Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)
https://doi.org/10.36930/40340411
Article received 19.04.2024
Article accepted 23.05.2024
UDC 004.[75+031.6]
Correspondence author: D. P. Kushniruk dmytro.kushniruk.mnpzm.2022@lpnu.ua

А. В. Фечан, Д. П. Кушнірук
Національний університет "Львівська політехніка", м. Львів, Україна

ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ CRDT-ТЕХНОЛОГІЙ У СИСТЕМАХ МОНІТОРИНГУ ІНТЕРНЕТУ РЕЧЕЙ

Оцінено ефективність використання конфлікт-вільних реплікаційних структур даних CRDT (англ. *Conflict-free Replicated Data Types*), для забезпечення узгодженості та цілісності даних у системах моніторингу Інтернету речей (IoT). Виклики, пов'язані з моніторингом великої кількості розподілених IoT-пристроїв, вимагають ефективного управління даними, тому це дослідження розглядає технологію CRDT як альтернативу, що може спрощувати цей процес. CRDT-структури – це розподілені типи даних, які забезпечують сильну кінцеву узгодженість (англ. *Strong Eventual Consistency*, SEC), а також мають такі властивості, як комутативність та ідемпотентність. У межах дослідження реалізовано та проаналізовано такі CRDT-структури як векторний годинник (*VClock*), багатозначний регістр (*MVReg*) та реплікаційну карту (*Map*), що дало змогу оцінити їх вплив на ефективність синхронізації даних та загальну продуктивність системи. Створено тестовий стенд з трьома IoT-пристроями для емуляції георозподіленої мережі, що дало змогу здійснити експериментальне дослідження, результати якого вказують на певні переваги використання технології CRDT, особливо в особливостях масштабованості та стійкості системи до збоїв у мережі. Розглянуто такі метрики, як тривалість досягнення глобальної узгодженості, частка успішних синхронізацій, кількість вирішених конфліктів, середня тривалість вирішення конфлікту, час відновлення системи після відмови та вплив відмови на доступність даних. З'ясовано, що застосування технології CRDT сприяє підвищенню ефективності процесів реплікації та синхронізації даних, мінімізуючи вплив затримок мережі та втрату інформації внаслідок перебоїв у роботі пристрою або мережі. Проте варто відзначити деякі обмеження, пов'язані з використанням технології CRDT, зокрема, збільшення витрат енергії та відносно високі вимоги до обчислювальних ресурсів обладнання. Визначено, що використання статично виділеної пам'яті призводить до зростання складності використання таких структур, а саме обмежується можливість динамічно змінювати кількість вузлів у системі. Такі обмеження вимагають подальшого дослідження та оптимізації архітектури систем IoT, що використовують технологію CRDT, для забезпечення їх більшої енергоефективності та зниження вимог до обладнання.

Ключові слова: синхронізація даних; реплікаційні структури даних; масштабованість систем; відновлення після відмов.

Вступ / Introduction

У сучасному світі Інтернет речей (IoT) стає щораз важливішим елементом в багатьох особливостях повсякденного життя, від промислового моніторингу до розумних будинків і міського управління [6, 8, 11, 13]. Розширення мереж IoT призводить до значних викликів у сфері управління даними, особливо щодо забезпечення їх узгодженості та надійності [7]. У традиційних підходах до управління даними велике значення надається централізованому контролю, що часто призводить до затримок в обробленні запитів і складностей координації у розподілених мережах, які характерні для систем IoT. У цьому контексті конфлікт-вільні реплікаційні структури даних (англ. *Conflict-Free Replicated Data Types*, CRDT) [12] пропонують новий підхід до роботи з розподіленими системами, який може спростити управління даними в IoT-пристроях, забезпечуючи їх узгодженість і надійність роботи.

Технологія CRDT – це клас реплікаційних структур даних, який дає змогу різним вузлам мережі незалежно вносити зміни в даних без потреби негайної координації з іншими вузлами. Ця можливість є особливо цінною в умовах, коли постійний зв'язок між вузлами неможливий або непрактичний, що є частим випадком для IoT-пристроїв, які розміщені у віддалених або важкодоступних локаціях.

Об'єкт дослідження – моніторинг IoT-пристроїв.

Предмет дослідження – підходи до застосування технології CRDT для забезпечення узгодженості даних у системі моніторингу IoT-пристроїв у реальному часі.

Мета роботи – об'єднати принципи роботи CRDT-технології та моніторингу IoT-пристроїв для аналізу нових особливостей їх комплексного застосування.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

1. Проаналізувати наявні підходи до забезпечення узгодженості даних у системах моніторингу IoT, що дасть змогу ідентифікувати ключові виклики та обмеження наявних рішень.

Інформація про авторів:
Фечан Андрій Васильович, д-р техн. наук, професор кафедра програмного забезпечення.
Email: andrii.v.fechan@lpnu.ua; https://orcid.org/0000-0001-9970-5497
Кушнірук Дмитро Петрович, магістр, кафедра програмного забезпечення.
Email: dmytro.kushniruk.mnpzm.2022@lpnu.ua; https://orcid.org/0009-0004-6653-0116

Цитування за ДСТУ: Фечан А. В., Кушнірук Д. П. Оцінювання ефективності застосування CRDT-технологій у системах моніторингу інтернету речей. Науковий вісник НЛТУ України. 2024, т. 34, № 4. С. 86–92.
Citation APA: Fechan, A. V., & Kushniruk, D. P. (2024). Evaluation of the effectiveness of CRDT technologies in iot monitoring systems. *Scientific Bulletin of UNFU*, 34(4), 86–92. https://doi.org/10.36930/40340411

2. Дослідити технологію CRDT як інструмент для забезпечення узгодженості даних, що дасть змогу вирішувати конфлікти даних без центрального координаційного вузла, знижуючи затримки та підвищуючи ефективність синхронізації в георозподілених системах.
3. Розробити підхід до застосування CRDT-технології в IoT-системах моніторингу, що уможливить автоматичне вирішення конфліктів даних, підвищення стійкості системи до мережевих збоїв і відмов, а також поліпшення загальної відмовостійкості та доступності системи.
4. Протестувати та здійснити валідацію розробленого підходу, що забезпечить підтвердження теоретичних переваг технології CRDT через практичні експерименти.
5. Проаналізувати вплив CRDT-технології на продуктивність та масштабованість IoT-системи, що дасть змогу оцінити, як застосування технології CRDT впливає на здатність системи швидко адаптуватися до змін у контексті мережі та кількості керованих пристроїв.

Аналіз останніх досліджень та публікацій. Автори роботи [14] розглядають проблеми управління базами даних, зокрема, зосереджуючись на контролі багатоверсійного паралелізму (англ. *Multiversion Concurrency Control,* MVCC). У цій роботі запропоновано новий підхід з використанням алгоритму генерації графів для перепланування транзакцій у разі конфліктів. Цей метод частково відкочує транзакції у разі збою системи або мережі, замість того, щоб перезапускати або повністю відкочувати їх. Автори провели експерименти для порівняння запропонованої методології з наявними підходами MVCC. Результати показали кращу тривалість виконання запропонованого методу порівняно з іншими. Хоча робота зосереджена на транзакціях з базами даних, основні принципи управління паралельними операціями та збереження цілісності даних застосовні і до моніторингу IoT. Пристрої IoT, як і транзакції в базах даних, часто працюють паралельно і потребують механізмів для забезпечення узгодженості даних.

У роботі [5] автори навели детальне дослідження механізму контролю паралелізму, розробленого для баз даних, що використовують енергонезалежну пам'ять (англ. *Non-Volatile Memory*, NVM). Дослідження заглиблюється в передові концепції узгодженості та паралелізму даних. У роботі наведено багатотактну ізоляцію знімків (англ. *Multi-Clock Snapshot Isolation*, MCSI) як механізм контролю паралелізму для однорівневих баз даних NVM. Вона має на меті покращити традиційні механізми, ефективно обробляючи згенеровані знімки і масив статусів транзакцій. Підхід MCSI може стати основою для розроблення стратегій вирішення конфліктів у системах IoT на підставі CRDT для вирішення конфліктів транзакцій та узгодженості даних. Методи, використані в MCSI для зменшення затримки транзакцій, особливо актуальні для оброблення даних у режимі реального часу.

У роботі [10] автори зосереджують увагу на вдосконаленому контролі паралелізму та відстеженні причинно-наслідкових зв'язків у розподілених системах з використанням векторного годинника з можливістю скидання (англ. *Resettable Encoded Vector Clock*, REVC). У роботі наведено REVC як реалізацію логічного годинника для вирішення проблеми високих темпів зростання і необмеженого зростання традиційних векторних годинників. Автори демонструють, як на підставі використання REVC можна вирішити проблему динамічного виявлення гонок у системах зі спільною пам'яттю, що робить його корисним джерелом для вирішення подібних проблем паралелізму в середовищах IoT.

Автори роботи [15] розглядають можливість застосування ключових технологій розподіленого зберігання даних на підставі IoT-систем у протипожежному захисті міст. Розглядувана проблема полягає в потребі моніторингу в режимі реального часу та віддаленого контролю різних протипожежних засобів у будівлях, що стало актуальним через збільшення частоти пожеж. До переваг такого впровадження можна віднести підвищення ефективності та результативності моніторингу та управління міськими системами протипожежного захисту завдяки використанню передових технологій, таких як IoT та хмарних обчислень.

У роботі автори роботи [3] навели платформу оброблення даних IoT, яка використовується для моніторингу та управління міською інфраструктурою. Мета такої платформи – забезпечити моніторинг поточного стану інфраструктури в режимі, близькому до реального часу, а також аналітику для географічно розподілених випадків використання. Платформа призначена для роботи в хмарі або локально в туманному (англ. *Fog*) середовищі з можливістю адаптації до будь-яких змін у робочому навантаженні з часом. Автори прагнули подати архітектуру, яка не є надто складною і складається зі зрілих програмних рішень з відкритим вихідним кодом.

Результати дослідження та їх обговорення / Research results and their discussion

Починаючи з розроблення CRDT-структур, дослідження фокусується на трьох основних типах структур: *VClock*, *MVReg* та *Map*. Кожна з цих структур вирішує специфічні завдання управління узгодженістю в розподілених системах, надаючи можливість ефективно обробляти конфлікти без потреби у блокуванні чи складних протоколах консенсусу.

Векторний годинник використовується для відстеження причинно-наслідкових залежностей між подіями в розподілених системах. Кожен векторний годинник є вектором цілих чисел, де кожен елемент відповідає локальному часу для конкретного вузла системи

$$VClock_i = \left[e_{i,j}, j = \overline{1, n_i} \right], i \in m \,, \qquad (1)$$

де: $e_{i,j}$ – локальна тривалість *j*-тої операції, виконана *i*-тим вузлом; m – загальна кількість вузлів у системі; n_i – загальна кількість операцій у *i*-му вузлі системи.

Формальний опис реалізованої структури *VClock*:

Нотація:

$VC \equiv$ Векторний годинник

$A \equiv$ Набір акторів, $A = \{a_1, a_2, \ldots, a_n\}$

$C \equiv$ Векторне відображення годинника, $C : A \to \mathcal{N}^0$

$Op \equiv$ Набір операцій {Inc, Merge, ResetRemote}

Визначення:

Векторним годинником *VC* називають кортеж (*A*, *C*), де: *A* скінченна множина акторів; *C* пов'язує кожного актора $a \in A$ з лічильником $c \in \mathcal{N}^0$

Процедури структури *VClock*:

процедура Increment(*VC*, *a*)

$VC[a] \leftarrow VC[a] + 1$

кінець процедури

процедура Merge(VC_1, VC_2)

$\forall a \in A, VC_1[a] \leftarrow \max(VC_1[a], VC_2[a])$

кінець процедури

процедура ResetRemove(*VC*, VC_{other})

$\forall a \in A$, if $VC[a] < VC_{other}[a]$, тоді видалити a з VC
кінець процедури

Порівняння та паралелізм:

Векторний годинник дає змогу порівнювати причинно-наслідкові зв'язки (2) та паралелізм (3), а саме:

$$VC_1 \le VC_2 \Leftrightarrow \forall a \in A,\ VC_1[a] \le VC_2[a], \quad (2)$$

$$VC_1 \parallel VC_2 \Leftrightarrow \exists VC_1 \le VC_2 \wedge \exists VC_2 \le VC_1. \quad (3)$$

Кожен актор (вузол, процес або компонент системи з унікальним ідентифікатором) має свій локальний векторний годинник, який ініціалізується нулями й оновлюється при кожній локальній операції, інкрементуючи лічильник відповідного актора.

Під час взаємодії між акторами, наприклад, під час відсилання повідомлення або синхронізації стану, векторний годинник відправника передається разом із даними. Отримувач порівнює та інтегрує отриманий годинник із своїм локальним.

Під час отримання векторного годинника від іншого актора, кожен актор об'єднує свій локальний годинник із отриманим, обираючи максимальне значення для кожного актора з обох годинників. Це гарантує, що локальний стан враховує інформацію про всі відомі до цього моменту події в системі.

Алгоритм дає змогу визначити, чи одна подія відбулась до іншої, чи події є незалежними. Це досягається шляхом порівняння векторних годинників: якщо кожен елемент одного годинника менший або рівний відповідному елементу в іншому, то можна вважати, що події відбулися одна до іншої.

Структура *MVReg* дає змогу зберігати декілька значень, що можуть виникати внаслідок одночасних записів у різних вузлах. Кожне значення асоціюється з векторним годинником, що дає змогу вирішувати конфлікти шляхом порівняння годинників. Структуру *MVReg* можна визначити як набір пар значення (*value*) та векторного годинника (*VClock*)

$$MVReg_k = \left\{(value_i, VClock_i), i = \overline{1, m_k}\right\}, k \in K\ , \quad (4)$$

де: ($value_i$, $VClock_i$) – кортеж, який містить значення та відповідний векторний годинник для i-го вузла; m_k – загальна кількість вузлів для k-го ключа; k – ідентифікатор ключа в їх наборі K.

Формальний опис реалізованої структури *MVReg*:

Нотація:

MVReg ≡ Багатозначний реєстр

V ≡ Набір значень, що зберігаються у *MVReg*

A ≡ Набір акторів у розподіленій системі

C ≡ Векторний годинник, де $C : A \to \mathcal{N}^0$

Op ≡ Операції *MVReg*

Визначення:

Структуру *MVReg* наведено у вигляді набору кортежів, кожен з яких $e \in MVReg$ це (C_e, v), де:

- C_e вектор, пов'язаний зі значенням v
- $v \in V$ є значенням

Процедури структури *MVReg*:

процедура *MVReg*Initialization
$MVReg \leftarrow \varnothing$
кінець процедури
процедура ApplyOperation(*MVReg*, *Op*)
if Op is Put(C, v) then
$MVReg \leftarrow MVReg \cup \{C, v\}$
end if
кінець процедури
процедура ResetRemove(*MVReg*, C)
$MVReg \leftarrow \{(C_e, v) \in MVReg \mid C_e\ C\}$
кінець процедури
процедура Read(*MVReg*)
return $\{v \mid (C_e, v) \in MVReg\}$
кінець процедури

Для додавання нового значення створюється операція Put з поточним контекстом. Ця операція впроваджується шляхом видалення всіх значень, які передують новому значенню (тобто тих, що повністю домінують його векторним годинником), та додавання нового значення до списку, якщо воно не домінується іншими.

Під час читання повертаються всі поточні конкурентні значення з реєстру, разом із їхніми контекстами, надаючи реалізувати додаткову логіку оброблення конфліктів на стороні клієнта.

Видаляються всі значення, контекст яких повністю передує заданому векторному годиннику, зменшуючи у такий спосіб кількість конкурентних значень у регістрі.

Структура *Map* дає змогу створювати пари ключ / значення, де кожне значення може бути незалежно оновлене. Це забезпечує високий рівень конкурентності та дає змогу уникнути втрати даних за конфліктів. *Map* можна подати як структуру, де кожен ключ асоціюється зі структурою *MVReg*, що дає змогу зберігати декілька версій для цього ключа, а саме:

$$Map = \left\{(key_k : MVReg_k), k = \overline{1, K}\right\}, \quad (5)$$

де K – загальна кількість ключів у структурі.

Формальний опис структури *Map*:

Нотація:

M ≡ CRDT *Map*

K ≡ Набір ключів в M

V ≡ Набір значень у M

A ≡ Набір акторів у розподіленій системі

C ≡ Векторний годинник, де $C : A \to \mathcal{N}^0$

Op ≡ Операції *Map*

Визначення:

Структуру *Map* M подають у вигляді кортежу (C_M, E) де:

- C_M – векторний годинник для структури *Map*
- E – це набір записів, кожен з яких $e \in E$ це кортеж (k, v, C_e)
- $k \in K$ є ключем
- $v \in V$ є значенням
- $C_e : A \to \mathcal{N}^{0-}$ векторний годинник, пов'язаний із ключем k та значенням v.

Процедури структури *Map*:

процедура MapInitialization
$Map \leftarrow (\varnothing, \varnothing)$
кінець процедури
процедура ApplyOperation(M, *Op*)
if *op* is *Up*(k, v, C) then
$E \leftarrow E \cup \{e \mid e = (k, *, *)\}$
else if *op* is *Rm*(k, C) then
$E \leftarrow E \setminus \{e \mid e = (k, *, *)\}$
end if
кінець процедури
процедура Merge(M_1, M_2)
for each $e_2 \in E_2$ do
if $\exists e_1 \in E_1 : e_1.k = e_2.k$ then
$C_{merged} \leftarrow$ merge C_{e1} and C_{e2}
Update e_1 in E_1 with C_{merged}

else
$E_1 \leftarrow E_1 \cup \{e_2\}$
end if
end for
$C_{M1} \leftarrow$ merge C_{M1} and C_{M2}
кінець процедури

Операції оновлення у структурі *Map* виконують через внесення змін до значень за певними ключами, з можливістю додавання нових ключів або модифікації наявних. Операції видалення використовують векторні годинники для забезпечення узгодженості, надаючи можливість видаляти ключі з урахуванням їхнього попереднього стану. Зчитування значень зі структури *Map* повертає поточний стан ключа разом з його контекстом, надаючи інформацію про версійність даних.

Для тестування обраного підходу до моніторингу було побудовано стенд для емуляції георозподіленої мережі. Стенд містить три плати відлагоджування STM32F407G-DISC1, які під'єднані до одного комп'ютера через інтерфейс UART (рис. 1).

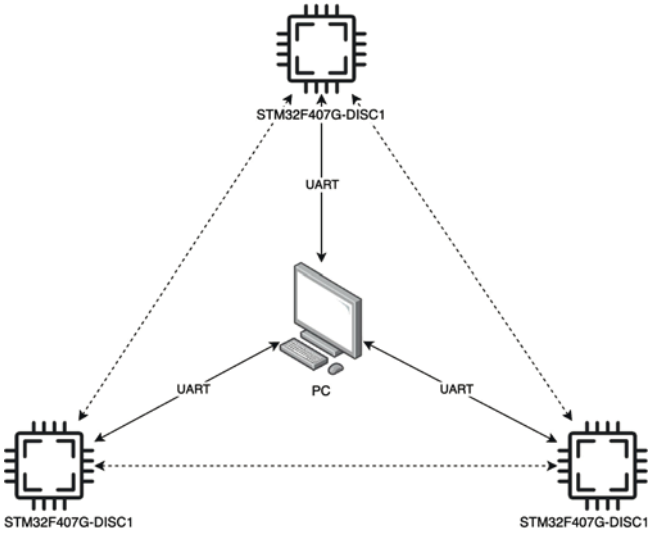

Рис. 1. Топологія тестового стенду емуляції георозподіленої мережі / Topology of the testbed for emulating a geo-distributed network [6]

У цьому стенді комп'ютер виконує роль наглядача, який проводить заміри метрик передачі даних. Також комп'ютер емулює сітчасту (англ. *Mash*) мережу між контролерами, що дає змогу в міру потреби змінювати параметри мережі.

Опис тестових сценаріїв для експериментального тестування:

Сценарій 1: Перевірка узгодженості даних з використанням структури *VClock*

Мета: Визначити ефективність структури *VClock* у забезпеченні узгодженості даних між вузлами під час одночасних змін.

Процедура:

1. Усі вузли одночасно генерують зміни в локальних даних;
2. Кожен вузол оновлює свій *VClock* при зміні та розповсюджує ці зміни разом із *VClock* на інші вузли;
3. При отриманні змін, вузли порівнюють структури *VClock* для визначення причино-наслідкових зв'язків;
4. Фіксація результатів синхронізації та виявлення конфліктів.

Очікувані результати: Усі вузли мають зберегти узгоджений стан даних після синхронізації, із мінімальною кількістю конфліктів.

Сценарій 2: Оброблення одночасних записів з використанням структури *MVReg*

Мета: Оцінити здатність структури *MVReg* ефективно розв'язувати конфлікти при одночасних записах.

Процедура:

1. Встановлення ініційованого значення в *MVReg* на всіх вузлах;
2. Симуляція одночасних записів різних значень від кожного вузла;
3. Кожен вузол відправляє свій запис разом із асоційованим *VClock* іншим вузлам;
4. При отриманні записів, вузли використовують логіку структури *MVReg* для вирішення конфліктів;
5. Збір та аналіз даних про кількість і типи розв'язаних конфліктів.

Очікувані результати: Вузли мають зберігати всі конкурентні записи з асоційованими структурами *VClock*, надаючи ручне або автоматичне розв'язання конфліктів.

Сценарій 3: Масштабування і відмовостійкість з використанням структури *Map*

Мета: Аналізувати ефективність структури *Map* у сценаріях масштабування та відмовостійкості.

Процедура:

1. Створення кількох ключ/значення пар у структурі *Map* на одному вузлі;
2. Розповсюдження *Map* структури між усіма вузлами;
3. Симуляція відмови одного вузла та оцінювання впливу на доступність та узгодженість даних.

Очікувані результати: Система здатна швидко відновитися після відмови, з мінімальним впливом на доступність та узгодженість даних.

На підставі описаних сценаріїв досліджено такі метрики системи:

- тривалість досягнення глобальної узгодженості;
- частка успішних синхронізацій;
- кількість вирішених конфліктів;
- середня тривалість вирішення конфлікту;
- час відновлення системи після відмови;
- вплив відмови на доступність даних.

Для кожного сценарію випадково генерується 10000 операцій. Оскільки оновлення коштують дорожче, ніж зчитування, частка операцій оновлення (зчитування) змінюється між 10(90), 25(75), 50(50), 75(25) та 90(10), щоб спостерігати за впливом робочих навантажень з різним співвідношенням оновлення/зчитування (рис. 2).

У Сценарії 1 кожен вузол після зміни свого внутрішнього стану розсилає оновлення іншим двом вузлам. Тривалість передачі одного повідомлення становить в середньому 3 мс. Однак, оскільки оновлення можуть бути відіслані паралельно, загальна тривалість передачі не буде прямо пропорційною кількості вузлів. Натомість, визначальним буде тривалість найповільнішої передачі. Припускаючи, що для гарантування узгодженості потрібен ще один раунд синхронізації між вузлами, щоб узгодити останні зміни:

Тривалість досягнення глобальної узгодженості становить $2 \cdot 3$ мс ≈ 6 мс. Однак на практиці можуть виникати затримки через оброблення даних, перевірку конфліктів і мережеві затримки, які впливають на кінцевий результат. Для оцінювання частки успішних синхронізацій, експерементально визначено, що у 2 % випадках виникають конфлікти, які не можуть бути автоматично вирішені без втручання алгоритму вирішення конфліктів, на підставі цього визначено, що 98 % усіх повідомлень успішно синхронізовано.

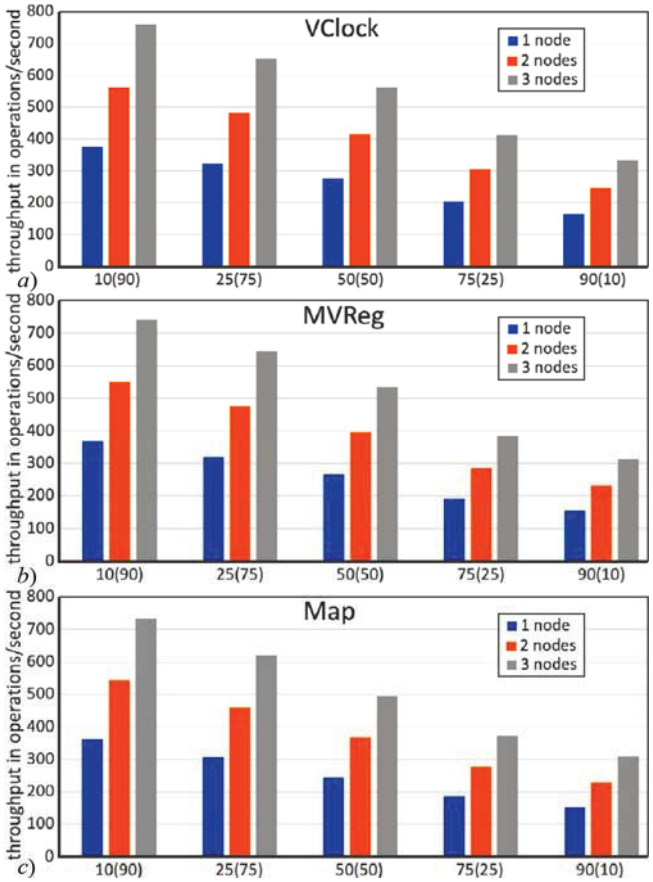

Рис. 2. Пропускна здатність CRDT-структур / Bandwidth of CRDT structures: a) *VClock*; b) *MVReg*; c) *Map*

Для Сценарію 2, враховуючи, що загальна кількість операцій на кожен вузол становить 10000, і співвідношення оновлення до читань змінюється від 10(90) до 90(10), кількість потенційних конфліктів збільшується зі зростанням частки оновлень. Для співвідношення 50(50), де 5000 з 10000 операцій на кожен вузол є оновленнями, загальна кількість оновлень для трьох вузлів становить 15000 · 2 % = 300.

Однак, враховуючи недетермінізовану природу розподіленої мережі, це значення може відрізнятись.

Середня тривалість вирішення конфлікту містить час на оброблення даних, виявлення конфлікту, його вирішення за допомогою алгоритму структури *MVReg* і синхронізацію оновленого стану з іншими вузлами. Враховуючи мережеві затримки та тривалість оброблення було визначено, що середня тривалість на вирішення одного конфлікту становить приблизно 10 мс.

Ця тривалість може змінюватися залежно від специфіки мережі, потужності обчислювальних ресурсів вузлів і ефективності алгоритму структури *MVReg*.

У Сценарії 3 аналізується ефективність структури *Map* у контексті масштабування та відмовостійкості. Він зосереджений на оцінюванні того, як система впорається з відмовами окремих вузлів і як це вплине на доступність та узгодженість даних.

Після симуляції відмови одного вузла, система використовує реплікацію даних між робочими вузлами для відновлення втраченої інформації. Враховуючи мережеві затримки, тривалість оброблення та синхронізацію даних, загальний час відновлення в середньому становить 24 мс.

Якщо перед відмовою всі запити на оновлення були успішно оброблені, а після відмови і запуску механізмів відновлення, система змогла обробити 98 % запитів (з урахуванням того, що деякі дані могли бути тимчасово недоступні або потребувати часу на відновлення), то вплив відмови на доступність (метрика більш відома як Service Level Objective) розраховується за формулою

$$P = \frac{m}{M} \cdot 100\ , \%, \qquad (6)$$

де: P – вплив відмови на доступність; m – кількість успішно оброблених запитів зчитування після відмови; M – загальна кількість запитів.

У разі співвідношення операцій 50(50) вплив відновлення на доступність даних дорівнює 98 %. Це означає, що впродовж 98 % загального часу спостереження після відмови, дані в системі залишалися доступними для користувачів або процесів. Іншими словами, тільки впродовж 2 % від загального часу спостереження дані були недоступні через процеси відновлення, реплікації або переконфігурації системи.

Статично виділена пам'ять, яка використовується для реалізованих структур, залежить від кількості акторів у розподіленої мережі (таблиця).

Таблиця. Використання статично виділеної пам'яті для ініціалізації структур / Statically allocated memory for initializing structures

Структура	Розмір для 2-х акторів, байт	Розмір для 4-х акторів, байт	Розмір для 8-ми акторів, байт	Розмір для 16-ти акторів, байт
Map	592	2664	15160	100440
MVReg	120	392	1416	5384
VClock	48	88	168	328

Ключовою особливістю у роботі з CRDT є те, що ці структури дають змогу системі толерувати випадки втрати даних під час передачі, адже кожен вузол зберігає достатньо інформації для відновлення узгодженого стану без потреби постійної синхронізації з іншими вузлами.

Отже, у системі, де кожен давач змінює тільки свої дані, основними сценаріями конфліктів є розбіжності, пов'язані з причинно-наслідковими залежностями та обробленням застарілих даних. CRDT ефективно вирішують ці проблеми, забезпечуючи високий рівень узгодженості. Важливо зазначити, що хоча CRDT ефективно вирішують проблеми узгодженості та доступності для децентралізованих систем, вони можуть вимагати більше ресурсів для зберігання даних (наприклад, зберігання декількох версій даних у структурі *MVReg*) та обчислень (наприклад, об'єднання станів у *Map* структурі).

Обговорення результатів дослідження. У роботі [1] наведено протокол "Cure" – новий підхід до підтримки причинно-наслідкової узгодженості в георепліkованих, розподілених системах. Протокол Cure розроблений для забезпечення балансу між високою узгодженістю даних, високою доступністю та низькою затримкою в розподілених системах баз даних. Це досягається завдяки використанню комбінації CRDT і векторних годинників для управління реплікацією даних між декількома центрами оброблення даних. Цей протокол зберігає кілька версій кожного об'єкта, що дає змогу обслуговувати запити з причинно-наслідкових знімків, і використовує векторний годинник для кодування причинно-наслідкових залежностей між транзакціями. Стаття також містить обширну експериментальну оцінку, яка порівнює продуктивність Cure з тради-

ційними системами за різних сценаріїв робочого навантаження.

Робота авторів [4] присвячена проблемі у сфері розподілених систем, зокрема, причинно-наслідковому доступу в георепліковані системах зберігання даних. Автори пропонують багатокритеріальну оптимізаційну задачу і представляють набір рішень, в т.ч. різні стратегії зчитування, які мають на меті збалансувати затримку і вартість. Дослідження включає емпіричну оцінку запропонованих стратегій з використанням реальних хмарних сервісів зберігання даних, таких як AWS S3, DynamoDB та MongoDB. Результати демонструють покращення латентності системи.

Проблема, яку вирішили автори роботи [2], полягала в обмеженнях застарілої системи моніторингу мережі. Рішенням стала "REDEMON", відмовостійка децентралізована система моніторингу, розміщена на розподілених і взаємопов'язаних периферійних пристроях. Система використовує структури даних на підставі CRDT, реалізовані в AntidoteDB, для забезпечення надійного, узгодженого моніторингу мережі. До переваг цієї реалізації автори відносять відмовостійкість, децентралізацію та автоматизацію. До недоліків – значне використання процесора та оперативної пам'яті системою моніторингу на малопотужних пристроях та значний мережевий трафік, необхідний для досягнення відмовостійкого та узгодженого рівня зберігання даних.

Автори роботи [9] пропонують нову архітектуру для розподіленої координації у туманних платформах. Архітектура основана на використанні CRDT для забезпечення кінцевої узгодженості. Автори аргументують це тим, що використання CRDT для управління даними конфігурації дає змогу вузлам вносити зміни без узгодження з іншими вузлами системи. Потенційно, реалізація запропонованої архітектури зменшить затримку запитів і збереже систему доступною у разі нестабільної мережі.

Аналіз досліджень дає змогу зрозуміти, що використання CRDT-структур позитивно впливає на підтримку причинно-наслідкової узгодженості, але у контексті застосування таких структур у системах IoT, на підставі отриманих результатів дослідження, рекомендується обмежити використання CRDT-структур до специфічних застосувань, де переваги масштабованості та стійкості до мережевих перебоїв переважають недоліки, пов'язані з високим споживанням ресурсів і складності впровадження. Для IoT-систем, що працюють у режимі реального часу і мають обмеження за обсягом пам'яті та енергоспоживанням, важливо знайти баланс між потребою у високій доступності даних та необхідністю ефективного використання обмежених ресурсів.

Отже, за результатами виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – набула подальшого розвитку модель застосування конфлікт-вільних реплікаційних структур даних в обмежених середовищах виконання.

Практична значущість результатів дослідження – розроблено CRDT-структури із використанням статично виділеної пам'яті, досліджено метрики ефективності застосування таких структур; результати дослідження можна використовувати під час інтеграції CRDT-структур в обмежених середовищах виконання.

Висновки / Conclusions

Дослідження ефективності використання CRDT-структур у системах моніторингу IoT виявило як потенційні переваги, так і значні обмеження. CRDT-структури пропонують інноваційний підхід до управління даними, який підвищує масштабованість та витривалість до мережевих перебоїв, але їх практичне застосування в IoT обмежене через високі вимоги до обчислювальних ресурсів. За результатами дослідження можна зробити такі основні висновки:

1. Запропоновано новий підхід застосування CRDT-структур у контексті систем IoT для забезпечення причинно-наслідкової узгодженості, що дасть змогу зменшити кількість помилок синхронізації, спричинених збоями у мережі, та підвищити загальну стійкість систем до несподіваних відмов.
2. Розроблено CRDT-структури із використанням статично виділеної пам'яті, що дасть змогу оцінити стабільність і продуктивність структур в умовах обмежених ресурсів.
3. Розроблено тестовий стенд для емуляції георозподіленої мережі, що уможливить детальне вивчення поведінки CRDT-технології у різних мережевих умовах, що імітують реальні операційні сценарії в IoT-системах.
4. Проведено експеримент на підставі трьох тестових сценаріїв для дослідження основних метрик системи, що забезпечить можливість отримати цінну інформацію про поведінку системи під час реальних операційних умов, а також оцінити вплив технології CRDT на продуктивність та надійність системи.
5. На підставі отриманих результатів виявлено, що хоча CRDT-структури надають обнадійливі перспективи для покращення масштабованості, надійності та доступності даних у розподілених системах, їх використання в контексті IoT вимагає додаткових уточнень та інновацій для подолання наявних обмежень.

References

1. Akkoorath, D. D., Tomsic, A. Z., Bravo, M., Li, Z., Crain, T., Bieniusa, A., Preguiça, N., & Shapiro, M. (2016). Cure: Strong Semantics Meets High Availability and Low Latency. *36th International Conference on Distributed Computing Systems* (ICDCS), 405–414. https://doi.org/10.1109/ICDCS.2016.98
2. Centelles, R. P., Selimi, M., Freitag, F., & Navarro, L. (2020). REDEMON: Resilient Decentralized Monitoring System for Edge Infrastructures. *20th ACM International Symposium on Cluster, Cloud and Internet Computing* (*CCGRID*), 91–100. https://doi.org/10.1109/CCGrid49817.2020.00-84
3. Geldenhuys, M. K., Will, J., Pfister, B. J. J., Haug, M., Scharmann, A., & Thamsen, L. (2021). Dependable IoT Data Stream Processing for Monitoring and Control of Urban Infrastructures. *International Conference on Cloud Engineering* (IC2E), 244–250. https://doi.org/10.1109/IC2E52221.2021.00041
4. Lima, S., Araujo, F., Guerreiro, M. de O., Correia, J., Bento, A., & Barbosa, R. (2023). Efficient Causal Access in Geo-Replicated Storage Systems. *Journal of Grid Computing*, *21*(1), 8. https://doi.org/10.1007/s10723-022-09640-z
5. Liu, X., Chen, K., Liu, M., Cai, S., Wu, Y., & Zheng, W. (2022). Multi-Clock Snapshot Isolation Concurrency Control for NVM Database. *Tsinghua Science and Technology*, *27*(6), 925–938. https://doi.org/10.26599/TST.2021.9010036
6. Munoz, M., Guzman, J. L., Torres, M., & Acien, F. G. (2022). An IoT Platform for Data Management in an Industrial-Scale Microalgae Cultivation Plant, 10, 127128–127139. https://doi.org/10.1109/ACCESS.2022.3226334
7. Naas, M. I., Lemarchand, L., Raipin, P., & Boukhobza, J. (2021). IoT Data Replication and Consistency Management in Fog Computing. *Journal of Grid Computing*, *19*(3), 33. https://doi.org/10.1007/s10723-021-09571-1

8. Nobrega, L., Tavares, A., Cardoso, A., & Goncalves, P. (2018). Animal monitoring based on IoT technologies. 2018 IoT *Vertical and Topical Summit on Agriculture – Tuscany* (IOT Tuscany), 1–5. https://doi.org/10.1109/IOT-TUSCANY.2018.8373045
9. Pfandzelter, T., Schirmer, T., & Bermbach, D. (2022). Towards Distributed Coordination for Fog Platforms. 22nd International Symposium on Cluster. *Cloud and Internet Computing* (CCGrid), 760–762. https://doi.org/10.1109/CCGrid54584.2022.00087
10. Pozzetti, T., & Kshemkalyani, A. D. (2021). Resettable Encoded Vector Clock for Causality Analysis With an Application to Dynamic Race Detection. *Transactions on Parallel and Distributed Systems*, 32(4), 772–785. https://doi.org/10.1109/TPDS.2020.3032293
11. Putra, A. S., & Warnars, H. L. H. S. (2018). Intelligent Traffic Monitoring System (ITMS) for Smart City Based on IoT Monitoring. *Indonesian Association for Pattern Recognition International Conference* (INAPR), 161–165. https://doi.org/10.1109/INAPR.2018.8626855
12. Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict-Free Replicated Data Types. B X. Défago, F. Petit, & V. Villain (Ed.), Stabilization, Safety, and Security of Distributed Systems, 6976, 386–400. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24550-3_29
13. Vedaei, S. S., Fotovvat, A., Mohebbian, M. R., Rahman, G. M. E., Wahid, K. A., Babyn, P., Marateb, H. R., Mansourian, M., & Sami, R. (2020). COVID-SAFE: An IoT-Based System for Automated Health Monitoring and Surveillance in Post-Pandemic Life. *In IEEE Access*, Vol. 8, 188538–188551. https://doi.org/10.1109/ACCESS.2020.3030194
14. Vikram, P., Abdul Moiz, S., & G R, A. (2019). Multiversion Concurrency Control with the Precedence Graph Generation Algorithm. *International Journal of Innovative Technology and Exploring Engineering*, 8(9S2), 95–100. https://doi.org/10.35940/ijitee.I1019.0789S219
15. Yin, W., Han, Z., Feng, Y., Zhou, G., & Shen, Y. (2022). Application of Key Technologies of Distributed Storage Based on the Internet of Things in Urban Fire Protection. *Scientific Programming*, 1–9. https://doi.org/10.1155/2022/9381139

A. V. Fechan, D. P. Kushniruk

Lviv Polytechnic National University, Lviv, Ukraine

EVALUATION OF THE EFFECTIVENESS OF CRDT TECHNOLOGIES IN IOT MONITORING SYSTEMS

The effectiveness of using Conflict-free Replicated Data Types (CRDTs) to ensure data consistency and coherence in the Internet of Things (IoT) monitoring systems is evaluated. The challenges associated with monitoring a large number of distributed IoT devices require efficient data management, hence this study considers CRDT technology as an alternative that can simplify this process. CRDT structures are distributed data types that provide Strong Eventual Consistency (SEC) and have such properties as commutativity and idempotence. As part of the research, such CRDT structures as a vector clock (*VClock*), a multi-valued register (*MVReg*), and a replication map (*Map*) were implemented and analyzed, which made it possible to assess their impact on the efficiency of data synchronization and overall system performance. A testbed with three IoT devices was created to emulate a geo-distributed network, which allowed for an experimental study, the results of which indicate certain advantages of using CRDT technology, especially in terms of scalability and system resilience to network failures. Metrics such as the duration of achieving global consistency, the proportion of successful synchronizations, the number of resolved conflicts, the average conflict resolution time, the system recovery time after a failure, and the impact of a failure on data availability are considered. It was found that the use of CRDT technology helps increase the efficiency of data replication and synchronization processes, minimizing the impact of network delays and loss of information in the event of device or network outages. However, it is worth noting some limitations associated with the use of CRDT technology, in particular, increased energy consumption and relatively high requirements for hardware computing resources. The utilization of statically allocated memory is determined to lead to an increase in the complexity of using such structures, specifically, the ability to dynamically change the number of nodes in the system is limited. Such limitations require further research and optimization of the architecture of IoT systems using CRDT technology to ensure their greater energy efficiency and reduce hardware requirements.

Keywords: data synchronization; replicated data structures; system scalability; disaster recovery.