлення на системних вузлах, час очікування початку читання запису на оновлених серверах iz , кількість змін запису в базі даних

NoSQL і час їх обробки, можливість заборони доступу до журналу бази даних і так далі. Ці сценарії слід оцінювати на етапі проектування системи, оскільки е дозволяє уникнути ручного вибору значень параметрів, необхідних для численні типи записів бази даних на етапі налагодження системи та необхідності повна картина перевантаження системи.

Будучи основоположною технологією інформаційних систем Дані NoSQL дуже нові, потрібні обчислювальні моделі щоб оцінити рівні узгодженості, вироблені в репліках, відсутніх або присутніх чого не вистачає

Таким чином, розробка відповідних математичних моделей і програмного забезпечення , які дозволяють оцінювати системи NoSQL на етапі проектування індикатори еквівалентності та вибору необхідних параметрів доступні фактично в день _ .

Цілі та завдання дослідження

Метою проекту є розробка математичних моделей та програмні засоби для оцінки відповідних представлень символів у базах даних NoSQL знаходиться на стадії проектування інформаційних систем.

Для досягнення поставленої мети:

1) аналіз та розробка імітаційних моделей процесів паралелізму в базах даних NoSQL;

2) аналіз показників угоди про розробку обладнання, його моделі на етапі проектування інформаційних систем;

3) використання розроблених інструментів для аналізу індикації прийняття та вибір параметрів реплікації на етапі дизайну інформаційних технологій.

Метою дослідження є порівняння продуктивності в розподілених базах даних NoSQL.

Методи дослідження – методології та програмне забезпечення для керування реплікацією в розподілених базах даних NoSQL.

Наукова новизна. Запропоновано модель координації реплікацій, що дозволяє оцінювати параметри випадкового часу очікування початку читання запису в оновленому кластері серверів .

*Практична цінність.*Для практичного застосування теоретичних результатів розроблено програмне забезпечення для аналізу процесу зіставлення в базах даних NoSQL, що дозволяє оцінювати показники узгодженості, збереження змін документів, збоїв та їх відновлення шляхом отримання запису бази даних на етапі налаштування інформаційної системи.

РОЗДІЛ 1
ОГЛЯД ВІДОМИХ МЕТОДІВ УПРАВЛІННЯ РЕПЛІКАЦІЄЮ В БАЗАХ ДАНИХ NOSQL

1. Реляційні бази даних і зберігання NoSQL

Реляційна модель була запропонована Коддом у 1970 році в його статті [2]. Ця модель добре підходить для клієнт-серверного програмування операційних систем і стала стандартною технологією зберігання структурована інформація, включаючи онлайн-бази даних.

Можна пояснити наступні переваги реляційних баз даних, які відображені в їх існуванні:

1) теоретичні основи реляційних баз даних (РБД);

2) потужна і відносно проста в освоєнні описова мова і маніпулювання даними (мова SQL), що стало стандартом;

3) оптимізатор запитів, який виконує декомпозицію запитів на підзадачі, гарантуючи, що ці завдання виконуються одночасно багатопроцесорні або багатомашинні комплекси;

4) засоби забезпечення належного функціонування системи: керування ACID транзакції [3] (атомарні , статичні, ізольовані, довговічні) і оплата записів;

5) численні загальносистемні ресурси (СУБД, механізми баз даних, . інструменти CASE тощо).

Однак із зростанням складності вирішуваних завдань зростає і обробка даних проблеми реляційних баз даних стали ставати більш очевидними:

1. Комунікаційні бази даних характеризуються проблемою відсутньої кореспонденції [4] . Вона проявляється в тому, що реляційні бази даних не дозволяють зберігати в собі агрегати чітко Наприклад, щоб відобразити всю

інформацію про клієнта та всі замовлення (зведені) , програміст повинен агрегувати дані з кількох пов’язаних таблиць : клієнти, замовлення, частини року замовлення тощо. Оскільки кількість столів зросла в геометричній прогресії, виробник зазвичай простий забути призначення тієї чи іншої таблиці, стає важко об’єднати таблиці , коли запит, що означає, що агрегат дуже важко виконати.

Поява набору бібліотек для об’єкта – діаграми зв’язків, напр такі як Hibernate та iBATIS, які не вирішили проблему [4] . Зі збільшенням складності завдань і обсягу даних, що обробляються, недоліки пов’язаних баз даних ставали все більш очевидними:

1. Для реляційних баз даних характерна проблема відсутньої відповідності [4]. Це проявляється в тому, що реляційні бази даних не дозволяють зберігати агрегати в складній формі. Наприклад, щоб відобразити всю інформацію про клієнта та всі його замовлення (одиниці) , програмісту потрібно зібрати дані з кількох пов’язаних таблиць : Клієнти, замовлення, одиниці замовлення тощо. Коли кількість таблиць значно збільшується, розробник часто просто забуває про призначення тієї чи іншої таблиці, ускладнюється об’єднання таблиць , коли у b , виконання запиту, тобто створення агрегату, стає набагато складнішим.

Існує багато бібліотек для відображення об’єкта – відношення, напр наприклад Hibernate і BATIS, що не усунуло проблему [4].

1.2. Переваги та недоліки баз даних NoSQL

Нещодавно термін NoSQL (що означає «не лише SQL») використовувався для опису великого класу баз даних, які не мають функцій традиційних реляційних баз даних і часто не мають мови запитів, відмінної від SQL (Structured Query Language). Це питання знову підняли великі компанії,

такі як Google і Amazon, які використовують технологію NoSQL для запуску власних баз даних з метою зберігання та обробки великих обсягів даних [11, 12]. Інші компанії наслідували цей приклад: Facebook, Twitter, eBay, сайт Digg тощо.

Можна виділити наступні переваги баз даних NoSQL:

1. Наразі обсяги даних зростають із величезною швидкістю, а бази даних NoSQL дуже добре підтримують горизонтальну масштабованість (тисячі вузлів) – обмін даними та дублювання (реплікацію) на кількох серверах. Це дозволяє підтримувати велику кількість простих операцій читання/запису одночасно та забезпечує значне зменшення системних помилок.

2. Немає необхідності в дизайні бази даних, як це прийнято в контактних базах даних. Тут система більш гнучка. Дані зберігаються як записи <ключ, значення>. Різні записи можуть мати різні параметри, тобто параметри поля в полі «значення» можуть відрізнятися.

3. У полі «ціна» можна зберігати багато інформації, наприклад, всю інформацію про клієнта та його замовлення (рис. 1.1). Тобто немає необхідності читати дані з різних таблиць. Агрегат може мати складну структуру (наприклад, документ Json).

4. Майже всі бази даних NoSQL є безкоштовними та працюють на різних операційних системах. Процес їх групування відносно простий (порівняно з реалізацією паралельної бази даних контактів). Кластер можна створити на великій кількості малопотужних вузлів і розгорнути в хмарі.

5. Багато баз даних NoSQL оснащені підтримкою паралельних обчислень (технологія MapReduce [13]).

Багато фреймворків NoSQL (CouchDB, HBase, Riak тощо [14]) підтримують стандарт REST для маніпулювання репрезентативним станом, який є стилем побудови архітектури розподіленої програми [15]. Отримання або редагування документів здійснюється за допомогою HTTP-подібного запиту (рис. 1.2).

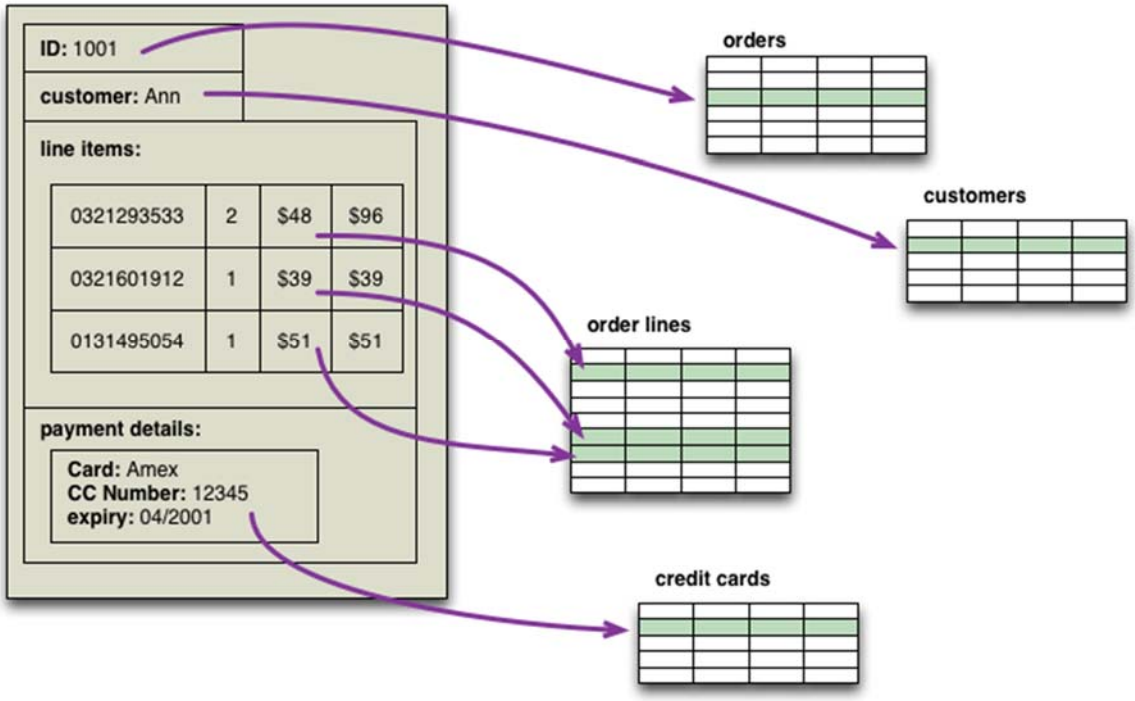

Рис. 1.1. Підготовка документа в базу даних

Стандарт REST був описаний у 2000 році одним із програмістів HTTP, Рой Філдінг [16]. Дані в REST повинні передаватися як невелика кількість стандартних форматів (наприклад, HTML, XML, JSON).

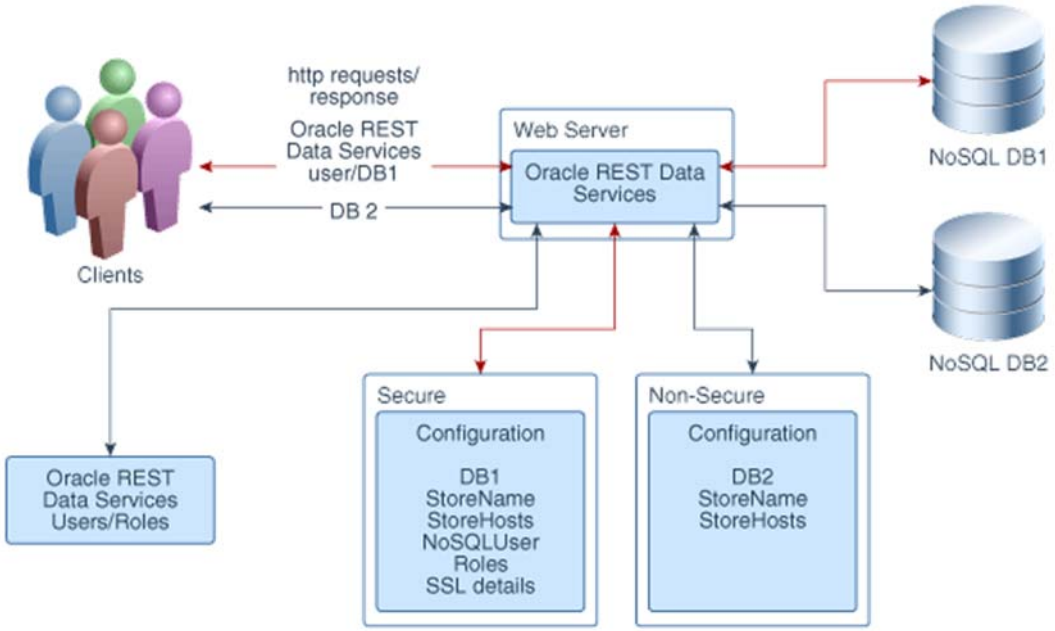

Рис. 1. 2 . Доступ до бази даних NoSQL за допомогою фреймворку REST

Антиподом REST є метод, заснований на виклику віддалених процедур (Віддалений виклик процедури - RPC). Метод RPC архітектури «Сервер додатків» [17], дозволяє використовувати невеликий обсяг мережеві ресурси з численними каналами та надійним розкладом. на у підході REST кількість методів і складність програми сильно обмежені, тому кількість окремих ресурсів, що використовуються, може бути високою [14].

Щоб досягти високої горизонтальної масштабованості та підвищити продуктивність у NoSQL, необхідно послабити обмеження ACID [3]. Це призвело до ряду істотних недоліків:

1. NoSQL не можна використовувати в додатках, де необхідні транзакції ACID: у банківських та інших фінансових системах [18].

2. Бази даних NoSQL забезпечують узгодженість копій записів (реплік) на різних серверах, але існують певні затримки, пов’язані з шляхом поширення оновлень на кілька зображень (синхронізація в кінці, асинхронна реплікація). Тому можна читати застарілі дані. Якщо є сильна синхронізація запису/читання, якщо достатньо параметрів. Однак це призводить до збільшення часу, необхідного для операцій запису/читання.

3. У NoSQL доступ до даних повинен здійснюватися за допомогою мови програмування (на відміну від SQL, яка не є мовою програмування). При цьому збільшується складність програми, щоб об’єднати кілька таблиць у SQL достатньо кодувати одного працівника. Щоб реалізувати цей запит у NoSQL програміст повинен представити його в багатозадачному (завдання) форматі, для напишіть процедури Map і Reduce для кожної та виконайте їх локально MapReduce.

Відмова підтримувати маркетинг ACID має великий вплив на. Впровадження баз даних NoSQL. Однак активи ACID не були оприлюднені повний _. Наприклад, в СУБД Riak [19] можна вказати параметри до і після оновлення даних - це деяка мотиваційна еквівалентність [20]. Помічено, що атомарний формат увімкнено завдяки групуванню всіх динамічних даних в одному записі БД. Звичайно, якщо клієнту потрібно маніпулювати кількома

взаємопов’язаними агрегатами , атомарності не можна довіряти . Відмова NoSQL блокувати нові записи бази даних призвела до відмови від вимоги ізоляції, що підтримується транзакцією ACID.

Отже, і реляційні бази даних (RDB), і бази даних NoSQL мають переваги та недоліки. Кожен тип СУБД (RBD, NoSQL і т.д.) має свою нішу, і вони ще довго будуть співіснувати.

1.3. Класифікація баз даних NoSQL

У NoSQL дані зберігаються як записи <ключ, значення>. Ці основи значною мірою базуються на групі [4], тобто група виступає як «цінність». Існує чотири типи баз даних NoSQL [4, 22, 23]: «ключ-значення», документ, «стовпець», граф.

У базі даних ключ-значення агрегат непроникний для NoSQL. Зчитування запису здійснюється за допомогою ключа. Тільки прикладна програма керує структурою блоку. Пошук за індексом можливий для атрибутів, що зберігаються в спеціальному заголовку запису (Riak). Прикладами високоцінних баз даних є Riak, Redis, Amazon Dynamo.

В інших базах даних агрегат є прозорим NoSQL, що означає, що база даних бачить структуру агрегату. Запис читається на основі накопичених значень полів. Індексний пошук можливий для атрибутів, що зберігаються в агрегаті.

У базі даних записів кожен агрегат має унікальний ідентифікатор, який використовується в додатку як ключ. Агрегатом можуть бути дані, структура яких визначається конкретною базою даних NoSQL. Наприклад, об’єкт зберігається в стеку бази даних MongoDB [14] . JSON (JavaScript Object Encoding) / BSON (JavaScript Object Binary) [25], . які можуть бути структурами з довільним розміщенням вкладеності.

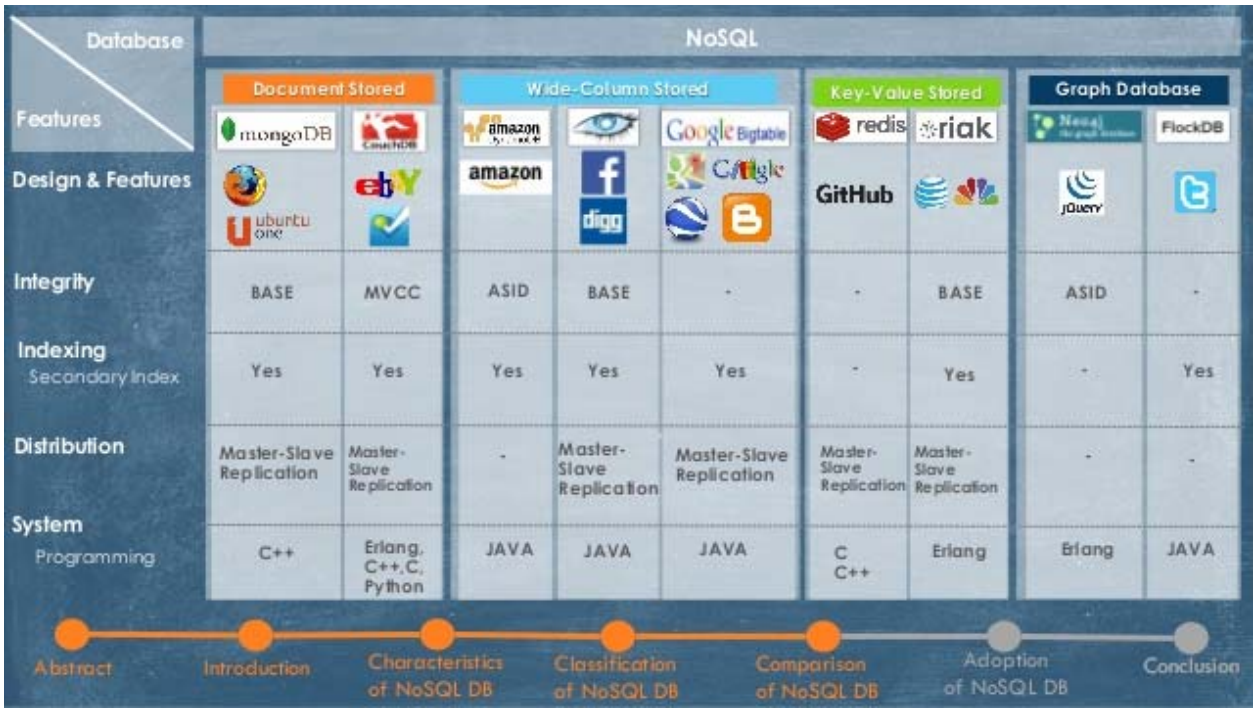

Рис. 1. 2 . Класифікація баз даних NoSQL

Прикладами баз даних документів є MongoBD, CouchBD. Бази даних на основі стовпців зосереджені на зберіганні даних у стовпцях. Набори стовпців у різних записах можуть бути різними. Прикладами цього типу бази даних є BigTable, HBase, Cassandra.

У графовій базі даних дані представлені у вигляді графа: у вигляді вершин і зв’язків між ними. Кожен вузол і зв'язок відповідають даним запису бази даних, набір яких зберігає властивості відповідного об'єкта рядок. Програма зчитує властивості вихідного вузла та підключення та виконує переходячи до наступного розділу. Прикладом бази даних є Neo4j.

1.4. Паралельні реалізації в базах даних NoSQL

Хоча бази даних NoSQL різноманітні, на рівні розгортання вони виконують деякі загальні завдання, пов’язані з укладенням контрактів на репліки:

- завантаження шаблонів документів бази даних у кластер та забезпечення узгодженості при редагуванні запису ;

- копійні версії (об'єднання кількох версій документів в один документ під час зберігання ревізій документа);

- ремонт (відновлення) образів після невдалого видалення вузла.

Бази даних NoSQL в основному використовують два методи відображення та реплікації: головний-підлеглий і кільцевий.

Перший метод полягає в зберіганні даних на головному вузлі та їх реплікації на підлеглі вузли. Усі зміни вносяться на головному вузлі та зберігаються в пам’яті цього вузла . Підлеглі вузли періодично опитують головний node, прочитати зібрані зміни та зберегти їх у своїй пам’яті .

Прикладами таких баз даних є MongoBD, HBase, Neo4j та інші. Розміщуючи дані «на кільці», потрібно знати, скільки сегментів (v-вузлів) воно міститиме. Ці компоненти розподілені між серверами (за кільцем). На рисунку 1.3 показаний приклад [14]. Тут визначено 64 розділи, які розміщено зовні на трьох фізичних серверах A, B, C. База даних виділить 21 або 22 розділи для кожного сервера (64/3).

Коли запис вставляється в базу даних, обчислюється хеш ключа, що вказує на номер сегмента (v-node), який вказує на сервер, де буде зберігатися цей запис. Додаткові копії запису (пронумеровані N-1) зберігаються в наступних станціях (N-1), розташованих за годинниковою стрілкою відносно першої станції. Прикладами веб-сайтів із розподіленими кільцями є Riak, Amazon Dynamo та інші.

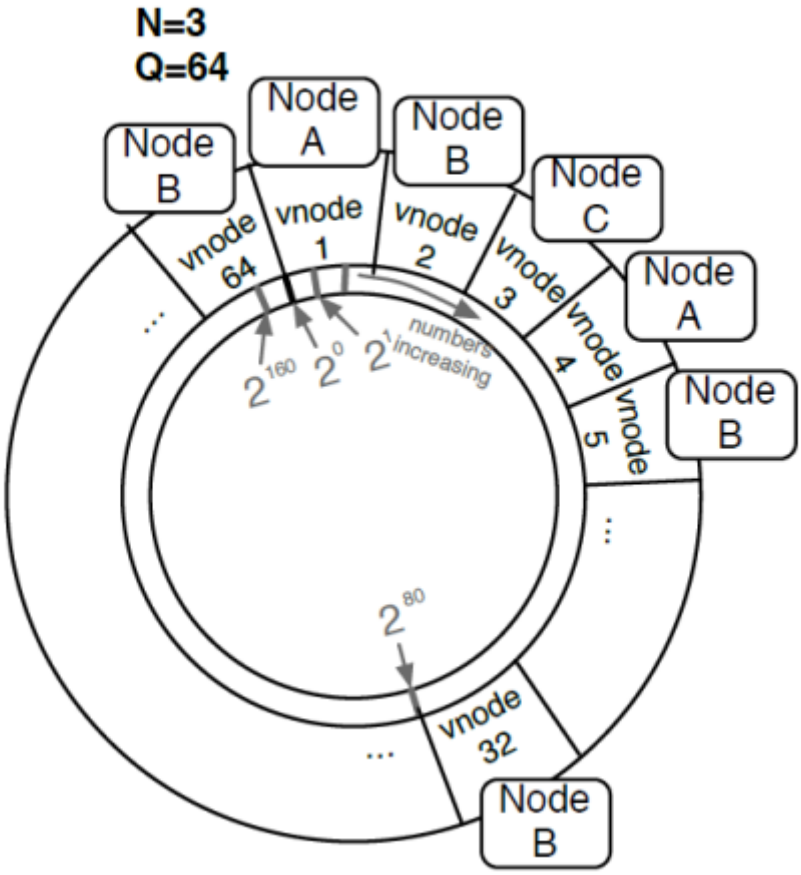

Рис. 1. 3 . Кільце з 64 v-вузлами та трьома серверами

Використання віртуальних вузлів (v-nodes) має такі переваги [26]:

- коли вузол стає недоступним (через збої або планове обслуговування), навантаження розподіляється порівну між вузлами;

- коли вузол знову стає доступним або новий вузол додається до системи, цей вузол несе майже однакове навантаження від кожного з інших доступних вузлів;

- кількість віртуальних вузлів, за які відповідає реальний вузол, можна визначити на основі потужності сервера, що дозволяє збалансувати навантаження в неоднорідній інфраструктурі.

Ітерації використовуються для підвищення продуктивності та стійкості до помилок. Коли запис оновлюється, подальші зміни вносяться не відразу, а в межах вікна конфлікту — часу, який мине від моменту оновлення запису на

кожному вузлі до завершення оновлення копій запису на всіх інших вузлах. Виділяють наступні види зручностей [17]:

- сувора узгодженість - будь-який запит на читання завжди повертатиме останній запис;
- слабка узгодженість - система не гарантує, що запит на читання завжди повертатиме останній запис;
- можлива синхронізація — це особливий вид слабкої синхронізації: репозиторій гарантує, що за відсутності нового запису в кінцевому підсумку запит на читання поверне останній новий запис.

Невідповідності тісно пов’язані зі стійкістю системи , а також із… стійкістю до втрати зв’язку . Цю взаємодію пояснив Ерік Брюер у концепції CAP [18] . Теорія стверджує, що можна створити розподілену систему, яка є 1) узгодженою (Coherent), 2) доступною (Available) і 3) стійкою до втрати зв’язку (Partition Tolerant), але три властивості, лише дві з яких можуть бути гарантовані. зараз. на можлива збіжність гарантується властивостями 2 і 3.

Крім того, в роботі розглядається точність збігу реконструйованих зображень і. узгодженість повторів у кінцевому транскрипті (KR-ratio). Давайте переглянемо наступні тексти:

N - вузли, на які врешті-решт буде відновлено запис (з деякою затримкою);

W — це кількість вузлів (ітерацій), до яких фактично мають бути записані дані перед тим, як надіслати відповідь користувачеві (або процесу) про успішне завершення операції (якщо $W < N$, то система продовжує повторно передавати дані до *N - W групи)* .решта);

R — кількість вузлів, від яких база даних очікує відповіді на успішне завершення читання журналів.

Відсутність блокувань дозволяє читати та змінювати один і той же запис бази даних на різних вузлах одночасно. Це призводить до конфліктів, які обробляються звичайними або ручними інструментами NoSQL.

Перший і найпростіший спосіб розв’язати конфлікти – це використовувати мітку часу, призначену кожному запису. Коли виникають конфлікти, пріоритет надається останньому запису. Однак, як зазначено в [18], такий підхід важко реалізувати в наборі вузлів.

Другим способом боротьби з конфліктами є підтримка змін записів, що здійснюється за допомогою векторних годинників (Vector Clock - VC) [14, 26]. Вектор годинника складається з пар <користувач, номер версії запису для цього користувача>, які описують порядок оновлення цього запису.

На рисунку 1.4 показано приклад запису вектора годин (в дужках вказано вектор записаних годин). Документ *D* (наприклад, якийсь документ) редагується користувачем A1, A2, A3. Користувач A1 послідовно виконує перші два налаштування. Далі користувачі A2, A3 читають і оновлюють цей запис одночасно (випадковим чином). У базі даних зберігаються дві версії документа: D1 і D2. Коли запис D зчитується, дві версії запису з однаковим ключем (D1 і D2) повертаються користувачеві A1. Він вносить зміни, наприклад, об’єднує правки, внесені користувачем A2, A3. База даних зберігає одну статичну версію запису, що містить вектор годин, включаючи ідентифікатори трьох користувачів.

Переваги тактового вектора: відсутність єдиної точки відмови системи, тому що при використанні часових позначок в записах необхідна точна синхронізація часу з одним вимірюванням.

Недоліки тактового вектора: відсутність можливості автоматичного вирішення конфліктів, а також збільшення довжини тактового вектора при частому оновленні запису. Однак у NoSQL є способи відрізати вектор годинника. Наприклад, в системі Ріак можна встановити частоту зрізу вектора на фазовому рівні, а також максимальний розмір (довжину) тактового вектора [19].

Наступний підхід до обробки конфліктів полягає у створенні нечасового ідентифікатора (можливо, хешу вмісту, GUID, номера тощо), який повертається користувачеві разом із необхідними даними з СУБД.

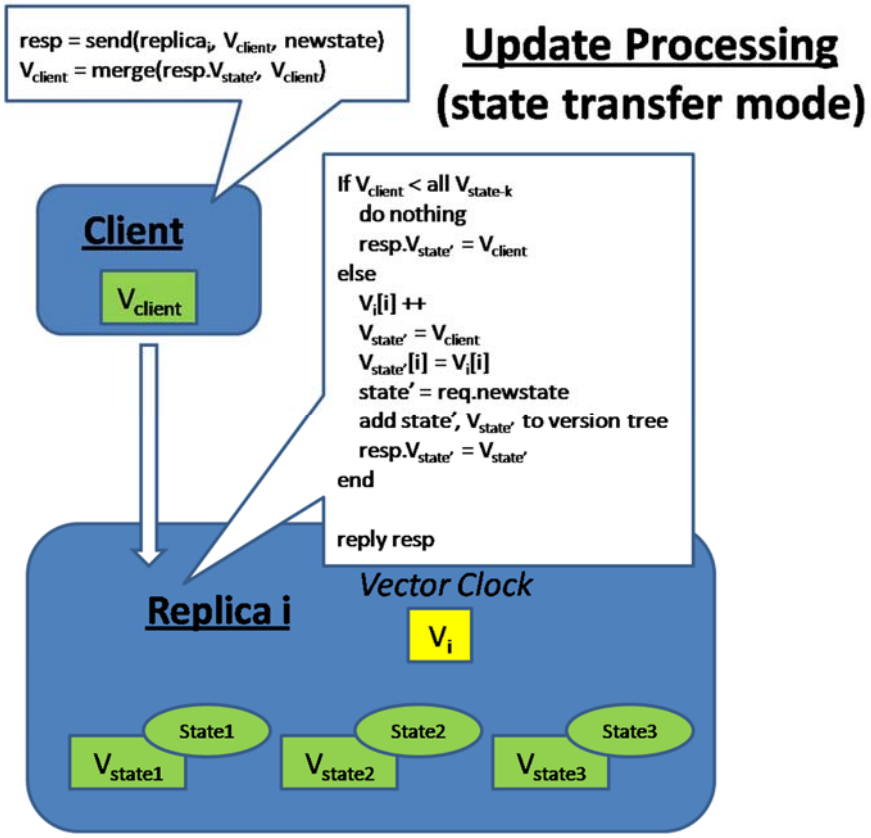

Рис. 1. 4 . Векторна модель захоплення годинника

Після того, як користувач вносить зміни, система порівнює отримане від користувача значення мітки з тим, що зберігається в базі даних. Якщо значення не збігаються, операція відхиляється. Користувач повинен прочитати та оновити запис ще раз. Такий підхід використовується в системі CouchDB [30].

Наведемо приклади аномалій даних, які призводять до появи різних варіантів запису, та способи їх усунення:

- кілька співробітників одночасно редагують копію одного і того ж документа (документа) - наприклад, пропозицію щодо розвитку компанії; вирішення конфлікту – інтеграція (вибір). корективи, внесені керівником групи.

- декілька експертів одночасно оцінюють одну копію паперу - наприклад, сліпа оцінка статті, представленої на конференції; вирішення конфлікту - вибір головою ради директорів.

- кілька користувачів працюють з одним примірником статті одночасно, і один користувач може бачити результати іншого користувача - наприклад, кілька авторів працюють над однією темою; можуть з'являтися різні версії документів, невідповідність усуває керівник авторського колективу.

- користувачі обговорюють одну подію в блозі. У багатофазній обробці запитів із використанням технології MapReduce вихідні дані однієї фази є вхідними даними іншої фази, наприклад, під час виконання операції Join під час сортування перегляду даних або під час виконання складних запитів у пошукових системах (репліки включених даних). у наступному розділі може бути недоречним); цей конфлікт усувається NoSQL (конфлікт на основі ймовірної причини) .

Програма створює ряд пов’язаних записів, їх вони зберігаються в репліках не одночасно один з одним послідовності - наприклад, результати індексації документів у пошукових системах; Відображення конфліктів усуває NoSQL (конфлікт через очевидну причину) .

З наведених вище прикладів ясно, що довіра забезпечує послідовність критичні для стабільної та нормальної роботи системи. Синхронізація створює конфлікти, які можна вирішити, наприклад, лідер групи на основі тактового вектора. Знання того, як розрахувати вікно конфлікту, допоможе вам запобігти конфліктам у цій ситуації.

Для багатокомпонентної роботи не повинно бути конфлікту, оскільки вихід одного компонента використовується як вхід іншого компонента. Знання затримки відповіді системи при забезпеченні ідеальної синхронізації дозволяє точно оцінити час, необхідний для всього багатофазного процесу.

Якщо з одним і тим же записом бази даних одночасно працює велика кількість користувачів, кількість версій цього запису може бути великою. Таким чином, виникає проблема розрахунку навантаження на користувача з точки зору того, скільки користувачів одночасно працюють з документом: оцінка кількості версій документів, часу їх обробки користувачем тощо. За

результатами цих досліджень можна сформулювати рекомендації щодо того, скільки користувачів можуть брати участь в обговоренні одночасно.

1.4. Огляд відомих моделей і підходів до покращення продуктивності баз даних NoSQL

Іноді на практиці комбінація параметрів *N, W, R* недостатня для контролю узгодженості даних. Це відбувається, коли якість констант потрібно підвищити на заданому рівні доступності, і навпаки. У [20] проблема спільного використання функцій безпеки та довговічності в розподілених центрах обробки даних розглядається з використанням «болтового» кластера. Цей шар передбачає причинно-наслідкову послідовність. Причинно-наслідковий союз — це союз, заснований на причинно-наслідковому відношенні <відбулося - з> . Таким чином, усі операції запису описують причинно-наслідкову історію. Цей спосіб гарантує точну відповідність.

Причинно -наслідкові зв'язки зазвичай мають дві форми: ймовірнісні та явні причинно-наслідкові зв'язки . У потенційному зв’язку всі записи, які можуть вплинути на інший запис, мають з’явитися до появи цього останнього запису. У статті [1 0]. архітектура шару "bolt-on" показана наступним чином. Увімкнено кожен машина замовника поставляється для встановлення прокладки метадані та локальне сховище. Клієнт вказує цю групу, яка, в свою чергу, вказує на базу даних, узгоджену в кінцевому рахунку. Цей рівень обмежує порушення узгодженості, які може бачити клієнт. Веб-сайт керує розповсюдженням налаштувань журналу: для читання оновлень в інших програмах кожна прокладка надсилає відповідний запит на веб-сайт.

Послідовні записи , які можуть операції читання та додавання можуть виконуватися постійно без будь-яких перерв обмеження безпеки. Цю категорію записів називали «причиною». data cut" $x1 \rightarrow y1 \rightarrow z1$(causal sat).

Попросіть іншого клієнта зберегти запис. $y2 \rightarrow NULL$.Під час пошуку нової версії запису $y2$рівень shim повинен переконатися, що запис існує в сховищі $y1$. Інакше причинно-наслідковий зв'язок $x1 \rightarrow y2$ буде втрачено. У статті розглядаються алгоритми для роботи з прокладками, а також процедури для перевірки цілісності причинно-наслідкових зв’язків. Запропонована архітектура схематично показана на рисунку 1.5.

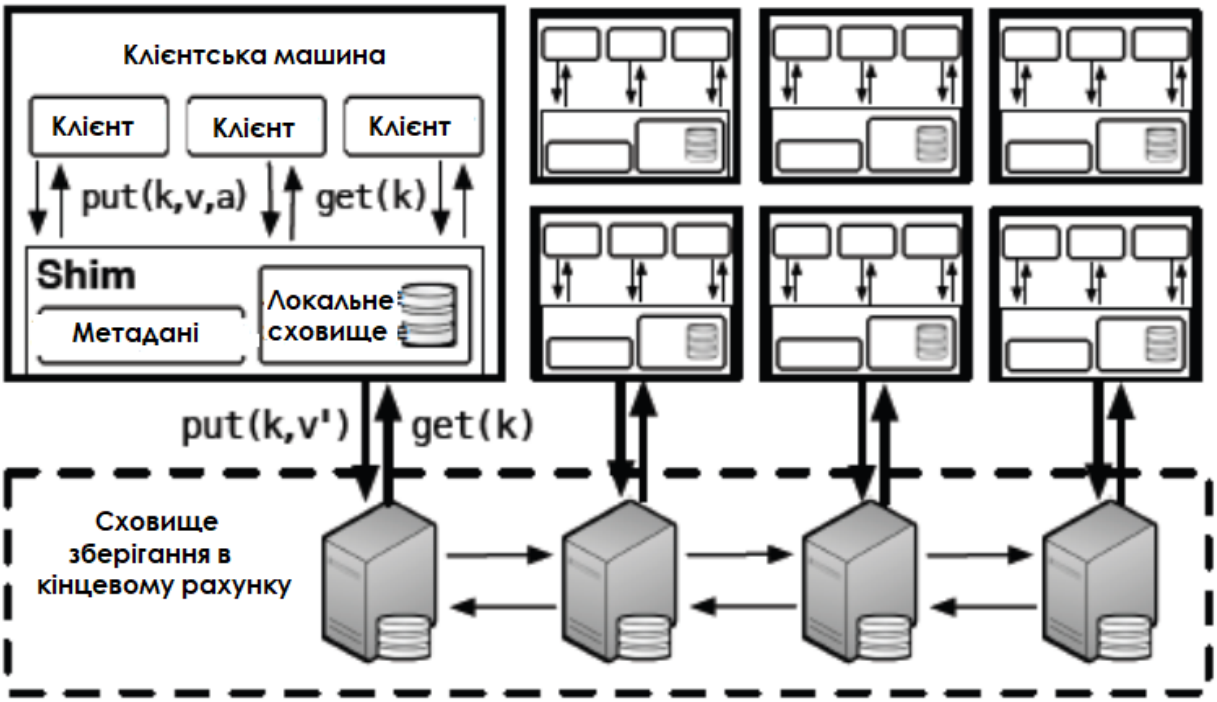

Рис. 1. 5 . Схема блоку зберігання

Перевагою розглянутого підходу є підвищення рівня сумісності до сильної інтеграції за рахунок включення додаткового рівня на клієнтську машину. Недоліком є часткове перенесення операцій з базою даних на клієнтську машину, а отже, збільшення обсягу збережених даних, наявність додаткової затримки для синхронізації локальної та повсюдної бази даних.

Цей підхід можна використовувати для коротких операцій, наприклад, щоб уточнити цілі під час пошуку даних. Розбіжності в кінцевому підсумку не розглядалися.

Стаття [14] пропонує підхід LibRe до інтеграції даних у базах даних NoSQL. Метод залежить від відповідних елементів у кінцевому документі.

Завдання полягає в тому, щоб поєднати величезну кількість доступних даних із високою інтеграцією даних. Наприклад, якщо $N = 3, W = 1, R = 1$, то KR-рівняння виконується, оскільки $W + R \leq N$. Запропонований у статті метод дозволяє посилити невідповідності. LibRe означає Library For Replication.

Використовується наступний метод. Модуль керування LibRe розташований поруч із контролером навантаження. Перед початком кожної операції запису/редагування завантажувач налаштовує набір вузлів у LibRe, на яких можна виконувати операцію. Після виконання завдання LibRe тимчасово зберігає інформацію про вузол, який його виконав. Для кожної операції читання LibRe повертає набір вузлів, які зберігають набори даних, що стосуються поточного запиту. На рисунку 1.6 показано, де LibRe знаходиться в системі.

Цей підхід має перевагу підвищення рівня узгодженості до суворої узгодженості завдяки введенню додаткового реєстру, який тимчасово зберігає дані, на яких відбулося оновлення певного запису. Недоліком є невелика затримка, необхідна для визначення потрібного профілю. Реєстр може стати «вузьким місцем».

Стаття [12] описує підхід PAXOS для забезпечення узгодженості в розподілених базах даних. Метод заснований на консенсусній реконструкції. Алгоритм зіставлення реплік у класичній версії (Paxos basic) виглядає наступним чином:

1. Як планувальник вибрано певний екземпляр (сервер).

2. Планувальник вибирає документ і надсилає його до всіх його параметрів, щоб переконатися, що одержувач може прийняти або відхилити документ.

3. Після того, як більшість реплікованих зображень отримають новий запис, він підтверджується, і всім зображенням надсилається команда підтвердження операції (здійснення).

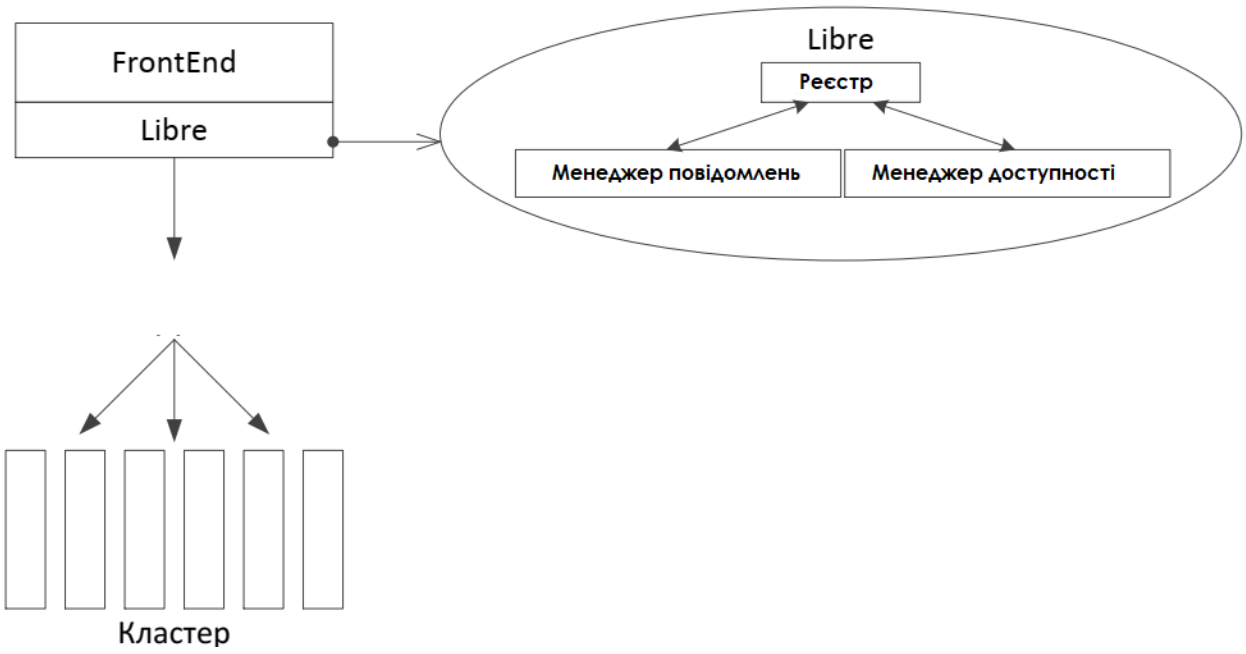

Рис. 1. 6 . libre в системі

При необхідності слова 1-3 можна повторити (через збій мережі). Роботодавець також може відмовити. Але paxos не вимагає одного органайзера, в будь-який момент органайзером може стати інший його екземпляр.

1.5. Постановка проблеми дослідження

Для кожної змінної асиметрії даних є одна прийнятна час розбіжності даних. Тому на стадії проектування системи необхідно передбачити, коли дані будуть невірними для кожного варіанту завдання, які необхідно вирішити, а також можливість зчитування суперечливих даних з часом дифузія змін. Водночас це слід розглядати як безпеку покращення якості відповідності, що призводить до збільшення часу реакції (через необхідність узгодити багато його рисунків і прочитати текст із багатьох відтворених розділів) і кількість залучених ресурсів.

Як уже було сказано, з великою кількістю людей, які працюють одночасно користувачів з одним записом бази даних, кількість версій цього запису може є великим. Таким чином, існує проблема оцінити навантаження на користувача з точки зору кількості користувачів, які працюють з документом одночасно: оцінка кількості змін, записаних о , час їх обробки користувачем та інші. За результатами цих тестів можна давати рекомендації щодо того, скільки користувачів можуть брати участь в обговоренні одночасно.

У великих кластерах вузли можуть виходити з ладу кілька разів, і їх можна відновити команди не завжди можуть справлятися із завданням. Отже, існує ймовірність того, що він буде недоступний із розподіленого сховища прочитати сценарій . Звідси теоретичний аналіз обставин відмовостійкість у великих групах за наявності багатьох копія кожного запису в базі даних є важливим завданням.

майстер-аналізі вирішуються наступні завдання :

1. Розробка паралельних аналітичних моделей у залежності з параметрів N, W, R:

- . $W + R \leq N$- послідовність в кінцевому розрахунку. Вони зробили це біда оцінки ймовірності запитів на читанняR Без зміни записів БД під час нових $N - W$реплікацій цього запису (і тому клієнт отримає застарілий запис). Розглянуто одночасний та асинхронний методи переговорів його приклад

- . $W + R > N$- саме те, що потрібно. Математична задача розв’язана характеристики випадкового часу очікування за вимогою до нього читайте кінець ітерації Wта коли ви читаєте текст , враховуючи це очікування.

і розробка системи зберігання змін документів для картографування одночасна обробка кожного документа кількома користувачами .

3. Розробка аналізу та моделювання моделей відмови та відновити доступ до запису бази даних NoSQL на основі доступності Nйого приклад

4. Розробка програмного забезпечення для аналізу сигналів паралелізм у базах даних NoSQL.

5. Використання розроблених моделей та інструментів записи про на етапі проектування інформаційної системи .

Висновки до першого розділу

1. Обговорюються переваги та недоліки реляційних баз даних і баз даних NoSQL. Було показано, що пропуск обмежень ACID у NoSQL викликає проблему узгодженості репліки.

2. Система ведення змін документа за вектором часу розглядається як один із способів усунення невідповідностей, спричинених відсутністю обмеження на бази даних NoSQL. Показано, що існує проблема з оцінкою навантаження на користувача відповідно до кількості користувачів, які працюють і розмовляють одночасно.

3. Було проведено опитування доступних алгоритмів, які дозволяють оновлювати репліки записів після відновлення вузлів. Показано , що теоретичний аналіз властивостей похибокостійкості у великих групах за наявності кількох зображень кожного запису в базі даних є актуальним завданням .

РОЗДІЛ 2
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ УПРАВЛІННЯ РЕПЛІКАЦІЯМИ В СИСТЕМАХ NOSQL

2.1. Приклади процесів підготовки дублікатів зображень під час оновлення кожного запису бази даних

У розділі 1 було зазначено, що бази даних NoSQL використовують два методи синхронізації:

1. Розбіжності в останній примітці ($W + R \leq N$).

2. Сильна інтеграція ($W + R > N$).

Перший спосіб забезпечує швидкий доступ до даних, але при цьому читає застарілі документи. Цей спосіб використовується для надання доступу до веб-документів таких типів: рекламні пости, пости в блогах, стрічки новин тощо.

Другий підхід гарантує читання остаточного варіанту тексту, але при цьому можливі затримки читання. Цей спосіб використовується для налаштування отримання таких типів записів: залишки товарів і ціноутворення в інтернет-магазині, чат, кеш користувачів, дані про сесії тощо.

На показники якості при координації репліки впливають такі параметри:

N- кількість параметрів документа.

W- кількість ітерацій, які журнал повинен зберігати до повернення відповіді про успішне завершення операції журналу.

R- кількість ітерацій, з яких документ повинен бути прочитаний до повернення відповіді про успішне завершення операції читання.

На цьому етапі вирішуються такі завдання:

1. Для відповідного підходу нарешті створюється модель, яка дозволяє обчислити ймовірність того, що запит на читання отримає Rзаписи БД, які не

оновлюються під час оновлення записів репліки $N - W$, тобто клієнтом буде отримано застарілий запис (синхронно та асинхронні методи розповсюдження оновлення).

2. У методі точного узгодження розробляється модель, яка дозволяє обчислювати випадкові параметри часу очікування, вимагаючи, щоб Wкінець оновлення зображень було прочитано та коли запис прочитано в цій надії на розгляд.

Давайте розглянемо синхронні та асинхронні способи поширення оновлень документів бази даних у її профілі. Синхронний режим означає, що після отримання запиту на оновлення даних планувальник послідовно надсилає оновлення для наступних зображень після завершення оновлення попереднього (рис. 2.1a).

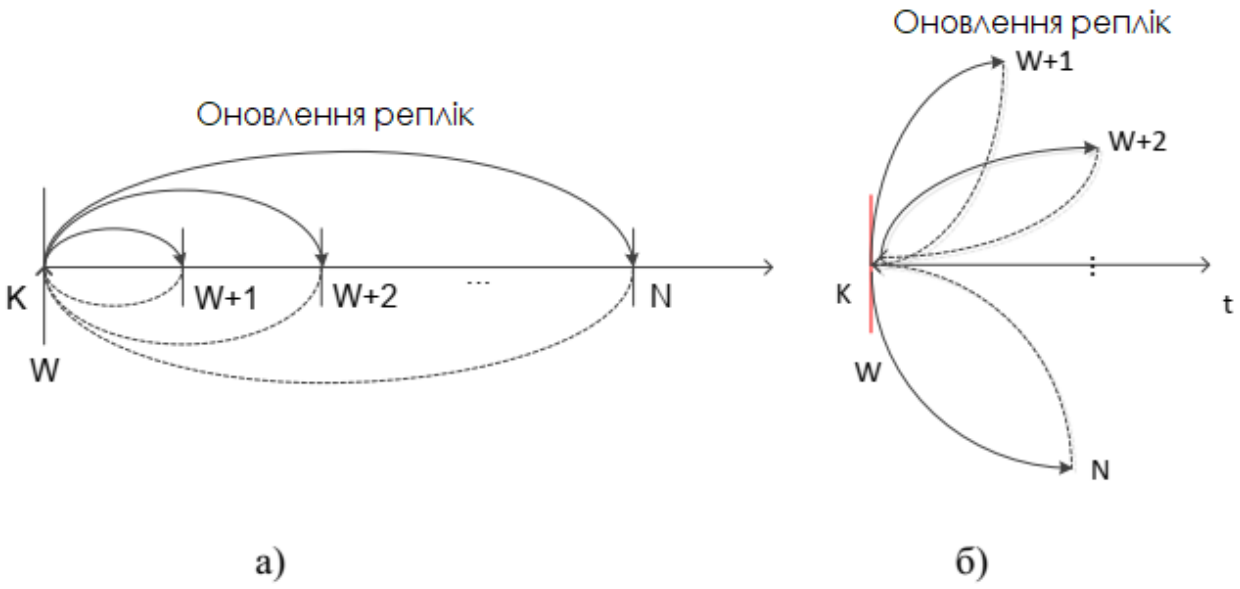

Рис. 2 . 1 . Варіанти поширення переходу: а) синхронний режим; б) асинхронний стан.

В асинхронному режимі планувальник надсилає оновлення свого наступного зображення, не чекаючи завершення попередньої версії (рис. 2.1b). Час для оновлення змін в асинхронному режимі менший, ніж у синхронному, оскільки зміни поширюються одночасно. На рисунку 2.1 оновлені зображення позначені $1,2,\ldots,N$,літерою К, позначеною як органайзер.

1. Синхронний режим. На рисунку 2.2 показано приклад розбіжності даних для випадку одиниці в кінцевій оцінці ($W + R \leq N$)[8, 9]. Потік запитів на читання з одного зображення вважається пуассонівським з параметром $\lambda, \Psi i$ (s)- перетворенням Лапласа-Стілтьєса функції розподілу ймовірностей часу оновлення і-го зображення.

Рис. 2.2. Приклад зіставлення параметрів у кінцевому розрахунку ($W + R \leq N$, стан узгодженості).

На кожному інтервалі часу $(W + i)\lambda$запити на читання від уже оновлених зображень надходять з накопиченою швидкістю, а зі швидкістю $(N - W - i)$ λ- від неоновлених зображень. Завдання полягає в тому, щоб обчислити ймовірність Pтого, що під час випадкового перевідображення відбудеться хоча б один запит, і відповідно прочитати застарілий документ у відображенні R.

Для вирішення поставленої задачі спочатку пропонується отримати похідну від $Q_{W+i+1}(z)$кількості запитів на читання запису, які надходять у $W + i + 1$випадковий час оновлення ( ) цього зображення запису, і Rзапису читання, який з його $N - W - i$неоновлених рисунків пояснює їх для цього $(i = 0 \ldots N - W - 1)$.

$\psi_{W+i+1}(s)$- Перетворення Лапласа-Стілтьєса функції розподілу ймовірності часу оновлення ($W + i + 1$) реконструйованого зображення. Покажіть йому цеg ймовірність того, що запити на читання з неоновленого знімка отримають $R - 1$неоновлені записи з інших знімків.

$$Q_{W+i+1}(z) = \psi_{W+i+1}(s)\big(\lambda \cdot g_i(N-W-i)(1-z)\big) \qquad (2.1)$$

$$g_i = \begin{cases} \frac{C_{N-W-i-1}^{R-1}}{C_{N-1}^{R-1}}, \text{якщо } N-W-i \geq R \\ 0, \text{ інакше} \end{cases} \qquad (2.2)$$

доведено на основі формули (2.2) традиційного визначення ймовірності є відношення рівних чисел комбінацій до загальної кількості шарів. У цьому випадку відповідна кількість кластерів є варіантами, коли після отримання запиту на читання записів із неоновленого знімка записи з $(R-1)$решти неоновлених знімків читаються з неоновленої $(N-W-i-1)$бази даних. Загальна кількість комбінацій — це кількість різниць, у яких $(R-1)$дані з параметрів цього запису можуть бути обчислені з $(N-1)$решти параметрів (оновлених і неоновлених реконструкцій).

З (2.2) отримуємо: $Q_{W+i+1}(0)$- це ймовірність того, що при $(W+i+1)$-му оновленні параметра не буде запитів на R читання записів з $N-W-i$неоновлених параметрів, - це $1-Q_{W+i+1}(0)$ймовірність того, що при -му оновленні параметра. ... th $(W+i+1)$прийде принаймні один запит на читання репліки та прочитає Rзаписи з реплік, які $(N-W-i)$не були оновлені.

Таким чином, ймовірність того, що під час $N-W$оновлення відображення прийде принаймні один запит на читання пари <ключ/значення> і відповідно прочитає R-записи з неоновлених відображень, дорівнює:

$$P = (1-Q_{W+i}(0) + \sum_{i=2}^{N-W}\big(1-Q_{W+i}(0)\big)\prod_{j=1}^{i-1} Q_{W+i}(0) \qquad (2.3)$$

Вираз (2.3) випливає з формули абсолютної ймовірності. Ось що це таке ймовірність того, що клієнт прочитає застарілий документ під час розповсюдження нових документів у своїх $N-W$профілях.

База даних Riak NoSQL підтримує режим синхронного оновлення репліки. Система Hadoop також підтримує цю функцію. Тільки тут нова інформація поширюється послідовно від попередньої ітерації до наступної.

На рисунку 2.3 показано приклад паралельного моделювання для випадку $W + R \leq N$(асинхронний режим) [18, 19]. Вхідний потік запитів на читання з одного зображення також вважається пуассонівським із параметром λ.

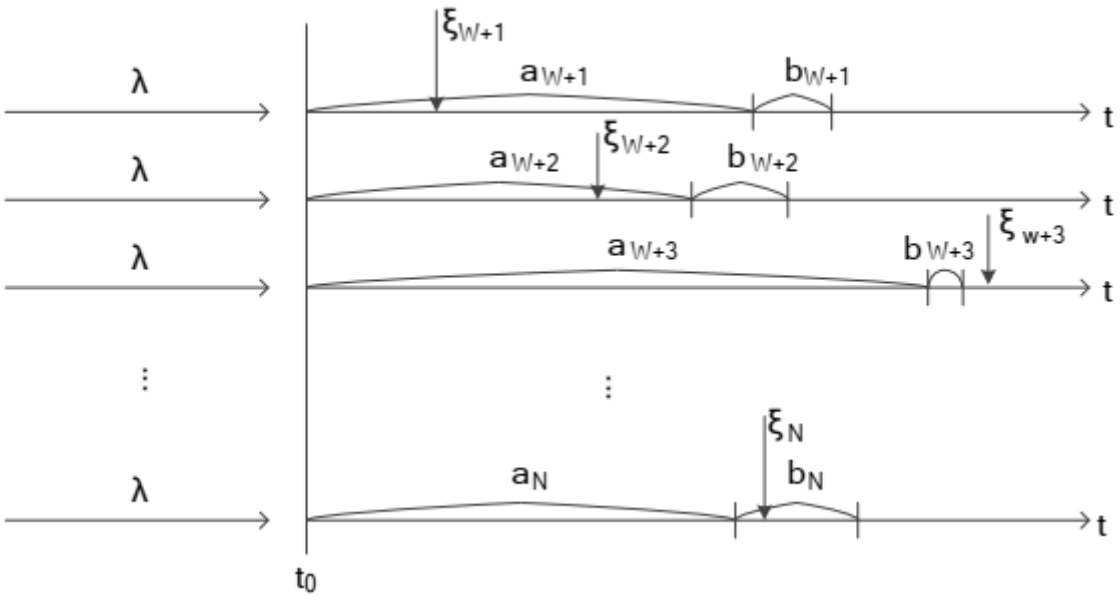

Рис. 2 . 3 . Приклад зіставлення параметрів у кінцевому розрахунку ($W + R \leq N$, асинхронний режим).

На рисунку 2.3 введено такі назви: a_i- випадкова мережа частина часу оновлення *i -oї* репліки b_iє випадковою локальною частина часу поновлення *i -ї* репліки. У кожному інтервалі часу і зі швидкістю λ запити на читання надходять незалежно записів БД з кожного її профілю. Вони t_0є _ ξ_iЗавдання полягає в тому, щоб обчислити ймовірність *H* того, що $N - W$принаймні один запит на читання відбудеться під час оновлення, і прочитати документ за його неоновленими параметрами ($R = 1$).

База даних Amazon Dynamo NoSQL підтримує асинхронний режим оновлення реплік, а також бази даних master-slave і master-master.

2.2. Аналіз прийнятних реплік у кінцевому документі

Результати аналізу ймовірності читання замовником наведені нижче документ застарів у новому розповсюдженні, документ у його$N - W$ репліка, для нормального поширення змін (формула (2. 3)).

Параметри властивості (інтенсивність продуктивності) визначали за допомогою програми тестування дизайну AIDA64 [15] . Статистика були генеруються на основі наступних значень параметрів властивості.

1. Пристрій - Mobile DualCore Intel Core i5-2450M, 2900 МГц. Оскільки виміряне значення кількості робочих циклів обраного процесу, . виробляються за секунду відповідно $\mu_P = 2900 \cdot 10^6$(1/с).

2. Зовнішня пам'ять - Momentus 5400 640423 Seagate <ST9640423AS> 5400 об / хв 16 Мб; швидкість передачі даних на диск μ_{D1P} =130 ·1024 ·1024 (байт/с).

3. Оперативна пам'ять - DDR3-1333 PC3-10667. Труднощі література даних з ОР дорівнює μ_{ns} =9842 ·1024 ·1024 (байт/с).

4. Продуктивність локальної мережі компонента дорівнює 1 Гбіт/с; швидкість передачі даних по шині локальної мережі дорівнює μ_n == 125 ·106 (Байт/с).

5. Пропускна здатність мережі між компонентами 128 Мбіт/с; швидкість передачі даних по мережевій шині, що з'єднує підмережі, дорівнює

μ_{ns} == ·16 106 (байт/с).

рисунку 2.4 показано залежності від ймовірності того, що замовник прочитає застарілий документ при розповсюдженні на ньому нових $N - W$документів репліки, від складності надходять запитів (вимог) і різноманітності значення N (синхронний режим). Загальна кількість віртуальних вузлів становить 20 вузлів вона розділена на два сегменти мережі по 10 вузлів у кожному. Кількість віртуальних вузлів (V -вузли) - 64. Довжина

поля " k " ключа реєстру - 20 байт, довжина " v ". значення поля дорівнює байту 512. $W = R = 1$.

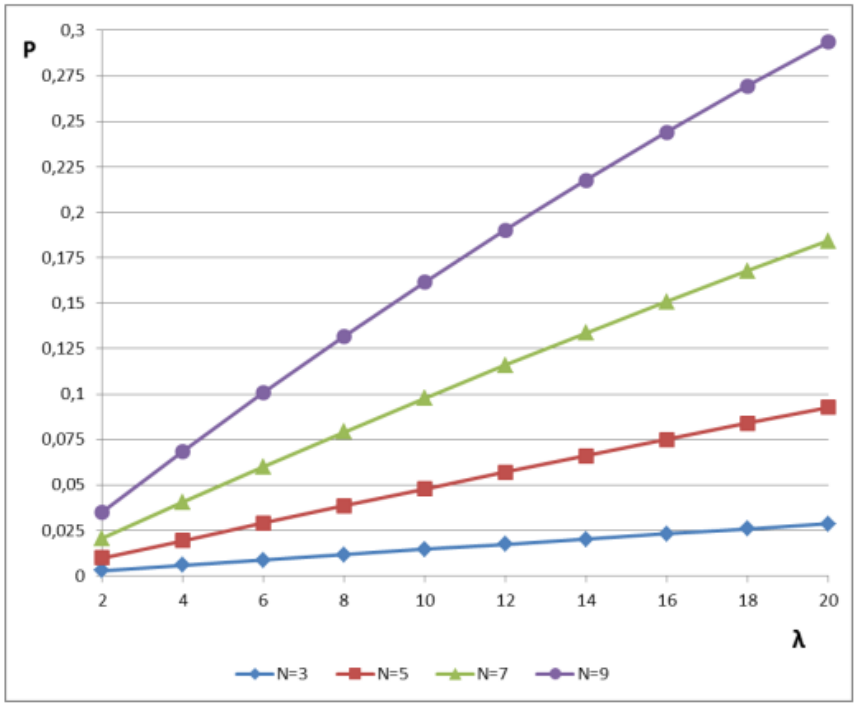

Рис. 2 . 4 . Ймовірності читання клієнтом застарілого запису під час розподілу нових записів на його *N-1* профілях залежно від швидкості запиту читання *λ (1/c)* на *N* (синхронний режим).

При *великих значеннях N* ймовірність може досягати 0,3 складність читання питань. Оцініть ймовірність того, що клієнт прочитає застарілий документ час розповсюдження нових записів на його профілі *СЗ* , *за* асинхронний метод зі змінним розповсюдженням (формула (2. 3)).

Властивості аналогічні описаним раніше. На рисунку 2.5 показані залежності ймовірності, з якою клієнт прочитає застарілий документ час розповсюдження нових записів на своєму *СЗ* профілі, від складність вхідних запитів (запитів) при *різних значеннях N* (Стан оновлення асинхронної репліки).

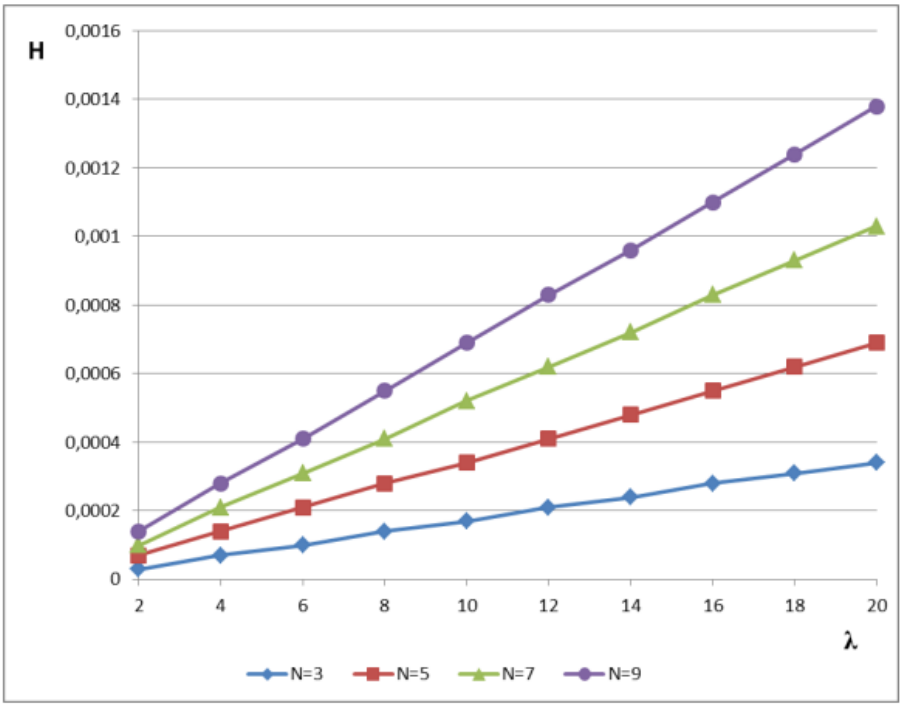

Рис. 2 . 5 . Імовірності того, що клієнт прочитає застарілий запис під час розподілу нових записів за його *NW* координатами, залежно від швидкості запиту на читання λ *(1/c)* протягом *N різних* (асинхронний режим).

У випадку, якщо $N =$ 3навіть якщо ймовірність висока λ, вона не перевищує 0,0004 (імовірність згоди майже вісім чотири: 0,9996) . Ми порівнюємо ймовірність отримання x даних x несумісних з одночасним доступом і некогерентні механізми поширення переходу. Таблиця 2.1 показує значення ймовірності того, що клієнт прочитає застарілий документ з часом розподіл записів оновлень у кожному $N - W$профілі для різних значень інтенсивність вхідних запитів має високі значення N (7 і 9)для синхронний і асинхронний режими. тут $W = R = 1$_.

Нижче наведені результати зразкових тестів з обчислення часу очікування за вимогою підрахунку виконання нових параметрів *W (формула (2.3))*. Розрахунки проводились для наступних значень параметрів властивості:

1. Пропускна здатність локальної мережі сегмента дорівнює 100 Мбіт/с; швидкість передачі даних по шині локальної мережі дорівнює μ_n == 12,5 ·106 (байт/с) .

2. μ_{ns}Продуктивність мережі між її компонентами не враховується (без підмережі). Інші параметри приладу відповідають нормальним значенням. На рисунку 2.6 показано, як залежить статистичне очікування (MS) часу очікування запиту на читання $W = N/2 + 1$відновлених зображень від складності вхідних запитів (запитів) при *різних значеннях N* . Довжина поля « *k* » ключа запису становить 20 байт, довжина поля « *v* » значення – 2048 байт.

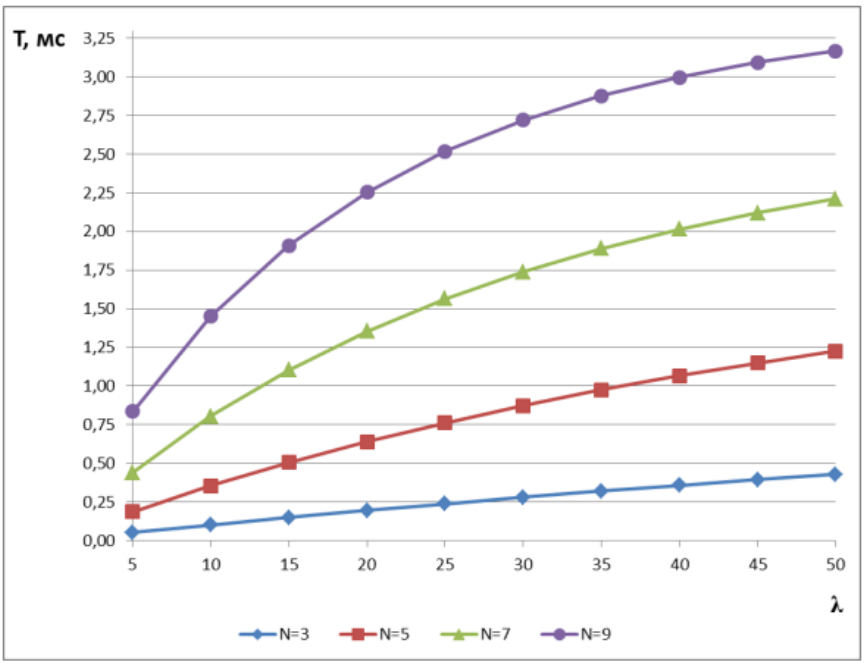

Рис. 2 . 6 . Залежність обчислювального очікування від часу очікування завершення реконструкції параметрів *W* від складності запиту на читання *λ (1/c)* при *різних N* .

З рисунка 2.6 видно, що час очікування для вимоги завершити реконфігурацію параметрів *W* вимірюється при значенні *N = 3* у десятих частках мілісекунди. Однак, якщо N велике і складність вхідних запитів на

читання висока, цей час може досягати 3,2 мс. На рисунку 2.7 показана залежність *параметрів часу читання MC R* при розгляді часу очікування вимог читання в кінці перевідображення W ($W = R = N/2 + 1$) від складності вхідних запитів при *різних значеннях N* .

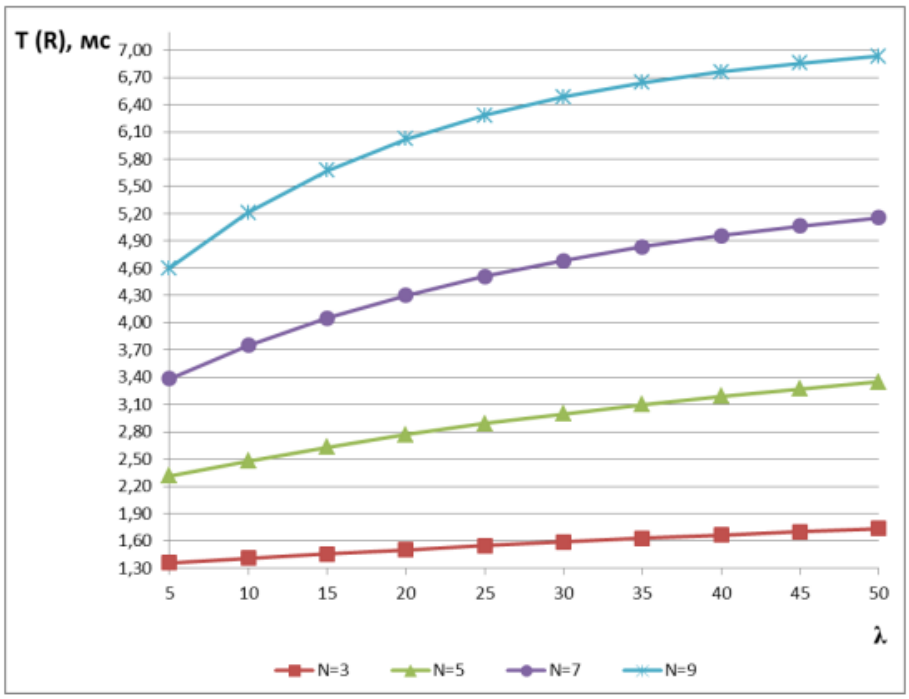

Рис. 2 . 7. 7 . MS читає часову залежність параметрів *R* з урахуванням часу очікування завершення запиту на читання нових параметрів W через складність запиту на читання *λ (1/c) при* різних N .

2.3. Математична модель збоїв і пошуку документів у відновленні баз даних NoSQL

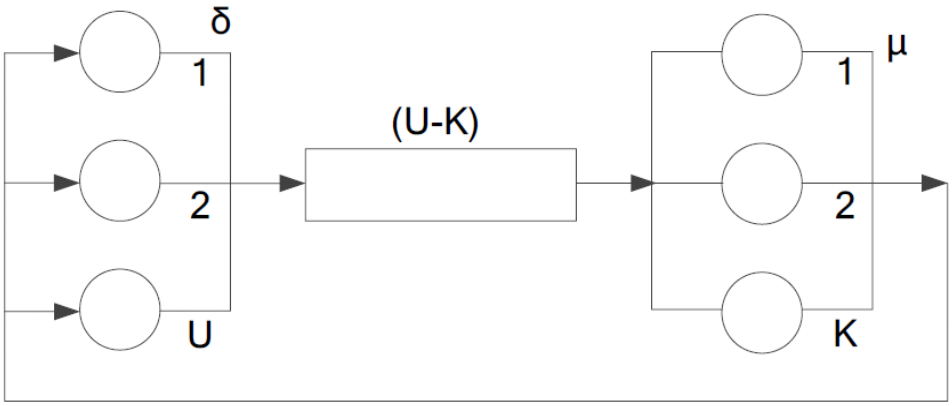

Рис. 2 . 8 . Як обробляти відмову та відновлення вузлів кластера

Розподілимо час обробки для відмови вузла та час його обслуговування (відновлення) за експоненціальним законом параметра $1/\delta$і μ. Тоді цей процес можна описати у формі процесу краудсорсингу $M/M/K/U/U$. Що можливоp_i що iпрограма в системі еквівалентна:

$$P_i = \left\{ p_0 \cdot \left(\frac{\delta}{\mu}\right)^i \cdot C_U^i, \text{якщо} i < \text{K}; \; p_0 \cdot \left(\frac{\delta}{\mu}\right)^i \cdot C_U^i \cdot \frac{i!}{K!} \cdot K^{K-i}, \text{якщо} i \geq K \right\} \qquad (2{,}4)$$

У розглянутій вище аналітичній моделі час відновлення вузла розподілений за експоненціальним законом. Але цього разу може бути велика різниця. У цьому випадку необхідно використовувати модель $M \,/\, G \,/\, K \,/\, U \,/\, U$з бажаною функцією розподілу ймовірностей на момент відновлення. Для такої моделі немає аналітичного рішення. У цьому випадку необхідно використовувати імітаційну модель. Ця модель представлена в [14]. Для оцінки ймовірності p_0розроблено спеціальний підхід, який представлено нижче при розгляді розробки імітаційної моделі мовою GPSS, яка показана на рисунку 2.9.

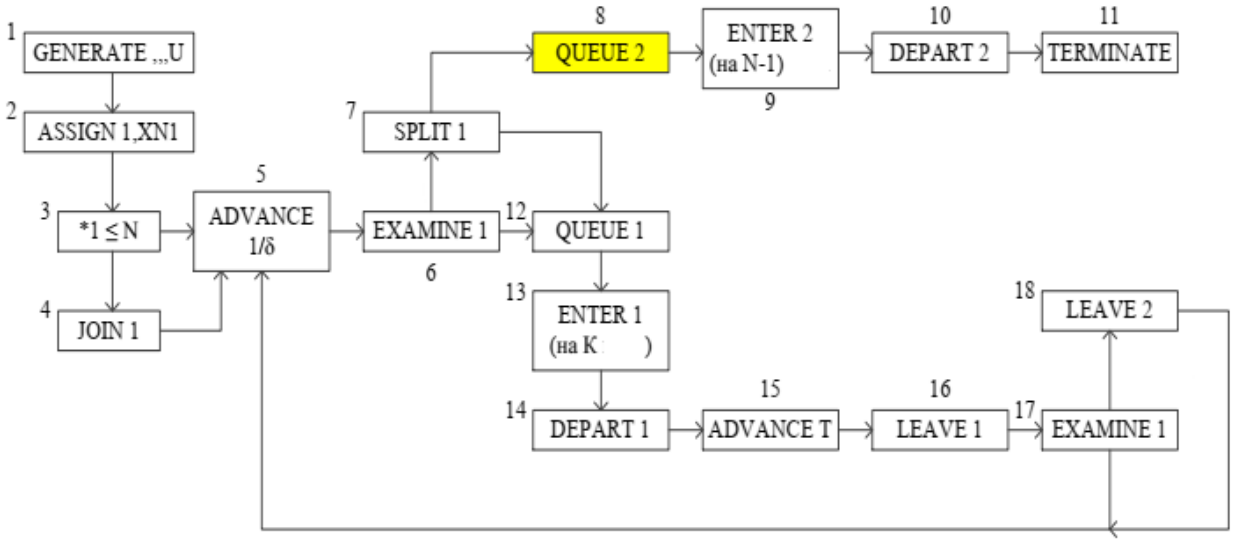

Рис. 2 . 9 . Структура імітаційної моделі доступна мовою GPSS

Вони здійснюють *транзакції U (вузли)* (див. мітку 1); номер транзакції (мітка 2) зберігається в параметрі 1 транзакції; перші *N* трансактів були згруповані в 1 групу (мітки 3,4). Група 1 складається з вузлів, де *зберігаються N зображень кожного запису.* Це перші *N* транзакцій (через симетрію моделі номери вузлів не потрібні).

Крім того, транзакції затримуються на час безпомилкового виконання (експоненціальний розподіл ймовірностей із середнім *1/δ*) (див. Мітку 5). Після збою вузла визначається, чи належить транзакція до групи 1 (мітка 6). Якщо «так» (вузол зі своїм профілем виходить з ладу), транзакція копіюється (мітка 7).

Код передається на вхід рядка 2 (мітка 8), потім на вхід пам'яті (мітка 9) (об’єм пам’яті N -1), потім зменшуються 2 рядки (мітка 10) і код видаляється із системи (мітка 11).

Копія транзакції (шаблон запису) відкладається в рядку 2 (ім’я 8), лише якщо в системі вже є пошкоджені вузли з профілями N-1. У цьому випадку час очікування транзакції в цій черзі - це одночасна позиція всіх вузлів з *N* репліками в системі (усі вузли з репліками вийшли з ладу). Частка загального часу очікування від загального часу моделювання є довільною ймовірністю p_0.

Цей дріб відповідає значенню, що зберігається в Q_{a2}(середня довжина рядка 2 - цей рядок може бути будь-яким 0 торгів , або 1).

Група транзакцій 1 (вихідне повідомлення) або транзакція без власника групі, він надсилається на вхід лінії 1 відмови вузла (мітка 12). Далі він займає місце в пам'яті 1 (мітка 13), якщо вільне місце є (тобто багатоканальний пристрій - його ємність дорівнює кількості ремонтників *K*).

Потім рядок 1 зменшується (точка 14), вказуючи на те, що команда оновлення змушує вузол працювати (точка 15). *T* - час відновлення випадкового вузла. Потім ремонтну команду звільняють (16 балів). Якщо транзакції немає в профілях досліджуваної групи (мітка 17), то вона повертається в блок з міткою 5. Якщо транзакція знаходиться в групі 1, то перед цим вона зменшує пам'ять 2 до 1 (мітка 18) - графік відновлюється.

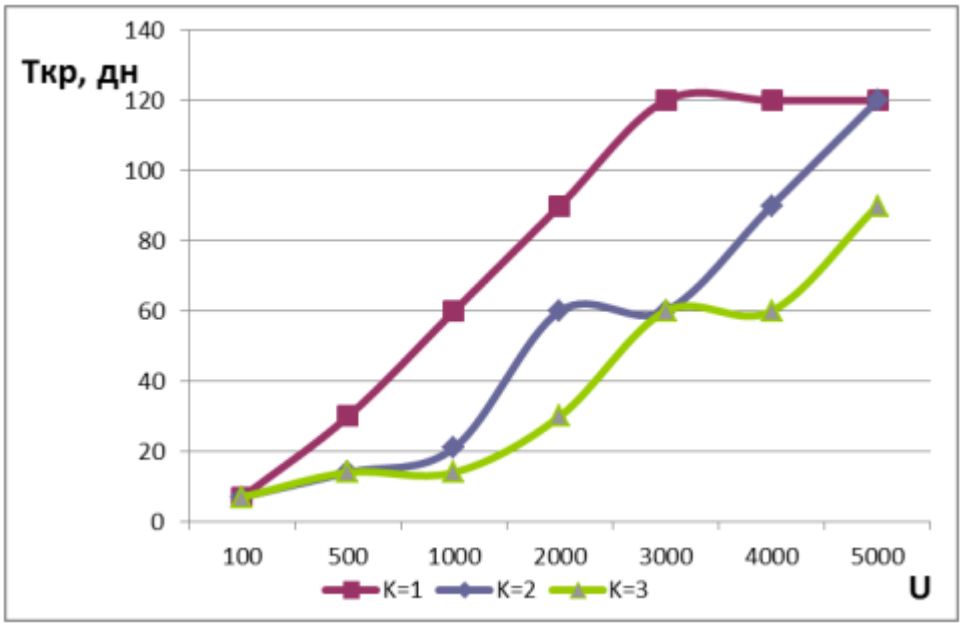

Рис. 2 .10 . Часова $T_{кр}$залежність від *кількості вузлів U* з

Слід зазначити, що моделювання має дуже м’яку частину (і) для *кожного K* , тобто воно $T_{кр}$насправді не змінюється протягом цієї фази. На рисунку 2.11 показано залежність ймовірності від кількості p_0вузлів *U* при *N = 1* (відсутність реплікації).

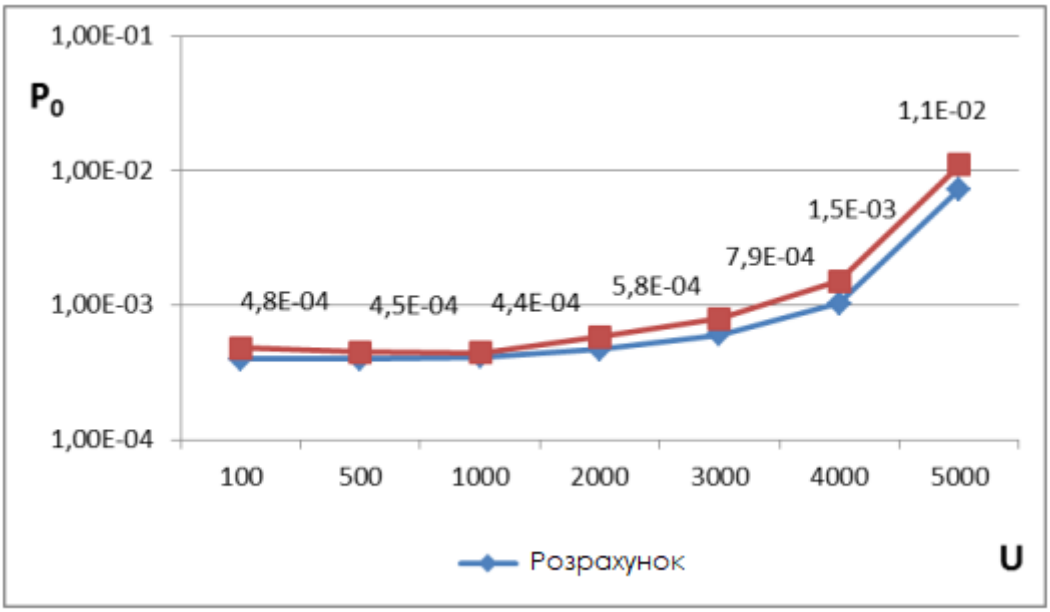

Рис. 2 . 11 . Імовірність p_0 залежить від кількості вузлів $N = 1$ ($K = 2,1$ / $\delta = 3$ міс).U

2.4. Дослідження адекватності моделі керування реплікацією на базах даних типу NoSQL

Дослідити адекватність розроблених моделей процесів планування версії репліки та запису , серія хмарних польових тестів [13] із розміром до 24 вузлів була виконана з використанням бази даних Riak NoSQL [19].

У деяких випадках використання групування у формі хмари. На ринку є багато компаній, які забезпечують інфраструктуру хмарних обчислень. Інструменти віртуальні та спеціалізовані. Виділені хмарні ресурси — це фізично різні вузли та сервери. На відміну від фіксованих вузлів, віртуальні вузли є віртуальними машинами. Оренда виділених серверів значно дорожча, тому в нашому експерименті ми використовували віртуальні вузли, надані компанією DigitalOcean (DO) [23]. Оскільки досліджувані значення (можливість читання застарілого запису, умови очікування завершення

оновлення тощо) оцінювалися, коли користувач працював з одним записом <key / value>, база даних NoSQL розвантажує оперативну пам’ять, що дозволяє їй найняти недорогі віртуальні сервери з невеликою оперативною пам'яттю .

Усі віртуальні вузли, надані хмарними ресурсами, використовують ресурси багатопроцесорних машин із SSD-дисками. Це дозволяє припустити, що інші клієнти DO, налаштовані з віртуальними вузлами на тому самому фізичному сервері, не будуть завантажувати той самий ЦП, на якому працює наша віртуальна машина. Тому продуктивність нашого віртуального вузла менше залежить від фонового навантаження процесора. Під час запуску вузла є кілька варіантів, зокрема можливість перевірити опцію приватної мережі. Коли цей параметр увімкнено, гарантується, що всі вузли, орендовані користувачем, знаходяться в одному центрі обробки даних (базі даних), що означає відсутність підкластера мережі. Тому $\boldsymbol{\mu_{ns}}$параметр - швидкість передачі даних по мережі, підмережі зв'язку - можна не враховувати.

Під час початкового налаштування вузла (Droplet) необхідно вибрати операційну систему (ОС) або образ , попередньо створений користувачем. Використовував попередньо встановлену ОС Ubuntu Server. База даних Riak використовувалася як система NoSQL. Щоб налаштувати та налаштувати Riak, вам потрібно виконати декілька дій на кожному вузлі кластера. Установка сайту «з нуля» займає багато часу. Тому, щоб прискорити підготовку кластера, більшість дій конфігурації системи, описаних у [19], виконувалися один раз на

одному вузлі, потім вузол клонувався, який потім тиражувався на серце інших вузлів.

Окрема версія дій програмування Riak, описана в [19], зберігається як сценарій bash. Після того, як систему було встановлено та налаштовано, усі вузли були підключені до кластера, якому було надано наступні команди Riak:

1) група адміністратора riak додає riak @ <first_ip_node>;

2) групова політика riak-admin;

3) груповий коміт riak-admin.

1 команда була виконана на кожному вузлі, за винятком однієї з приєднаними іншими вузлами (ip_first_node). Після виконання команди 1 необхідно перевірити конфігурацію кластера на кожному з вузлів командою 2, а потім командою 3 зберегти зміни в уже збережених даних з першого вузла на всіх інших. Тим часом система може відповідати на невидимі запити клієнтів.

Для кожного із зразків були розроблені робочі процеси для проведення експериментів. Веб-сайт Riak надає бібліотеки для доступу до системи в Java, Erlang, Python, Ruby, з яких вибирається бібліотека Java. Програми Java транслюються в проміжний байт-код, який виконується на кожній віртуальній машині, що полегшує виконання налагодження програми. Програми запускалися безпосередньо на хмарних вузлах, запити до бази даних робилися на основі статистики та зібраної статистики (журналів). Після експериментів розрахунки були передані в локальну машину для аналізу.

Обговорюється метод з параметрами реплікації $W = R = 1$.. . У моделі узгодження репліки в кінцевому рахунку є вхідним потоком вимог читання для однієї репліки Пуассона було прийнято для кожного параметра λ . Зіставити тест з моделлю, вимоги до читання вона повинна відбуватися автоматично для кожного з *2 ... N* зображень інтенсивності λ . 1 вузол а отримує запит на повернення запису у з серйозністю 1,25 (1/секунду).

Нижче наведені алгоритми програм, що виконуються на вузлах *1 ... N* в дизайн тесту та під час обробки записів для оцінки ймовірності того, що клієнт (процес

читання у вузлі 2 ... *N*) прочитає застарілий запис після періоду розповсюдження нових записів його профілів *N-1* .

На рисунках показана залежність ймовірності *P* того, що клієнт прочитає застарілий запис від λ при різних значеннях *N* . Моделювання базується на результатах природних (експериментальних) і модельних (прототипних) експериментів.

Алгоритм:

```
REC_VAL = 1
ЦИКЛ по N_ITER_W
  TIME_STAMP = CURR_TIME
  ЗАПИСАТИ в БД запис <KEY, REC_VAL>
  ЗАПИСАТИ в журнал <CURR_TIME, REC_VAL>
  REC_VAL += 1
  DELAY = EXPONENTIAL(1.25)
  DELAY -= (CURR_TIME - TIME_STAMP)
  ЗАТРИМАТИ виконання на час DELAY
КІНЕЦЬ ЦИКЛУ
```

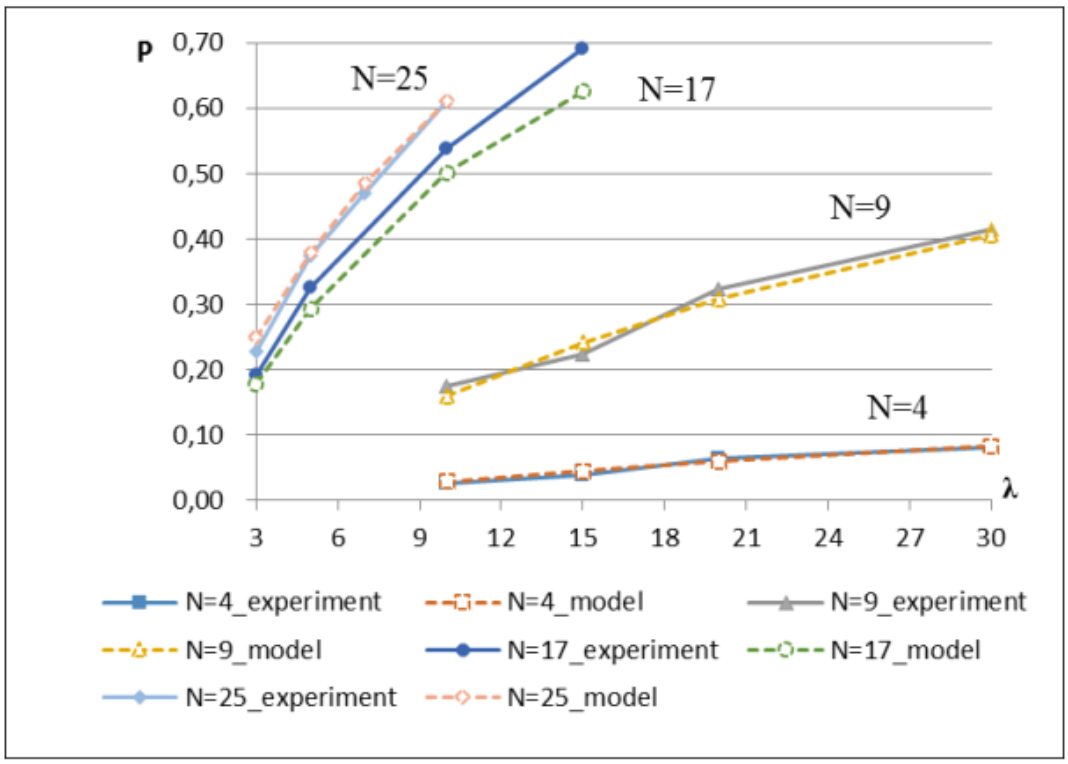

Рис . 2.15 . Залежність ймовірності *P* прочитати застарілий запис в λ від серія дослідів .

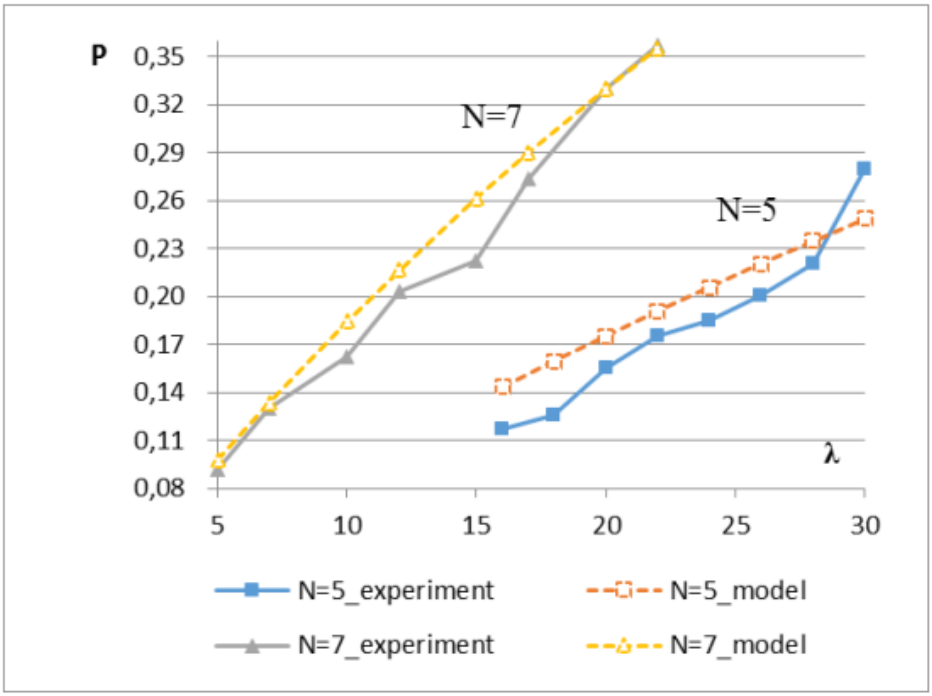

Рис . 2.16 . Залежність ймовірності *P* прочитати застарілий запис в λ від друга серія експериментів .

Середня стандартна помилка двох послідовних експериментів становить 7,86%.

Висновок до другого розділу

1. Розроблено аналітичний приклад рівняння реплік нарешті, у базах даних NoSQL, враховуючи синхронні та асинхронні способи поширення змін. Модель дозволяє розрахувати ймовірність того, що клієнт з часом прочитає застарілий документ , розповсюджуючи нові документи у своїх профілях *NW*

2. Розроблено аналітичну модель сильної координації репліки в базах даних NoSQL. Модель дозволяє оцінити ситуації кінець випадкового інтервалу, необхідного для підрахунку нових версій W їхні профілі, а також час читання записів R, враховуючи це очікування.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КЕРУВАННЯ РЕПЛІКАЦІЄЮ В БАЗАХ ДАНИХ NOSQL

3.1. Архітектура системи та продуктивність

Розроблені моделі можуть бути використані на етапі проектування інформаційних систем, які використовують базу даних NoSQL для зберігання та обробки даних. Однак аналітичні оцінки на основі цих моделей є складними. Щоб полегшити дизайнеру роботу із зображеннями, був розроблений інструмент. Програмне забезпечення дозволяє переглядати результати аналізу та моделювання у візуальному форматі без необхідності знати подробиці використаних зразків. Структура показана на рисунку 3.1.

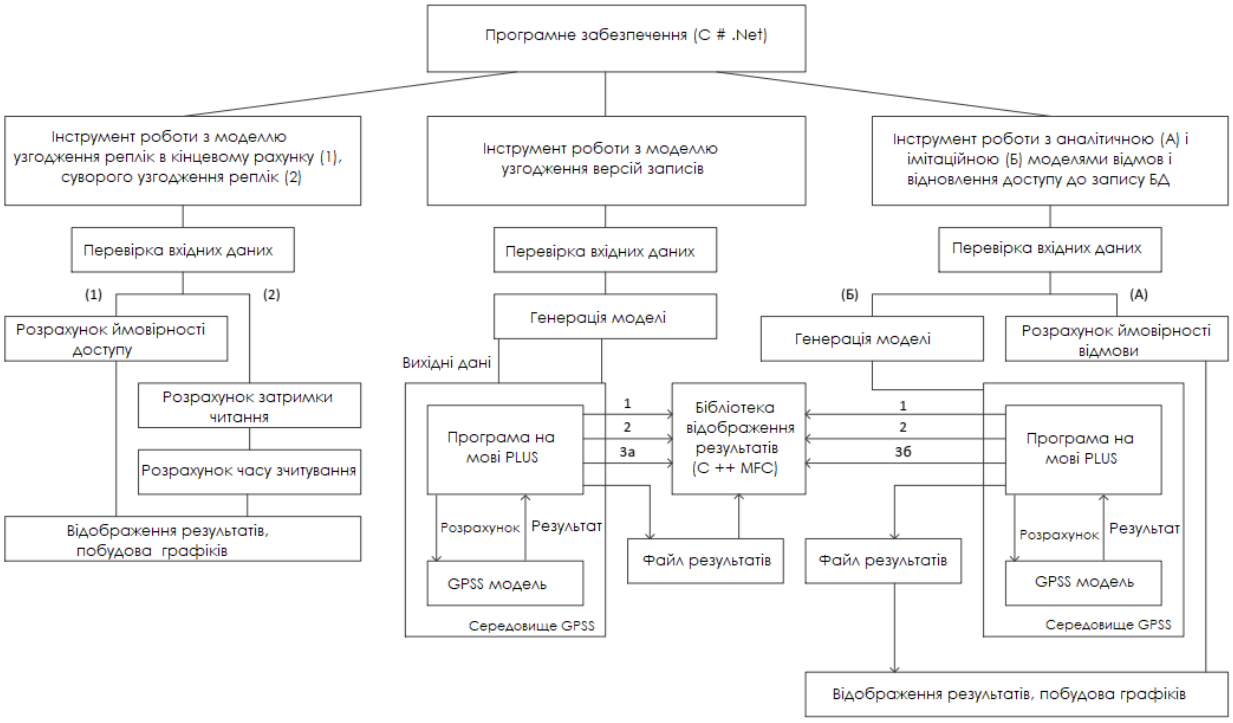

Рис. 3.1 . Структура розробленого програмного забезпечення

Основний програмний модуль реалізовано на C# (мінімальна версія .Net Framework 4.0). Додаток працює під керуванням ОС Windows. Вибір версії .Net зумовлений необхідністю побудови профілів на основі обчислених значень. В області головного вікна (рисунок 3.2) розташовані вкладки «Розбіжність у підсумковому рахунку / складна звірка», «Зміни в документах», «Допуск на відмову».

Програмне забезпечення дозволяє оцінити ймовірність того, що клієнт прочитає застарілий запис під час виділення нового запису в своєму профілі NW - для узгодженості остаточного обчислення (формула (2.3), (2.4)), період, протягом якого потрібно чекати на запит для читання W завершення нового зображення та час читання R очікування зображення - для тісної інтеграції У випадку бази даних NoSQL, розробленої для кожного типу запису (сегмента) статті, існує надійність, дозволена для доступу до непослідовних даних, а також максимально допустимий час очікування (затримка читання). У додатку наведена оцінка фіксованих показників варіації вхідних даних: кількість ітерацій N (кілька значень за;) і діапазон варіації інтенсивності вимог читання від одного параметра (λ). Така конфігурація вхідних даних дозволяє розробнику досліджувати вплив змін у сигналах, коли λ *змінюється* для кожного *N*.

Для обчислення другої похідної системи використовуються чисельні диференціальні методи. Однак точність вбудованих типів даних C# недостатня для таких обчислень. Наприклад, другий тип дозволяє виконувати обчислення з точністю до 10^{-15}. Для вирішення цієї проблеми була використана бібліотека mpreal [17] - модифікація бібліотек mpfr [18] і mpir [19] для C++. Бібліотека mpreal дозволяє регулювати точність обчислень (до 500 символів), але ця бібліотека доступна лише в режимі C++. Для його використання в середовищі C# розроблено бібліотеку StrongDouble.dll (входить до складу програмного забезпечення) – це бібліотека CLR [10], яка реалізує доступ до необхідних функцій mpreal: точний набір статистики, прості математичні операції, відображення. Бібліотеки CLR зручні, оскільки ними можна керувати

та їх можна використовувати на будь-якій мові програмування, яка підтримує .Net. Точність оцінок числовими дисперсійними методами *безпосередньо залежить від кроку в дисперсійній таблиці h* . У програмних алгоритмах значення $h = 10^{-15}$точності алгоритмів встановлюється рівним 200 розрядам.

На рисунку 4.2 показаний приклад оцінки ймовірності того, що клієнт прочитає застарілий документ під час розповсюдження нових документів NW. При *великих значеннях N і* λ дифузія може досягати більш високих значень.

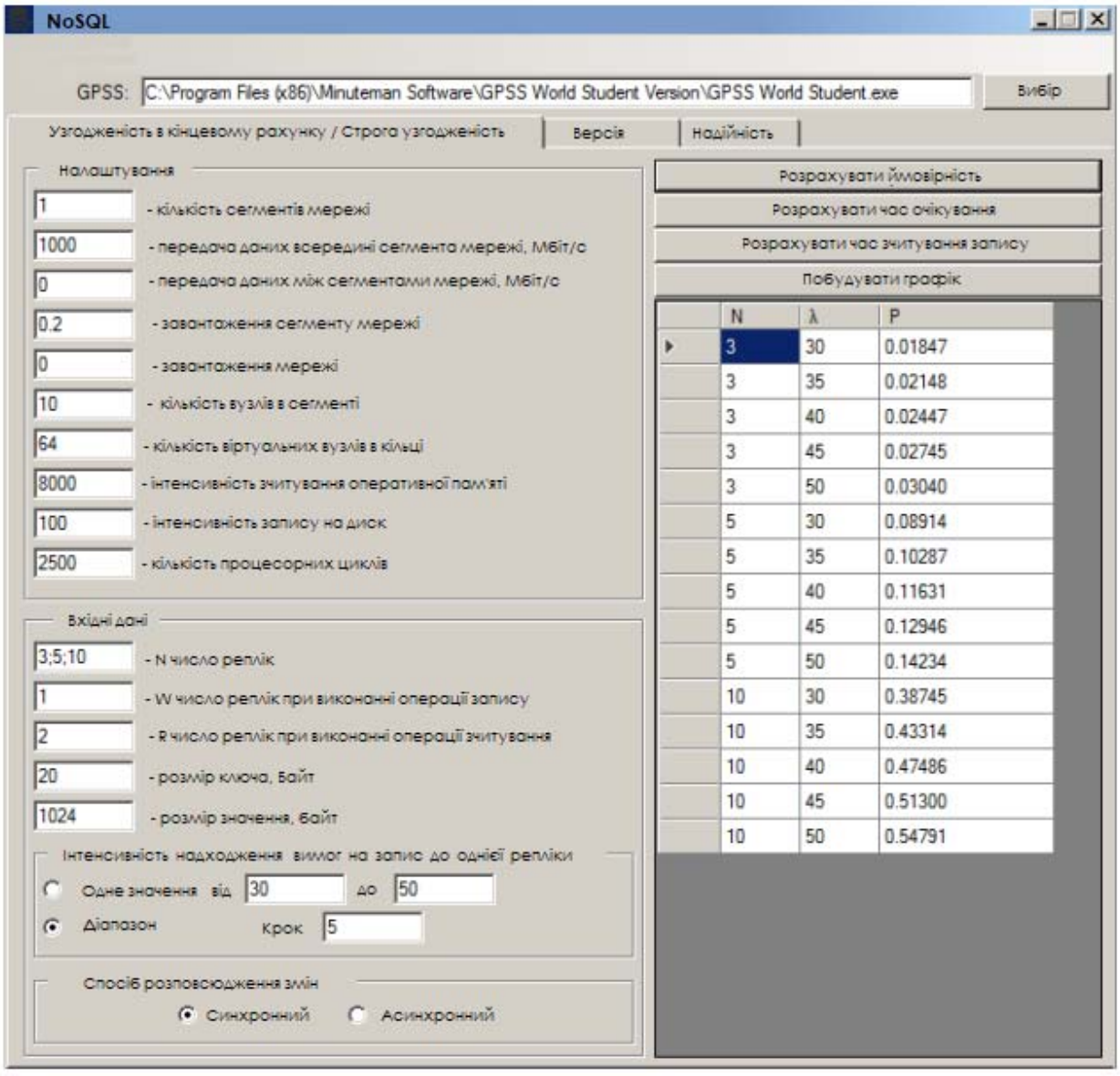

Рис . 3.2 . Основна форма системи

Для моделі зміни записів можна вказати відразу кілька значень вхідних параметрів для одного процесу моделювання - це число клієнтів K і число k (в моделі r), яке враховує час користувача використання аналізує результати процесу (за допомогою маркера крапки з комою). Така конфігурація вхідних даних дозволяє розробнику досліджувати вплив змін у сигналах, коли k змінюється для кожного K .

Вікно «Розклад» (друга вкладка, рис. 3.2) містить елементи керування для налаштування імітаційної моделі: змінна моделі (1 або 2), кількість клієнтів, число k, час моделювання (для одного запуску), час окремої функції розподілу ймовірностей клієнт застосовує одну версію запису (варіант 1) або одне оновлення запису (варіант 2). Саме моделювання виконується у форматі GPSS. Для автоматичної заміни джерел даних у тексті моделі розроблено програму на мові PLUS [20]. Інструмент автоматично виконує запис програми на PLUS із зразком на GPSS після запису вихідних даних. На рисунку 3.3 наведено схему програмного забезпечення з прикладом запису змінних.

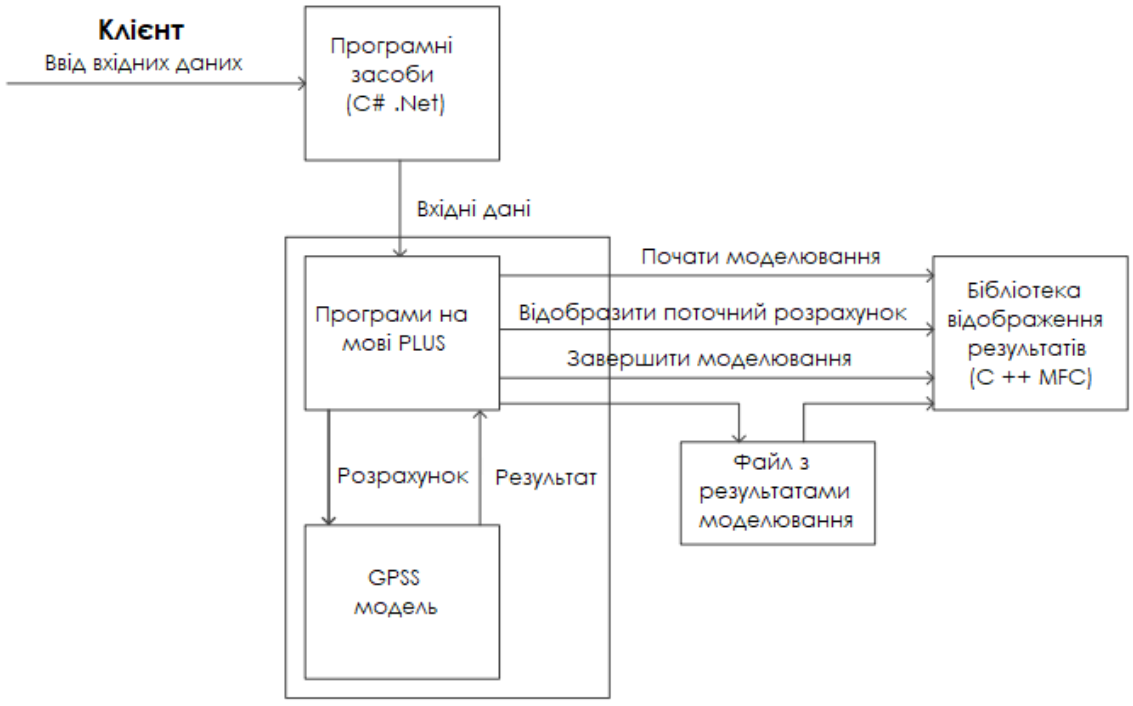

Рис .3.3 . Приклад робочого процесу обладнання та змінних рахунок _

Мова PLUS надає можливість викликати програми C із програмного коду GPSS за допомогою функції CALL. Для демонстрації процесу моделювання та результатів (блок «Показати бібліотеку результатів» на рисунку 3.3) створюється бібліотека C++ (фреймворк MFC [12]) NoSQLGPSSLib.dll (доступна в програмному забезпеченні).

Нижче наведено бібліотечні функції, цитовані з PLUS:

1. SetResultFileName - функція, яка приймає рядковий аргумент - шлях до файлу результатів моделювання. Викликається перед початком моделювання для ініціалізації імені файлу результатів моделювання.

2. SetTotalProgress - функція, яка приймає числовий аргумент - кількість виконаних зразків. Встановлює загальну кількість прогонів («Почати моделювання» на рисунку 3.4).

3. SetProgress - функція, яка приймає числовий аргумент - числове значення поточного виконання моделі. Він встановлює прогрес відповідно до кількості запусків, що виконуються на даний момент (див. стрілку «Показати поточну швидкість» на рисунку 3.4).

4. SetFinishVersions - функція без параметрів, сповіщає систему про те, що моделювання завершено та результати моделювання потрібно відобразити (див. стрілку «Завершити моделювання» на рисунку 3.4).

Для спрощення програми PLUS в головний модуль передається не сума числа k, а час, за який клієнт обчислює суму обчислених середніх результатів змін операцій і записів.

На рисунку 3.4 показано приклад вікна з результатами моделювання, отриманими за допомогою програмного забезпечення. Моделювання було виконано з вхідними даними, показаними на рисунку 3.2.

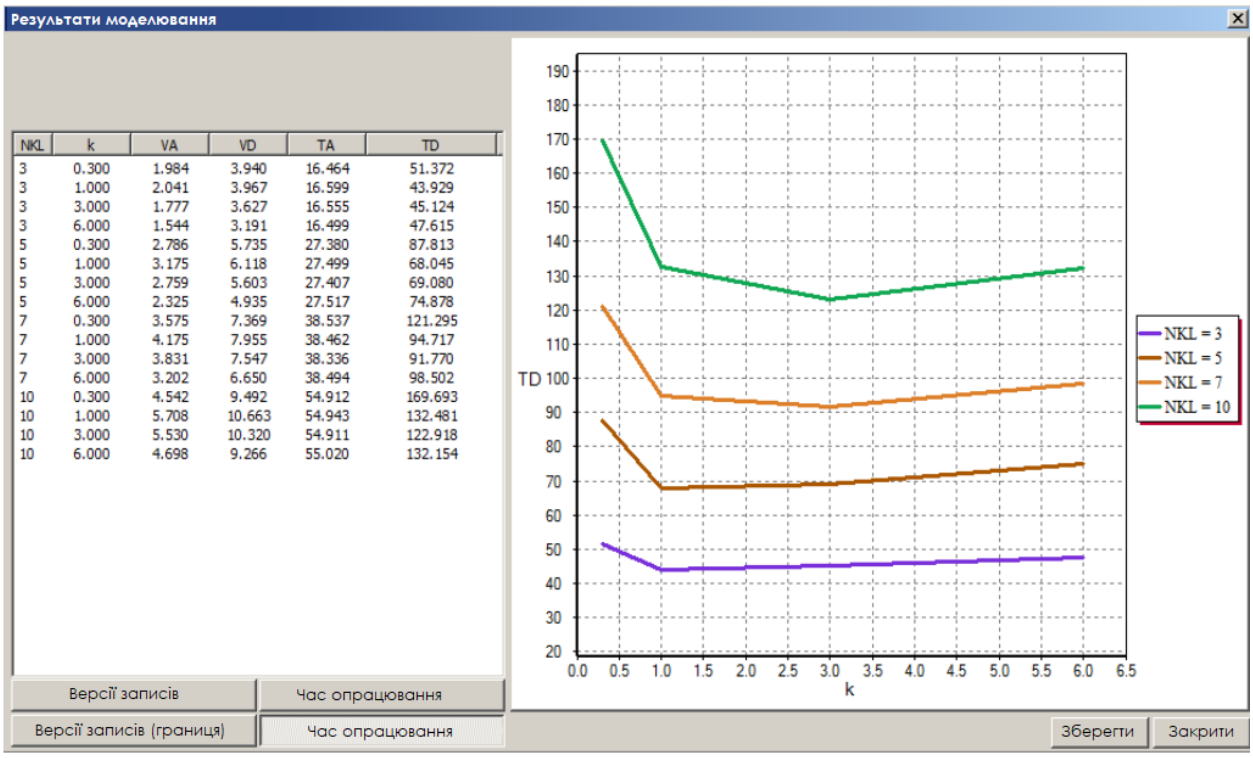

Рис . 3.4 . Приклад робочого процесу обладнання та змінних _

3.2. Використання програмного забезпечення на етапі проектування системи

Використовувалися розроблені моделі та програмне забезпечення коли мова йде про інформаційну систему (ІС) « Статистика пацієнтів – NoSQL», яка має забезпечити зберігання та обробку великої сукупності відомості про захворювання, пацієнтів, лабораторні дослідження тощо.

Завданням аналізу був вибір матеріалів (N, W, R). розподіленої бази даних NoSQL з урахуванням конкретних аспектів предмета і вимоги замовника до показників ефективності, інтеграції та контроль помилок.

При вирішенні проекту реалізовано наступне кроки:

1. Опис предмета.

2. Причини відмови від реляційної моделі даних на користь технології NoSQL не існує.

3. Визначте параметри, доступні для систем NoSQL використовується для реалізації досліджуваної інформаційної системи.

4. Побудова системи зберігання інформаційних технологій (опис агрегати).

5. Працювати, взаємодіяти з контролювати помилки інформаційних технологій проектування з використанням прикладів, розроблених у попередньому розділі про бази даних NoSQL. Параметри реплікації на основі вимог для продуктивності, інтеграції та відмовостійкості в ІБ, а також розглянуто виконані тести.

Розроблена система призначена для збору, передачі, зберігання та аналізу даних хвороби людей і тварин разом з інформацією, що стосується них зразки та лабораторні дослідження. Це дозволяє маніпулювати даними за допомогою. хвороби людини (медичний модуль системи), ветеринарні хвороби (в ефірному модулі системи) та відповідні лабораторні аналізи (лабораторний модуль). Одна з особливостей системи інтеграція галузей епідеміології та ветеринарії, а також а також поєднання кожного з цих компонентів з лабораторними даними.

Модуль обробки був розроблений для збору даних, як описано уражених осіб (зазвичай це особливо небезпечні захворювання, . які вимагають негайного повідомлення, як-от сибірська виразка, чума тощо) і відповідно до уражених груп (зазвичай це захворювання низького ризику, . які вимагають регулярного надання звітів, наприклад у щомісячних звітах).

Інформація про інфіковану особу включає демографічна інформація про людину, інформація про екстрене повідомлення, діагноз (початкова , клінічна, кінцева), клінічні симптоми, дані епідеміологічного дослідження, відомості про контакти особи, її рецептури, інформацію про зібрані зразки та результати аналізу, . інтерпретація цих даних.

Адміністративна одиниця описує заяву групи (наприклад регіон), часовий інтервал (наприклад, один місяць), перелік випадків і кількісні

характеристики (наприклад, кількість пацієнтів віком до 1 року). або кількість хворих вагітних).

Модуль ветеринарної імунології містить інформацію з галузі. діагнози, опис структури та характеристики такого госп його продуктивність (наприклад, тип пасовища, схема годівлі та тощо) , опис складу стада чи зграї з посиланням на вид (у разі великої рогатої худоби зі специфічними ознаками тварин), відомості про клінічні та ознаки конкретних порід або особин, відомості про вакцинація тварин, інформація про зібрані зразки та для аналізу, результати аналізи та їх інтерпретація.

Інформація про відібрані проби на лабораторне дослідження, записані експерименти та результати, відповідно до культури, . виведені з прикладів, про посів і розмноження, на результати випробувань , які зразки я зберігаю , інформацію про співвідношення зразків із ними випадки захворювань, інформація про різні види повсякденної діяльності в лабораторії (наприклад, відокремлення зразків, знищення зразків, транспортування для подальшого навчання в іншому закладі).

Лабораторний модуль дозволяє персоналу лабораторії отримати повідомлень про випадки захворювання людини чи ветеринара, кількість яких збільшується отже, ймовірність правильної інтерпретації виконаного аналізу моделі. IN отже, медичний і ветеринарний модулі дозволяють доступ до. laboratory x data x , що забезпечує доступ до останніх діагностика на основі детальних лабораторних даних. Тому, забезпечена інтеграція тваринного та медичного модулів лабораторія

Крім того, є можливість поєднати лікувальний і. хвороби тварин перемежовуються, і як хвороби записуються так звані епідемії , тобто група хвороб, що виникають приблизно в один час і майже в одному місці. Такий процес дозволяє стежити за поширенням хвороб тварин.

Реальність обсягу предмета вимагає спільного аналізу кількох факторів для проведення якісного епідеміологічного дослідження.

Вже згадана система складається з двох блоків обробки даних, які показано на рисунку 3.5 .

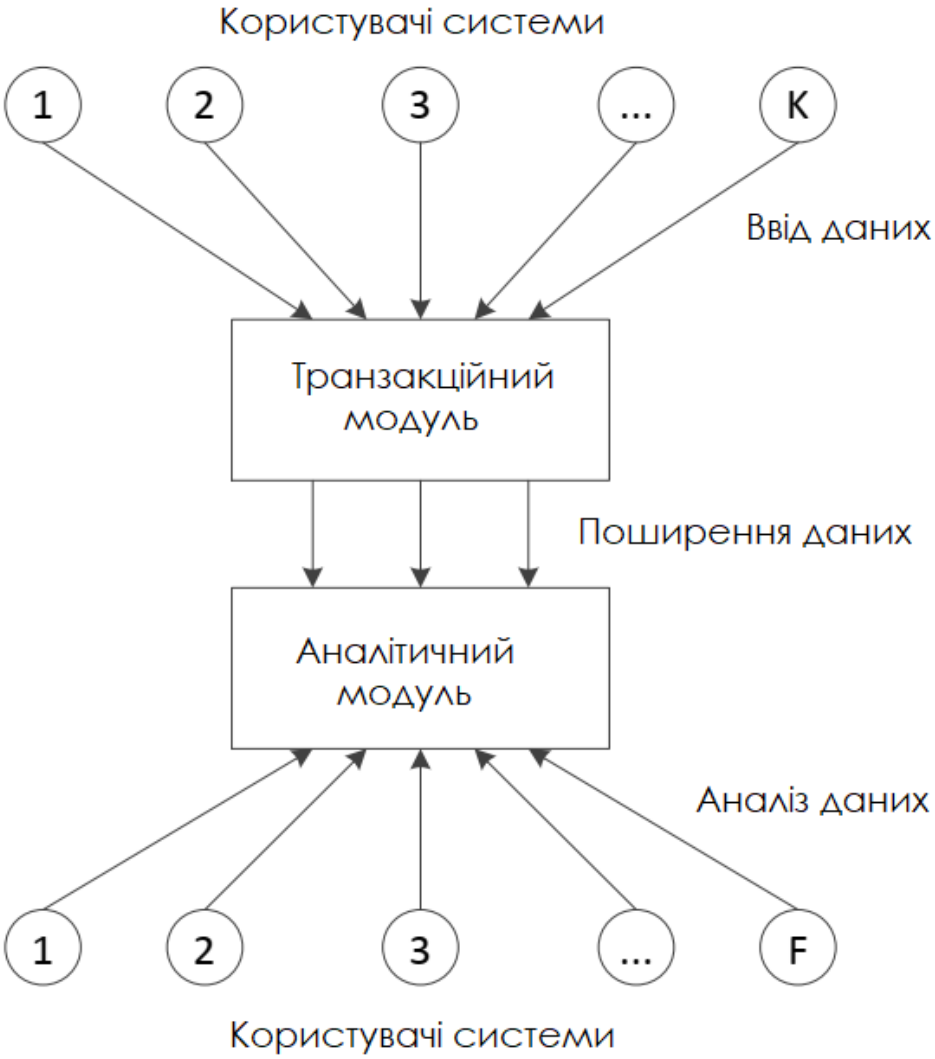

Рис. 3.6. Компонент бази даних, відповідальний за побудову та підтримку

1. Бізнес-модуль - це вузол, який з'єднується, взаємодіє з ним. пересилання даних до модуля аналізу. Зараз Елемент системи може працювати на будь-якому надійному інтерфейсі СУБД, що підтримує транзакції ACID.

2. Аналітичний модуль - група (кластер) задіяних вузлів аналітична обробка даних. Це одна з ключових особливостей умови. Це дозволяє представляти зібрані дані різними способами типи даних, створення різних зображень і відображення відомості про географічні об’єкти на карті. Зараз

елемент системи допускає паралельну обробку аналізу питання та надає інформацію різного рівня складності.

У штатному режимі роботи системи кількість вхідних запитів аналітичний модуль невисокий , з навантаженням можна впоратися група вузлів. Однак під час пандемії навантаження на систему може бути непосильним Вода Наприклад, при спалаху захворювання, пов'язаного з вірусом Ебола [2 3] у Західній Африці кількість випадків зросла в 100 разів за 4 місяці [1 4].

У розробці таких захворювань є багато дослідників у всіх країнах буде зацікавлений в отриманні актуальної інформації про швидкість передача, кількість хворих тощо. Це буде розглянуто далі аналітичний модуль системи « Статистика пацієнтів - NoSQL».

3.3. Причина вибору технології NoSQL

До недавнього часу перегляд веб-сторінок був широко поширеним використовувати стандартну систему керування базою даних контактів (RSDB) (RSDB). Зараз більшість даних зберігається в RDBMS. ВІН пояснюється поверхневою простотою реляційної моделі існування такої дин і в той же час проста у вивченні мова пошуку даних, така як SQL, також має доступність оптимізаторів для транзакцій і виконання запитів до бази даних, які забезпечують їх «подрібнення» на більш дрібні завдання та паралельні одна одній виконання цих завдань на багатопроцесорних або багатомашинних комплексах.

Давайте розглянемо основні проблеми з дизайном даних, які можуть виникнути виникає при використанні реляційної бази даних у робочому процесі дослідницький модуль:

1. Проблема неорганізованості та неефективності. як Як зазначено в розділі 1 , це очевидно з того факту, що основи відносин дані не дозволяють зберігати

агрегати. Наприклад, показати відомості про пацієнта, місцезнаходження, місцезнаходження, . госпіталізація тощо, має бути скомпільовано програмістом в оперативну пам’ять дані з кількох таблиць. Оскільки кількість столиків різко зросла часто дизайнер просто забуває про призначення того чи іншого столу, . при виконанні запиту стає складніше об'єднувати таблиці, а значить, істотно ускладнюється структура групи. Кілька таблиць мають функції з’єднання вони виготовляються відносно повільно. Теоретично можливий варіант за допомогою одного плоского столу. Але в даному випадку таблиці немає він буде в третьому нормальному стані і матиме свої недоліки , такі як неефективність, можливі протиріччя тощо При цьому розмір бази може збільшуватися в геометричній прогресії.

2. Проблема поширення інформації. Як зазначалося в розділі 1, зі збільшенням кількість даних, що зберігаються, виконує функцію поділу таблиць бази даних дані на різних серверах, об’єднаних у кластер. але паралельний бази даних взаємозв'язків систем виявилися простими масштабованість [5]. При складанні складних запитів в результаті ефективність системи значно знижується м і масовий обмін даними між серверами кластера [6].

3. Проблема забезпечення допустимості помилок. Як зазначено в розділі 1 ст у паралельних базах даних кількість параметрів невелика , що не може гарантують високу толерантність до помилок. Більш того, якщо вони відмовляться кожного вузла необхідно перезапустити всю систему.

Бази даних NoSQL не мають зазначених недоліків , тому доцільно використовувати цей тип бази даних у цьому проекті.

3.4. Аналіз показників ефективності контролю реплікації та відмови в доступі до запису бази даних

У цьому розділі оцінюються такі показники ефективності:

1) середній час очікування запитів і читань завершення відновлення параметрів W для методу критичної апроксимації;

2) ймовірність того, що клієнт з часом прочитає застарілий документ розповсюдити нові записи у своїх профілях NW щодо ситуації упорядкування реплік у кінцевому документі;

3) можливість відмови в доступі до документа БД.

Визначено вимоги до режимів відповідності реплік модуля аналізу фреймворку « Статистика захворювань – NoSQL»: розд Human Word - єдність у остаточному документі, розділ Human Word Приклад - . сильна взаємодія.

Клієнт встановлює розмір кластера до 15 вузлів (максимум Рекомендований розмір кластера Riak становить 5 вузлів [8]). Додавання/видалення вузлів (серверів) працюють у Riak за допомогою однієї команди, а дані він автоматично перерозподіляється між усіма вузлами у фоновому режимі. ВІН це значно спрощує процес зміни розмірів системи. Віртуальні вузли. їх функція схожа. Відповідно до [2 3] рекомендовано встановити кількість віртуальних вузлів у кільці так, щоб на кожному вузлі зарезервовано принаймні 10 розділів (віртуальних вузлів). Отже, давайте оцінимо його 256 відповідних кілець.

Для аналізу були використані наступні значення атрибутів апаратні властивості:

- швидкість читання даних з оперативної пам'яті - 8000 Кб/с;

- продуктивність процесора - 2000 мільйонів робочих місць на село.

Під час спалаху можуть бути мережі передачі даних і дискові масиви. дуже скупчено, тому були використані такі параметри:

- швидкість передачі даних в мережі - 1 Гбіт/с (Швидкість за замовчуванням);

- код мережі передачі даних в базі даних - 0,6;

- швидкість введення/виведення диска - 40 Мбайт/с;

- кількість груп, які готують групи - 1;

- час безвідмовної роботи при відмові одного вузла - 90 днів;

- якщо вузол виходить з ладу: ймовірність випадкового збою системи 0,835, ймовірність збою апаратного забезпечення 0,15, . ймовірність виходу з ладу диска - 0,015;

- час перезавантаження операційної системи - 10 хвилин;

- час відновлення апаратного забезпечення вузла - 4 години;

- час відновлення диска - 5 годин;

Змусити систему запрацювати на території з населенням 42,8 млн осіб через 5 років. Якщо припустимо, що в середньому людина хворіє на будь-яку хворобу раз на 5 років хвороби, зареєстровано в системі, тоді середню кількість записів можна розрахувати як (42,8 млн / 5) · 5 · 3 = 128,4 млн (3 означає, що один запис зберігається на розділ «Людська справа» та два записи – у розділі «Людська справа».

За даними [1 5], є 476 районів, 165 підрайонних міст, ми маємо запис у систему 476 + 165 = 641. Середній рівень відновлення кожного запису можна визначити за 0,0012 1/с (1 раз за 13,5 хвилин). Середній розмір колекції буде визначено наступним чином наступним чином: розділ "Human Case" - 6000 байт, "Sample Human Case" - 2125 сума байтів Ці значення взяті з повного пакета зображення втрачені частини. Тоді середній розмір записів становить близько 3417 байт ((6000 + 2 · 2125) / 3). Один вузол зберігає 128,4 мільйона / 15 = 8,56 мільйона записів, середня швидкість запису на вузол - 8,56 млн · 3417 байт = 27,25 Кількість Гбайт.

Давайте розглянемо проблеми узгодженості, які можуть виникнути, коли. робота з системою:

1. Може вагатися читати примітки з розділу «Людська справа». Зразок», оскільки під час епідемій рівень читання та маніпуляції подробиць розділу багато.

2. З великими труднощами читаю інформацію з розділу «Людська справа». застарілі дані можуть бути повернуті користувачам (зі старого тексту), оскільки цей аспект визнається в остаточному тексті.

Проведено дослідження показників паралельності та контролю похибок яка була створена за допомогою розробленого програмного забезпечення. На рисунку 3 . 7 показує залежності середнього часу очікування (T). вимоги та для оновлення параметрів W у кінці читання необхідні труднощі для читання нотаток з розділу «Людина». Приклад справи" для одного прикладу (λ). $W = R = N / 2 + 1$, якщо N дорівнює, і $W = R = (N + 1) / 2$, . якщо N непарне.

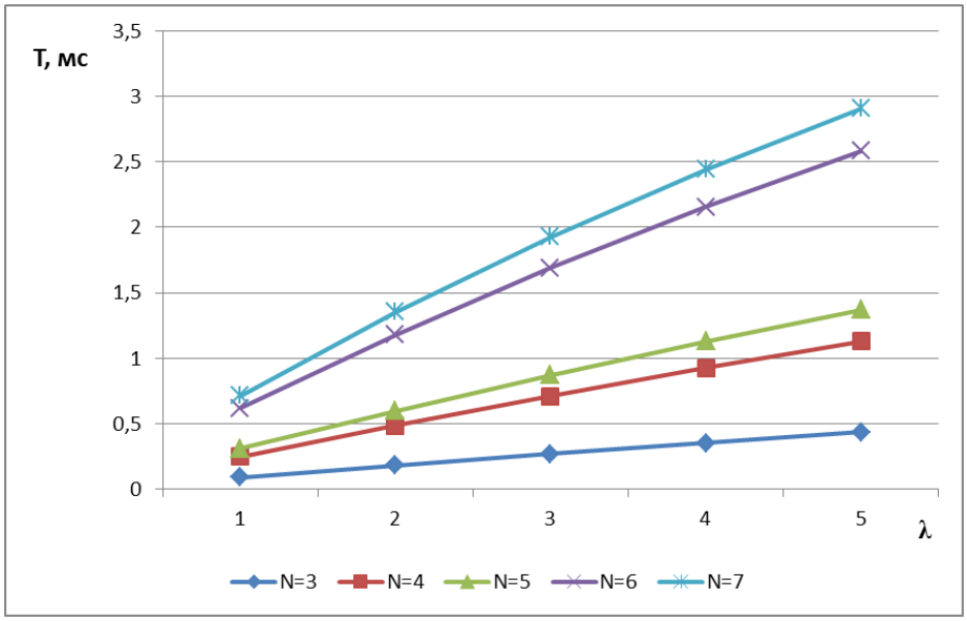

Рис. 3.6. Фактори, що залежать від інтенсивності середнього часу очікування T (мс).

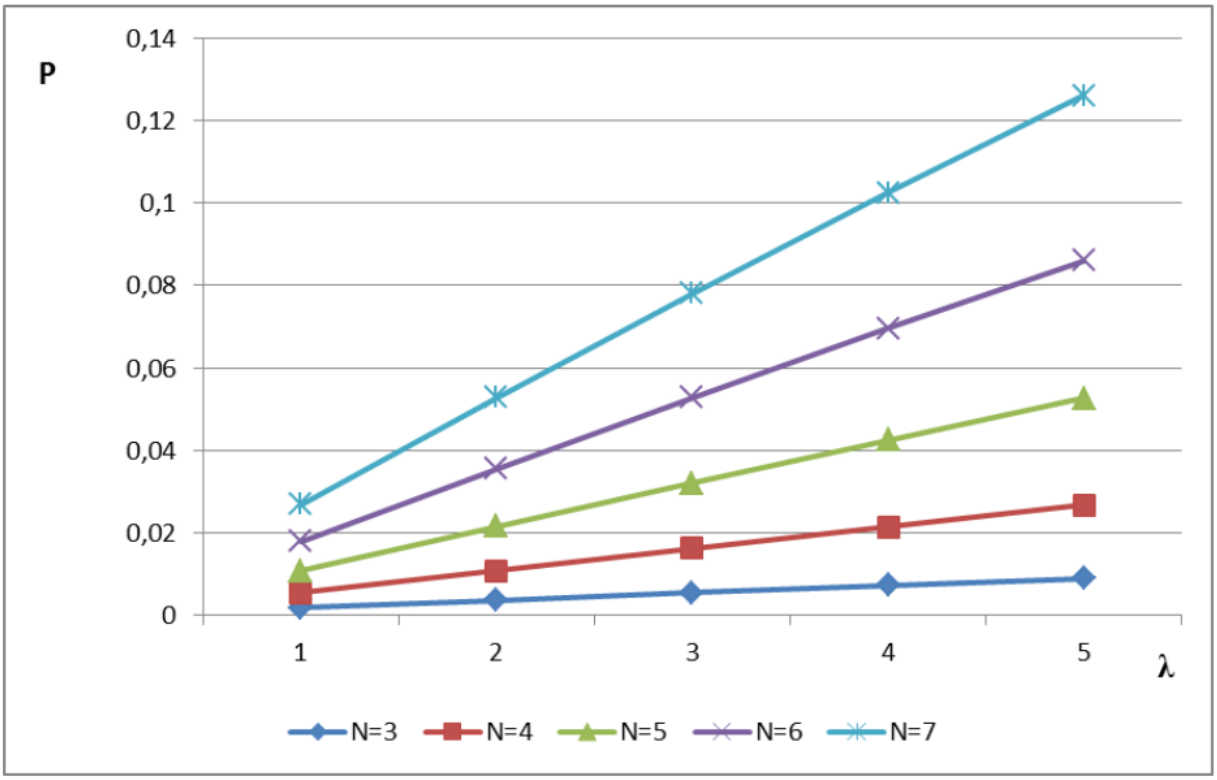

Рис. 3. 7 . Залежність ймовірності *P* від інтенсивності надходження вимоги до читання для *одного вузла групи λ (1/c)* у різних значеннях кількість примірників *H*

Вибір реплікації блоків зберігання NoSQL було зроблено на основі вимог високої затримки читання, узгодженості та … відмовостійкості. Замовник системи поставив такі вимоги:

1. Затримка читання (час очікування читання запиту після зміни параметрів *W*) - не більше 2 мс (це важливо, якщо під час виконання запиту зчитується велика кількість записів).

2. Імовірність неузгодженості даних (ймовірність того, що клієнт прочитає застарілий запис під час розповсюдження нових записів за його параметрами NW) не перевищує 0,1.

Висновки до третього розділу

1. Наведено обґрунтування вибору технології NoSQL для реалізації модуля аналізу системи.

2. Обговорюються варіанти баз даних NoSQL, враховуючи такі фреймворки: MongoDB, Cassandra, Riak. Аналіз показує, що база даних Riak краща для обраної статті.

3. Деталі основних властивостей документів JSON наведено в розділах «Human Word» і «Human Word Model» створеної структури.

4. Були визначені залежності для середнього часу очікування читання документів з бази даних, ймовірності читання застарілих документів, ймовірності відмови в доступі до документа бази даних і ступеня віку читання питань.

ВИСНОВКИ

1. Розроблено аналітичні моделі процесів узгодження реплікації які, дозволяють обчислити ймовірність читання застарілого запису з бази даних NoSQL для синхронних і асинхронних сценаріїв створення нових записів.

2. Запропоновано аналітичну модель для точності реконструкцій, що дозволяє обчислювати параметри випадкового часу очікування, зчитуючи нові групи серверів початку запису .

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Маклаков С.В., та ін. BPwin та ERwin: CASE інструменти для розробки інформаційних систем. - М.: Бесіда-міфи, 1999. - 295 с.

2. Федорова Д.Є., Семенов Ю.Д., Чижик К.Н. Технології CASE. - М.: Гаряча лінія телекомунікацій, радіо і зв'язку, 2005. - 160 с.

3. Самойлов, В. Д. Побудова комп’ютерної моделі / В. Д. Самойлов. – К.: Фіз. - 2007. - 198 с.

4. Лаврищева, Є. М. Стратегії планування: теорія, технологія, практика / Є. М. Лаврищева. – К.: Фіз. - 2006. - 451 с.

5. Чмир, І. О. Системне моделювання в середовищі UML (Unified Modeling Language) : навч. посібник / І. О. Чмир, М. Ф. Ус; Черкаська акад. менеджменту – Черкаси: ЧАМ, 2004. – 100 с.

6. Андон, Ф. І. Логічні моделі інтелектуальних інформаційних систем / Ф. І. Андон, А. Є. Яшунін, В. А. Резниченко. - К: Фізіологія.-1999.-397 с.

7. Цитник В.Ф., і 1990. Системи отримання тексту. – К.: Техніка, 2005. –164с.

8. Мітчелл М., Олдем Д., Камуель А. Програмування для Linux. Професійна освіта. - М.: Вилямець, 2002. - 288 .

9. Лапінський В.В., Габруцев В.Ю. Перебудова: Посібник для студ. – К.: Вища школа, 2007 .

10. Таненбаум Е., Вудхалл А. Цілі продуктивності: підтримка та впровадження. Класичний CS. – КПб.: Питеп, 2006. – 576 .

11. Кодд, Едгар Ф.: Модель категорії даних для великих розподілених баз даних. в: Повідомлення ACM 13 (1970), червень, №. 6, стор 377-387 в.