

Міністерство освіти і науки України
Харківський національний університет
імені В.Н. Каразіна

Інститут проблем машинобудування
імені А.М. Підгорного НАН України

Кваліфікаційна наукова
праця на правах рукопису

Жолткевич Галина Григоріївна

УДК 004.942

ДИСЕРТАЦІЯ
МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ РЕПЛІКАЦІЇ ДАНИХ У
РОЗПОДІЛЕНИХ СХОВИЩАХ

Спеціальність 01.05.02 – математичне моделювання та обчислювальні
методи (технічні науки)

Подається на здобуття наукового ступеня кандидата наук

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело

 Г.Г. Жолткевич

Науковий керівник: Руккас Кирило Маркович, доктор технічних наук,
професор

Харків-2021

АНОТАЦІЯ

Жолткевич Г.Г. Моделювання процесів реплікації даних у розподілених сховищах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 01.05.02 – математичне моделювання та обчислювальні методи (технічні науки). – Харківський Національний Університет імені В.Н. Каразіна; Інститут проблем машинобудування ім. А.М. Підгорного НАН України, Харків. – 2021.

У дисертаційній роботі вирішується актуальна науково-практична задача розробки загального методу прийняття рішень, на який можливо спиратись при побудові мережі для розподіленого сховища даних (РСД) з метою запобігання певних ризиків некоректної поведінки систем при розподілених обчисленнях, обробці запитів у розподіленому середовищі.

Це дозволяє уникати в процесі проєктування постійного врахування ліміту запитів, доступної кількості кластерів та їх розташування, а натомість мати конкретні критерії для формування сховища за умов різноманітних потреб конкретних розподілених систем з врахуванням обмежень, встановлених CAP-теоремою.

РСД мають три основні критерії надійності - гарантії, які подані у CAP-теоремі: узгодженість, доступність та стійкість до розділень. CAP-теорема наголошує, що у будь-якій розподіленій системі можливо задовольнити тільки дві з трьох таких гарантій одночасно.

Отже, для досягнення поставленої мети в роботі отримано результати, що описано нижче.

Побудовано модель логічних годинників для РСД, що доводить однозначне визначення порядку версій реплік у будь-якому разі, навіть якщо репліки надійшли одночасно на різні вузли мережі. Доведення виведених

тверджень в цій моделі базуються на теорії множин та моделі логічних годинників відомого математика Леслі Лампорта.

Така модель запобігає утворенню численних конфліктів між репліками, які надходять у мережу розподіленого сховища і розповсюджуються паралельно, та дає можливість прийняти однозначне рішення: яка репліка повинна бути записана у той момент, коли більш старша репліка може перезаписати нову і порушити узгодженість у сховищі.

Побудовано загальну модель розподіленого сховища, що дає уяву про основні структурні компоненти будь-якого РСД. Побудовано теоретичний базис для розвитку методів забезпечення CAP-гарантій: моделі мережі сховища описують характеристики, які впливають на коректну роботу та обробку даних в РСД, моделі централізованого та децентралізованого сховища даних описують структуру даних для коректного управління ними, подано опис базового методу реплікації для демонстрації основного процесу в РСД, який безпосередньо впливає на забезпечення рівней CAP-гарантій.

На базі побудованих моделей сформовані числові характеристики (метрики) для оцінки рівня CAP-гарантій у РСД, що надають можливість вдосконалення систем моніторингу розподілених сховищ даних і доповнюють теоретичний базис для розвитку методів обчислення та забезпечення CAP-гарантій. Такі числові характеристики дозволяють оцінити рівень CAP-гарантій стохастичним та кількісним шляхом, що дає контроль над системою РСД і розуміння, наскільки задовольняється та чи інша характеристика з трійки CAP.

Розвинуто методи обчислення рівня узгодженості з використанням сформованих числових характеристик рівней CAP-гарантій та теорії ймовірності. Це також удосконалює систему моніторингу розподіленого сховища, впроваджуючи нову метрику – рівень узгодженості даних.

На базі побудованих моделей подано метод обчислення ціни відновлення узгодженості, тобто ціни всього процесу реплікації, що дає підставу для прийняття більш зваженого рішення на етапі побудови мережі

РСД та проектування РСД. Це надає можливість відрегулювати та перерозподілити ресурси у мережі та запобігає певних ризиків у подальшій роботі РСД та його підтримки.

Базуючись на механізмах балансування навантаження та вже побудованих моделях для розподіленого сховища даних, у дисертаційній роботі подано метод контрольованого балансування узгодженості. Основна ідея цього методу – зберегти CAP-гарантії на належному рівні залежно від конкретних потреб. Цей метод пропонує на етапі реплікації зберігати у певній частині мережі відображення, де кожній одиниці даних буде відповідати список вузлів, що мають одну й ту саму репліку, яка буде найновішою, що гарантується розвиненим базисом у моделі логічних годинників. Підтримка такого відображення не повинно бути дуже складним, щоб бути ефективним у великій мережі РСД або частині мережі. За наявності такого знання кожний вузол, або центральний вузол-шлюз, на який надійшов запит на отримання даних, зможе виконати певні програмні дії, щоб направити запит на вузол, що має вже найновішу репліку. Щоб не відбулося ефекту вузького міста (bottleneck), кількість таких вузлів у такому відображенні, має бути достатньою, щоб зберегти доступність (друга гарантія CAP). Залежно від пріоритету потреби узгодженості над доступністю чи навпаки, встановлюється певний числовий поріг – кількість узгоджених вузлів, яка має бути доступною завжди, і не можна вказаним способом навантажувати ці вузли. Якщо поточна кількість менше за поріг, то система автоматично вибере доступність як пріоритет і не буде відправляти запит на один з таких вузлів, а буде діяти згідно старій схемі відповіді на запит отримання репліки.

Такий алгоритм дій можливий у досить великій мережі у сотні та тисячі вузлів за умови, що знання про узгоджені вузли зберігається у сховищі, що має достатньо ресурсів, аби відповідати на запит на оновлення та отримання невеликої кількості даних. Таке сховище може бути побудоване як невеликий кластер вузлів у середині мережі РСД, або як окрема централізована структура. Важливо підкреслити, що у другому випадку система перестане

буде повністю децентралізованою. У першому ж випадку деякі вузли системи РСД просто виконують інші ролі, аніж зберігання реплік.

Таким чином, поданий метод дозволяє балансувати між гарантіями доступності та узгодженості, що, залежно від стану РСД, буде відповідати певним потребам системи.

За результатами проведених досліджень і поданих методів обчислення та забезпечення CAP-гарантій сформовано практичні рекомендації у вигляді покрокового алгоритму дій на етапі проектування будь-якого програмного забезпечення, що потребує зберігання даних у великій розподіленій мережі. Оскільки через неправильне проектування системи РСД його підтримка, вдосконалення і розробка з часом стає тільки дорожчою, такий алгоритм дій надає можливість запобігти певних ризиків некоректного функціонування системи РСД.

Доведення всіх тверджень в побудованих моделях та обчислювальних методах є математично вірними. Методи, що дають практичні результати для оцінки і забезпечення рівней CAP-гарантій, доведені теоретично та практично шляхом експериментальних досліджень. Експерименти проводились шляхом розробки програмного коду згідно з поданими імітаційними моделями та запуску симуляцій відповідних процесів РСД та обчислення рівней CAP-гарантій у симуляції РСД.

Результати експериментів шляхом запуску розробленого програмного коду демонструють вірність результатів, отриманих теоретично, коректність обчислених за поданими у дисертації формул рівня узгодженості та ціни процесу реплікації, ефективність підходу у методі контрольованого балансування узгодженості. Також запуск таких експериментів дозволяє оцінити часову та просторову складність впровадженого методу.

Новизна роботи полягає в наступному.

- зазнала подальшого розвитку стохастична модель розподіленої системи за рахунок забезпечення формалізації виконання CAP-гарантій, яка, на відміну від існуючих моделей, об'єднує формалізовані визначення для

САР-гарантій в одну цілісну структуру і дає підставу для оцінки гарантій на іншому рівні;

- вперше запропоновано систему статистичних метрик для оцінювання виконання САР-гарантій у РСД, що базується на формалізації визначень САР-гарантій, і замість бінарної оцінки гарантій РСД пропонується оцінювати систему гарантій за допомогою ймовірнісних величин, що дає підставу для розвитку метрик для оцінки рівня САР-гарантій;

- удосконалено методи обчислення ймовірностей втрати даних на основі запропонованої моделі, що дає можливість оцінки САР-гарантій і формування метрик для оцінки з більшою точністю, з врахуванням реальних умов таких систем, і, таким чином, це дає підставу для формування методу регулювання САР-характеристик на рівні, потрібному конкретній розподіленій системі;

- зазнали подальшого розвитку методи забезпечення САР-вимог шляхом розробки нової стратегії контрольованого балансування узгодженості у формі спеціального методу контрольованого балансування узгодженості, що може бути інтегрований в методи реплікації у РСД.

Ключові слова: розподілене сховище даних, САР-теорема, математичні моделі, стохастичні метрики, метод балансування, алгоритм прийняття рішень.

Список публікацій здобувача

[1] Zholtkevych G., Mallet F., Zaretska I., Zholtkevych G. Two Semantic Models for Clock Relations in the Clock Constraint Specification Language. Communications in Computer and Information Science. 2013.Vol. 1000, P. 190–209.

[2] Zholtkevych, G. Метрики для обчислення узгодженості у розподілених сховищах даних. Сучасний стан наукових досліджень та технологій в промисловості. 2020. № 2(12). С. 40–48.

[3] Rukkas, K., Zholtkevych, G. Стохастична модель для оцінки CAP-гарантій для розподілених баз даних. Сучасні інформаційні системи. 2020. № 4 (2). С. 47–50.

[4] Rukkas, K., Zholtkevych, G. Балансування узгодженості у розподіленому сховищі даних. Системи управління, навігації та зв'язку. Збірник наукових праць, 2020. №. 2 (60). С. 95–100.

[5] Zaretska, I., Zholtkevych, G., Zholtkevych, G., Mallet, F. Clocks Model for Specification and Analysis of Timing in Real-Time Embedded Systems . 9th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 9–22 June 2013. Vol. 1356, P. 475–489.

[6] Rukkas, K., Zholtkevych, G. Distributed Datastores: Towards Probabilistic Approach for Estimation of Dependability. 11th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 14–16 May 2015. Vol. 1356, P. 523–534.

[7] Rukkas, K., Zholtkevych, G. Distributed Datastores: Consistency Metric for Distributed Datastores. XV Міжнародна науково-практична конференція «Теоретичні та прикладні аспекти побудови програмних систем» ТАAPSD'2017, 4–8 грудня 2017. С. 89–100.

[8] Zholtkevych, G. PROBABILISTIC MODEL FOR COMPUTING TIME DELIVERY IN DISTRIBUTED DATASTORE NETWORK. VI міжнародна науково-технічна конференція «Комп'ютерне моделювання та оптимізація складних систем», 4–6 листопада 2020 р. С. 33–35.

ABSTRACT

Zholtkevych G. Modeling data replication processes in distributed datastores.
– Qualification scientific work with the manuscript copyright.

Thesis for candidate of technical science on the specialty 01.05.02 – Mathematical Modeling and computing methods (technical science). – Karazina Kharkiv National University; A. Pidhorny Institute of Mechanical Engineering problems of NASU; Kharkiv. – 2021.

Actual scientific-applied problem has been posed in the thesis – the problem of existence of general decision-making method, that would be the one of the base components on the stage of distributed datastore design for preventing set of risks incorrect and unexpected behavior of distributed computations, requests processing in distributed datastore.

This allows to make larger size of distributed network and not to think permanently about possible clusters and nodes allocation, but to have specific criteria for building distributed datastore in the conditions of various system requirements and in the condition of the CAP-theorem existing.

Distributed datastore have three main reliability criteria – guarantees that are described in the CAP-theorem: consistency, availability and partition tolerance. CAP-theorem states, that in any distributed system it is possible to fulfill only two of three guarantees at the same time.

So, to solve posed problem results, that are described below, were obtained.

The logical clock model for distributed datastore is built, that proves the ability of unambiguous ordering of replica's versions in any case, even if replicas came to different network nodes at the same time. The proof of derived statements in this model is based on the theory of sets and model of logical clock, which mathematician Leslie Lamport is known by.

Such a model prevents occurrences of lots of conflicts between replicas, that come to a distributed network to different network parts and distributed in parallel. The model gives the possibility to make explicit decision, which replica should be

written to the database at the moment, when two replicas compete on the right to be established in the database and there is a risk to break consistency in the datastore.

The general model of distributed datastore is built. It gives the perspective of components structure in any distributed datastore. The theoretical basis for evolving methods of CAP-guarantees assurance is formed: the model of distributed network describes characteristics, that impacts on correct operation of distributed datastore; the models of centralized and decentralized datastore describe data structures to manage them in a proper way, basic replication method is described to demonstrate the main process in distributed datastore that impacts directly on the CAP-guarantees level.

The numerical characteristics (metrics) are formed on the basis of built models for evaluating the level of CAP-guarantees. That gives the possibility to improve monitoring systems of distributed datastores and supplement the theoretical basis for evolving methods of CAP-guarantees computation and assurance. Such numerical characteristics allow to evaluate the CAP-guarantees level by means of stochastic and quantitative approach. That gives a control over the distributed datastore system and understanding the degree of fulfilling one or another characteristic from CAP threesome.

The methods of consistency level computation are evolved using formed numerical characteristics of CAP-guarantees levels and probability theory. This also improves the monitoring system of distributed datastore, introducing new metric – the level of data consistency.

The method to calculate the cost of consistency renovation on the basic of built models, i.e., the average cost of whole replication process, that affords ground for making weighted decisions on the stage of building distributed network and distributed datastore design. This makes it possible to adjust the number of resources and distribute them in more reasonable way to prevent certain risks in further distributed system operation and its maintenance.

This dissertation presents the method of controlled consistency balancing on the base of load balancing mechanisms and already built models for distributed

datastore. The main concept of the method is to keep CAP-guarantees at the appropriate level depending on system specific needs. This method proposes to store on the replication stage the mapping in a certain part of distributed network. This mapping is key-value storage where the list of nodes, that have the same replica (which is the newest, that is guaranteed by evolved basic in logical clocks model), can be obtained by the dataunit identifier. They are called consistent nodes. The maintenance of such a mapping should not be too complex to keep the efficiency in a large distributed network or part of distributed network. With this knowledge, each node, or central gateway node, that has received a user request for retrieving replica, could execute certain program actions to send a request to a node that already has the latest replica. To avoid the bottleneck effect, the number of such consistent nodes must be sufficient to keep the availability (second CAP-guarantee) at least on the minimal required level. Depending on the priority consistency over availability, or opposite, the certain numerical threshold is set configuring the system – the number of consistent nodes that must be available while distributed datastore is functioning. If the current number of consistent nodes is less than set threshold, the system will automatically choose the availability as the priority and will not forward requests to one of such nodes, preventing overload, but will operate according to default algorithm to return the requested replica.

Such an algorithm is possible in quite large network of hundreds or thousands of nodes, provided that the knowledge of such consistent nodes is stored in the repository (or any other storage) that have sufficient number of resources to respond to a request of mapping modification or obtaining small amount of data. Such a storage can be built as a small cluster of nodes in the middle of the distributed network, or as a separate centralized structure. It is important to emphasize that in the second case the system will not be completely decentralized. In the first case, some nodes of the distributed datastore simply perform other roles than storing replicas, so system still remain decentralized.

Thus, the presented method allows to balance between guarantees of availability and consistency, which will cover specific needs, depending on the state

of the distributed datastore system.

Based on the research results and methods of calculation and assurance of CAP-guarantees the practical recommendations are introduced in the form of the step-by-step algorithm of actions on the designing stage of any software, that requires storing data in large distributed network. Because of the incorrect design of distributed datastore system, its maintenance, improvement and development only become more expensive over time. Such an algorithm gives the ability to prevent certain risks of incorrect operating of distributed datastore in the future.

Proofs of all statements in the constructed models and computational methods are mathematically correct. Methods, that give practical results for evaluating and ensuring the levels of CAP-guarantees, proven theoretically and practically by experimental studies. The experiments were performed by developing software code in accordance with the provided program simulation models and running simulations of the corresponding processes of distributed datastore, and calculating the levels of CAP-guarantees in the datastore simulation.

The results of experiments by running the developed program code demonstrate following: the accuracy of the results obtained theoretically; the correctness of the formulas of the consistency level and the replication process cost calculated in the dissertation; the effectiveness of the approach in the method of controlled consistency balancing. Also, the running of such experiments allows to evaluate the temporal and spatial complexity of the implemented method of controlled consistency balancing.

The novelty of the dissertation is as follows:

- the stochastic model of the distributed datastore was further developed by formalizing the CAP-guarantees assurance, which, unlike existing models, combines formalized definitions for CAP-guarantees into one integral structure and provides a basis for evaluating guarantees from another perspective;
- for the first time a system of statistical metrics for evaluating the CAP-guarantees assurance is proposed, based on the formalization of definitions of CAP-guarantees, and instead of binary evaluation it is proposed to evaluate the guarantees

using probabilistic values, which affords grounds for evolving metrics to evaluate guarantees;

- improved methods for calculating the probability of data loss based on the proposed model, which allows to evaluate CAP-guarantees and to form evaluation metrics with greater accuracy, taking into account the real conditions of distributed systems, and, thus, provides a basis for forming a method to adjust CAP-characteristics at the level required by a particular distributed system;

- the methods of ensuring CAP-requirements have been further developed by implementing a new strategy of controlled balancing of consistency in the form of a special method of controlled consistency balancing, which can be integrated in the methods of replication in request processing of distributed datastore.

Keywords: distributed datastore, CAP-theorem, mathematical models, stochastic metrics, balancing method, decision-making algorithm.

List of publications of the applicant

[1] Zholtkevych G., Mallet F., Zaretska I., Zholtkevych G. Two Semantic Models for Clock Relations in the Clock Constraint Specification Language. Communications in Computer and Information Science. 2013. Vol. 1000, P. 190–209.

[2] Zholtkevych, G. Метрики для обчислення узгодженості у розподілених сховищах даних. Сучасний стан наукових досліджень та технологій в промисловості. 2020. № 2(12). С. 40–48.

[3] Rukkas, K., Zholtkevych, G. Стохастична модель для оцінки CAP-гарантій для розподілених баз даних. Сучасні інформаційні системи. 2020. №. 4 (2). С. 47–50.

[4] Rukkas, K., Zholtkevych, G. Балансування узгодженості у розподіленому сховищі даних. Системи управління, навігації та зв'язку. Збірник наукових праць, 2020. №. 2 (60). С. 95–100.

[5] Zaretska, I., Zholtkevych, G., Zholtkevych, G., Mallet, F. Clocks Model for Specification and Analysis of Timing in Real-Time Embedded Systems . 9th

International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 9–22 June 2013. Vol. 1356, P. 475–489.

[6] Rukkas, K., Zholtkevych, G. Distributed Datastores: Towards Probabilistic Approach for Estimation of Dependability. 11th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 14–16 May 2015. Vol. 1356, P. 523–534.

[7] Rukkas, K., Zholtkevych, G. Distributed Datastores: Consistency Metric for Distributed Datastores. XV Міжнародна науково-практична конференція «Теоретичні та прикладні аспекти побудови програмних систем» ТАAPSD'2017, 4–8 грудня 2017. С. 89–100.

[8] Zholtkevych, G. PROBABILISTIC MODEL FOR COMPUTING TIME DELIVERY IN DISTRIBUTED DATASTORE NETWORK. VI міжнародна науково-технічна конференція «Комп'ютерне моделювання та оптимізація складних систем», 4–6 листопада 2020 р. С. 33–35.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП	5
РОЗДІЛ 1 АНАЛІЗ СУЧАСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПОДІЛЕНИХ СХОВИЩ ДАНИХ.....	10
1.1 Централізовані і децентралізовані сховища даних і їх властивості.	10
1.2 Аналіз сучасних досліджень і актуальність досліджень РСД.....	15
1.3 Порівняльна характеристика реалізацій РСД з відкритим кодом	16
1.4 Маршрутизація запитів у розподіленій системі даних	24
1.5 Висновки за розділом	29
РОЗДІЛ 2 МАТЕМАТИЧНІ МОДЕЛІ В РОЗПОДІЛЕНИХ СХОВИЩАХ ДАНИХ.....	30
2.1 Моделі логічних годинників у РСД.	30
2.2 Загальна модель РСД.....	36
2.3 Модель мережі РСД.....	38
2.4 Моделі централізованого СД.....	42
2.5 Модель децентралізованого РСД.....	45
2.6 Висновки за розділом	51
РОЗДІЛ 3 ОБЧИСЛЮВАЛЬНІ МЕТОДИ ДЛЯ ЗАБЕЗПЕЧЕННЯ САР-ГАРАНТІЙ	52
3.1 САР-теорема як математичний факт	52
3.2 Загальні метрики для оцінки САР-гарантій у РСД.....	53
3.3 Метод обчислення рівня узгодженості.....	55
3.4 Метод обчислення ціни відновлення узгодженості	59
3.5 Метод контрольованого балансування узгодженості	67
3.6 Метод для прийняття рішень при проектуванні РСД.....	90

	3
3.7 Висновки за розділом	95
РОЗДІЛ 4 ОПИС ПРОГРАМНИХ МОДЕЛЕЙ ДЛЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ	97
4.1 Опис предметної області та інструментарію дослідження	97
4.2 Опис програмної моделі для оцінки рівня узгодженості	98
4.3 Опис програмної моделі для оцінки процесів реплікації	102
4.4 Теоретичні та практичні оцінки ефективності методу контрольованого балансування узгодженості	107
4.5 Висновки за розділом	111
ЗАГАЛЬНІ ВИСНОВКИ ЗА РОБОТОЮ	113
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	115
ДОДАТОК А СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА	126
ДОДАТОК Б АКТИ ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ	128
ДОДАТОК В РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ ОБЧИСЛЕННЯ РІВНЯ УЗГОДЖЕНОСТІ	131
ДОДАТОК Г РЕЗУЛЬТАТИ СИМУЛЯЦІЇ РЕПЛІКАЦІЇ У РСД	133
ДОДАТОК Д ПРОГРАМНА МОДЕЛЬ ОБЧИСЛЕННЯ МАТРИЦІ СЕРЕДНЬОЇ ДОСТАВКИ У РСД	134
ДОДАТОК Е РЕЗУЛЬТАТИ СИМУЛЯЦІЇ МЕТОДУ КОНТРОЛЬОВАНОГО БАЛАНСУВАННЯ УЗГОДЖЕНОСТІ	135
ДОДАТОК Ж РЕЗУЛЬТАТИ ВИМІРЮВАНЬ ПРОСТОРОВОЇ ТА ЧАСОВОЇ СКЛАДНОСТІ У СИМУЛЯЦІЇ МЕТОДУ КОНТРОЛЬОВАНОГО БАЛАНСУВАННЯ УЗГОДЖЕНОСТІ	137

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CAP – Consistency, Availability, Partition Tolerance (узгодженість, доступність, стійкість до розділень)

CA – Consistency, Availability (узгодженість, доступність)

CP – Consistency, Partition Tolerance (узгодженість, стійкість до розділень)

AP – Availability, Partition Tolerance (доступність, стійкість до розділень)

ACID - Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізолюваність, довговічність)

BASE – Basically Available, Soft state, Eventual Consistency (базова доступність, нестійкий стан, узгодженість врешті-решт)

CCSL – Clock Constraint Specification Language (Мова специфікації обмежень годинника)

FIFO – «First in – first out» queue (черга «першим увійшов – першим вийшов»)

MARTE - Modeling and Analysis of Real-Time and Embedded systems (моделювання та аналіз систем реального часу та вбудованих систем)

HA – High Availability (Висока доступність)

SDLC – software development lifecycle (цикл розробки програмного забезпечення)

SQL – Structured Query Language (мова структурованих запитів)

UML – Unified Modeling Language (уніфікована мова моделювання)

БД – База даних

РСД – Розподілене сховище даних

СД – Сховище даних

СУБД – Система управління базою даних

ВСТУП

Актуальність теми. В епоху повсюдного використання сучасних технологій, таких як Big Data, Internet of Things, хмарні обчислення тощо, дані стають все більш важливими і потребують більшого, надійного зберігання. Це призводить до збільшення розміру розподілених сховищ, що далі потребують величезної мережі по всіх розподілених вузлах. Але існує декілька невирішених проблем із використанням великих розподілених сховищ даних, і деякі з них тісно пов'язані з теоремою CAP (Consistency, Availability, Partition tolerance).

Враховуючи, що відома стратегія ACID (atomicity, consistency, isolation, durability), яка використовується скрізь у реляційних базах даних, не може задовольнити всі потреби систем класу розподілених сховищ даних, механізми передачі даних крізь розподілене сховище не забезпечують в достатній мірі узгодженість, надійність та толерантність до мережових розділів (основні фактори надійного сховища даних).

Масштабуванню розподілених систем останніми роками і дослідженню їх продуктивності приділяється багато уваги. Проблема CAP-теореми (не можна одразу задовольнити три основні характеристики у розподіленому сховищі) і порівняння BASE (Basically Available, Soft state, Eventual consistency) та ACID моделі вивчається вже понад 15 років. За останні 10 років з'явилося понад 40 тисяч статей, що присвячені цій темі. Є статті з аналізом CAP-теореми відомих американських учених Гілберта та Лінч, де розглядаються варіанти досягнення компромісу між AP (доступність, стійкість до розділень) або CP (узгодженість, стійкість до розділень) і CA (узгодженість, доступність). Google нещодавно розробили розподілену базу даних, яка забезпечує CP і майже забезпечує всі три гарантії, але це рішення дуже дороге у його підтримці. Тобто стає зрозумілим, що проблема CAP-теореми досліджена точково з різних боків, і немає систематизованого алгоритму підтримки узгодженості, доступності та стійкості до розділень, який був би стверджений стандартом. У роботі переслідується мета розробити

універсальний метод прийняття рішень, на який можливо спиратись, будуючи мережу для розподіленого сховища даних, що дозволить мати конкретні критерії для формування сховища за умов різноманітних потреб конкретної розподіленої системи. Для цього необхідно дослідити розподілені сховища даних, і щоб не порушити і не втрутитись у його роботу та заради уникнення помилок у дослідженні, будуються математичні і імітаційні моделі для такого сховища. Таким чином, постановка задачі дослідження даної дисертаційної роботи є актуальною.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалася на кафедрі теоретичної та прикладної інформатики факультету математики та інформатики Харківського національного університету імені В.Н. Каразіна відповідно до завдань прикладної НДР «Моделі інформаційних процесів та методи їх обробки» ДР № 0116U003141.

Мета і задачі дослідження. Метою дисертаційної роботи є забезпечення контрольованого рівня узгодженості реплік даних в РСД (розподіленого сховища даних) при необхідних рівнях доступності даних і стійкості сховища даних до розділення на етапі проектування розподіленої системи задля управління ризиками некоректного функціонування системи за рахунок побудови математичних моделей і розробки обчислювальних методів їх дослідження. Для досягнення даної мети в роботі поставлені та вирішені такі задачі:

- побудувати математичні моделі для формалізації, обґрунтування і розробки методів аналізу процесів реплікації;
- запропонувати метрики оцінювання рівня задоволення CAP-вимог;
- запропонувати обчислювальні методи для підвищення рівня узгодженості реплік РСД на основі раціонального збалансування запропонованих метрик;
- реалізувати імітаційні моделі для експериментального дослідження розроблених методів;

- оцінити часову та просторову складність розроблених обчислювальних методів.

Об'єктом дослідження є процеси реплікації даних у великих розподілених сховищах даних (РСД).

Предметом дослідження є математичні та імітаційні моделі процесу реплікації даних у РСД, методи забезпечення узгодженості.

Методами дослідження в цій дисертаційній роботі є математичний базис теорії графів та теорії ймовірності, програмне моделювання для імітаційних моделей розподіленого сховища даних, де використовувались інструменти мови UML та програмне забезпечення для проведення експериментальних досліджень на РСД.

Наукова новизна отриманих результатів полягає в такому:

- зазнала подальшого розвитку стохастична модель розподіленої системи за рахунок забезпечення формалізації виконання CAP-гарантій, яка, на відміну від існуючих моделей, об'єднує формалізовані визначення для CAP-гарантій в одну цілісну структуру і дає підставу для оцінки гарантій на іншому рівні;

- вперше запропоновано систему статистичних метрик для оцінювання виконання CAP-гарантій у РСД, що базується на формалізації визначень CAP-гарантій, і замість бінарної оцінки гарантій РСД пропонується оцінювати систему гарантій за допомогою ймовірнісних величин, що дає підставу для розвитку метрик для оцінки рівня CAP-гарантій;

- удосконалено методи обчислення ймовірностей втрати даних на основі запропонованої моделі, що дає можливість оцінки CAP-гарантій і формування метрик для оцінки з більшою точністю, з врахуванням реальних умов таких систем, і, таким чином, це дає підставу для формування методу регулювання CAP-характеристик на рівні, потрібному конкретній розподіленій системі;

- зазнали подальшого розвитку методи забезпечення CAP-вимог шляхом розробки нової стратегії контрольованого балансування узгодженості у формі

спеціального методу контрольованого балансування узгодженості, що може бути інтегрований в методи реплікації у РСД.

Практичне значення. На основі запропонованих методів та моделей реалізовано метод регулювання CAP-гарантій у формі алгоритму для покриття такого рівня CAP-гарантій, які будуть необхідні конкретному РСД. Цей метод може бути застосований на етапі проектування системи РСД для запобігання помилок і неоптимальних рішень побудови розподіленої мережі, а також невірного вибору або застосування методів забезпечення CAP-гарантій, тобто зменшення ризиків, які коштуватимуть дорожче на етапі застосування такої системи. Також такий метод містить елементи, які мають працювати на етапі застосування системи, а саме регулювання рівня узгодженості з метою утримування її на необхідному рівні. Розроблені в дисертаційній роботі моделі, метрики та методи можуть бути використані як компоненти систем моніторингу РСД з метою контролювання CAP-гарантій і оперативного реагування на інциденти в системі. Тож, такі методи використовуються для підвищення ефективності роботи РСД, мережі в яких налічують сотні і тисячі вузлів.

Використання результатів роботи. Результати роботи використані в освітньому процесі математичного факультету Запорізького національного університету за освітньо-професійною спеціальністю 122 – «Комп'ютерні науки» при викладанні дисципліни «Паралельні та розподілені обчислення», а також в освітньому процесі факультету комп'ютерних наук, фізики та математики Херсонського державного університету за освітньо-професійною спеціальністю 126 – «Інформаційні системи і технології» при викладанні дисципліни «Управління інформаційними системами та сховищами даних» (Додаток Б).

Особистий внесок здобувача. Наукові результати, викладені в дисертаційній роботі, одержано здобувачем особисто. У роботах, які написано в співавторстві, особистий внесок здобувача полягає у наступному: сформовані елементи мови для визначення моделі годинника, що може

використовуватися у розподіленій системі для обчислення точного часу при визначенні версій реплік і визначення, яка репліка найновіша, з метою коректної роботи операції заміни репліки на новішу [1, 5]; розширено модель оцінки CAP-гарантій за рахунок розвинення метрики оцінки часу для виконання повної реплікації в реальних умовах розподіленої системи [3]; застосовано метод балансування рівня CAP-гарантій залежно від потреб системи [4]; зроблено аналіз моделі якості розподіленого сховища даних, що дало можливість оцінити, які саме характеристики більш важливі для такої системи, формалізовано визначення для CAP-гарантій [6]; розвинено метрики для оцінки рівня гарантії узгодженості у РСД та доведено гіпотезу про час узгодженості у РСД, що дало підставу для розвитку оцінки часу для виконання повної реплікації, тобто відновлення системи до повністю узгодженої [7];

Апробація результатів дисертації. Основні положення і результати дисертаційних досліджень пройшли апробацію та представлені на 9th International Conference on ICT in Education, Research, and Industrial Applications ICTERI (Херсон, 2013), 11th International Conference on ICT in Education, Research, and Industrial Applications ICTERI (Львів, 2015), XIV Міжнародній науково-практичній конференції «Теоретичні та прикладні аспекти побудови програмних систем» ТАAPSD'2017 (Київ, 2017), а також на VI міжнародній науково-технічній конференції «Комп'ютерне моделювання та оптимізація складних систем» (Дніпро, 2020).

Публікації. За результатами дисертаційної роботи опубліковано 8 наукових праць, у тому числі 1 стаття в іноземному спеціалізованому виданні (наукометрична база «Scopus») [1], 3 статті у наукових фахових виданнях України з технічних наук [2–4], і 4 тези доповідей на фахових міжнародних наукових конференціях [5–8]. Список публікацій подано у Додатку А.

РОЗДІЛ 1

АНАЛІЗ СУЧАСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПОДІЛЕНИХ СХОВИЩ ДАНИХ

1.1 Централізовані і децентралізовані сховища даних і їх властивості.

За останні десять років популярність веб-додатків зростала. Незалежно від того, чи програми створюються для кінцевих користувачів як окремі додатки, чи як онлайн-сервіс, є розрахування на те, що програма знайде широке застосування, а завдяки широкому впровадженню відбудеться зростання операцій. Якщо програма покладається на постійність даних, що зберігаються, є велика ймовірність, що сховище стане вузьким місцем такої системи. Сучасне вирішення цієї задачі – масштабування. Існує дві стратегії масштабування будь-якого програмного забезпечення з серверною частиною: вертикальне і горизонтальне. Вертикальне масштабування, яке передбачає неперервне оновлення ресурсів сервера відповідно до збільшення навантаження на сервер, у сучасному світі є незручним і дорогим способом, особливо коли потенціал і, відповідно, потреби системи, неперервно зростають. Горизонтальне масштабування, що передбачає додавання інших серверів до системи і розподілення ресурсів, пропонує більшу гнучкість, але складніше в реалізації. Це є причиною появи розподілених систем, зокрема розподілених сховищ даних.

Розподілена база даних (або розподілене сховище даних) – це сукупність логічно взаємопов'язаних між собою баз даних, розподілених у комп'ютерній мережі. Логічний зв'язок баз даних в розподіленій базі даних забезпечує система управління розподіленою базою даних, яка дозволяє управляти розподіленою базою даних таким чином, щоб створювати у користувачів ілюзію цілісної бази даних. Для будь-якої розподіленої бази даних забезпечуються такі основні властивості і функції, встановлених К.Дейтом [9]:

- локальна автономія: вона означає, що управління даними в кожному вузлі виконується локально і незалежно від інших вузлів системи;
- незалежність вузлів: вважається, що в ідеальній архітектурі бази даних всі вузли рівноправні і незалежні, а бази даних є рівноправними постачальниками інформації в загальний інформаційний простір;
- прозорість розміщення даних: розподілена база даних для користувача – це чорний ящик – користувач не мусить знати розташування даних та внутрішні механізми їх модифікації. Під час роботи створюється враження, що дані знаходяться саме на його комп'ютері;
- прозорість мережі – забезпечують певні протоколи в мережі, що обслуговують розподілену базу даних;
- неперервність операцій: операції над даними можуть проводитися неперервно.

Також для розподіленої бази даних існують три найважливіші парадигми, за допомогою яких підтримується ефективна робота для користувача. Вони подані нижче.

Цілісність. Це складне питання в розподіленій системі даних. Його рішення – синхронна і узгоджена зміна даних у кількох локальних базах даних, які складають розподілене сховище. Цілісність досягається застосуванням двофазного протоколу фіксації транзакцій [10]. Якщо в розподіленому сховищі передбачено тиражування даних, це одразу пред'являє жорсткі вимоги до підтримки цілісності на вузлах, куди направляються потоки тиражованих даних. То ж конфлікти щодо змін, які необхідно відслідковувати – неминучі. Цілісність бази даних – комплексне поняття, за яким слідує інші властивості надійної бази даних, що подано нижче.

Узгодженість. Як і у всіх інших розподілених системах, одним з основних компонентів цілісності даних є узгодженість – гарантія того, що усі вузли мають однакові копії даних (які називаються репліками) за одною одиницею даних у будь-який момент часу. Підтримка узгодженості в базах

даних є ключовою в підтримці стабільної роботи розподіленого сховища даних.

За узгодженістю слідує *доступність* даних, що означає, що в будь-який момент клієнт може отримати дані зі сховища або негативну відповідь на некоректний запит за розумний обсяг часу.

Не менш важливою властивістю є *стійкість до розділення* системи – незважаючи на розділення на ізольовані секції або втрати зв'язку з частиною вузлів, система не втрачає стабільності і здатність коректно відповідати на запити.

CAP-теорема. Є відома теорема CAP, яка наголошує, що з трьох властивостей (узгодженість (consistency), доступність (availability), стійкість до розділення (partition tolerance)) неможливо забезпечити більше двох в одній і тій самій конфігурації системи: наприклад, якщо забезпечити узгодженість, втратиться одна з двох інших властивостей.

Така теорема потребує більш детального дослідження процесів управління даними у розподілених сховищах.

Важливо зазначити, що розділяють такі основні моделі узгодженості [11]:

- сувора (повна) узгодженість (Strict Consistency) – на будь-який запит на читання елемента даних сховище дає значення, яке відповідає найновішій версії;

- послідовна узгодженість (sequential consistency) – результат будь-якої дії такий самий, якщо б операції читання та запису всіх процесів у сховище даних виконувались би в деякому послідовному порядку, і при цьому операції кожного окремого процесу виконувались би у порядку, визначеному його програмою. Тобто будь-яке правильне чергування операцій читання і запису є допустимим, але всі процеси бачать одне й те саме чергування операцій;

- причинна несуперечливість (causal consistency) - операції на запис, які потенційно зв'язані причинно-наслідковим шляхом, повинні

обслуговуватися всіма процесами в одному і тому ж порядку, а паралельні записи можуть обслуговуватися на різних вузлах у різному порядку;

- узгодженість врешті-решт – при відсутності змін всі репліки поступово стають узгодженими.

Забезпечення суворої узгодженості - занадто дорога модель для будь-якої розподіленої бази даних, хоча деякі окремі бази це забезпечують, для яких застосування такої моделі - суворі технічні вимоги, що впливає з бізнес-логіки продукту, де така база використовується. Але такі конфігурації розподілених БД втрачають значення іншої характеристики – високої доступності даних, бо для великих БД потрібно багато часу, щоб досягнути суворої узгодженості, і клієнт змушений чекати, щоб мати гарантію на отримання узгоджених актуальних даних. Це є одним з окремих випадків, коли підтверджується CAP-теорема.

Важливо також зазначати, що класифікують реляційні (SQL) та нереляційні (NoSQL) бази даних [12]. У сучасних рішеннях CAP-теорема може виконуватись як в одному, так і в іншому виді баз даних. Але оскільки нереляційні БД не мають такого складного механізму обробки запитів, тобто потенційно обробляють дані швидше, а значить, можуть гарантувати більшу продуктивність, у цьому контексті частіше згадується потреба у високій доступності та, відповідно, питання CAP-теорему.

З огляду на описані вище умови сформувалися два принципово різні набори властивостей для баз даних, які конфліктують між собою, і кожний вибір сховища даних у процесі розробки програмного забезпечення має бути зваженим, оскільки ціна міграції з одної СУБД на іншу після вибору буде тільки збільшуватись.

Частіше для SQL рішень задовольняється ACID (Atomicity, Consistency, Isolation, Durability): атомарність гарантує гарантія, що транзакція буде зафіксована тільки повністю [11], коли виконяться всі її операції або не виконається жодна; узгодженість, про яку вже згадувалось; ізоляваність гарантує, що жодні проміжні зміни не будуть видимими за межами транзакції

до її успішного завершення, то ж в умовах паралельної роботи з даними, транзакції є незалежними. За останньою гарантією, вірно таке: якщо дві транзакції намагатимуться змінити одні й ті самі дані, то одну з них буде відхилено або призупинено до завершення другої. Такі властивості характерні для реляційних баз даних, які мають гарантувати цілісність БД [12].

BASE (Basically Available, Soft state, Eventually Consistent). Цей набір властивостей дозволяє досягти високої доступності (High Availability, HA [13]) у розподіленому сховищі даних (РСД). Перша вимога гарантує максимальну доступність, яку можна отримати у системі шляхом використання всіх вузлів у мережі РСД. Друга – те, що після запису отримання нового значення можливо тільки з певною ймовірністю – залежно від того, на який вузол прийде запит на отримання даних, оскільки система РСД ще не досягла повної узгодженості. Третя гарантує, що система врешті-решт зійдеться до повної узгодженості через невизначений час, коли не буде отримувати запитів на запис, і одна репліка розповсюдиться всім сховищем. Така характеристика як раз і гарантує узгодженість врешті-решт, що описана вище.

Така проблематика вимагає подальшого дослідження, розробки нових методів забезпечення гарантій узгодженості і доступності, а також вдосконалення існуючих методів забезпечення цих гарантій.

Дослідження розподілених систем практично можливо тільки шляхом моніторингу і фіксування певних метрик, а також шляхом тестування на окремому ізольованому середовищі для того, щоб не втручатися у неперервну роботу системи. Тестування РСД з великою мережею (в 10 і більше вузлів) у окремому середовищі – це надто дорогий шлях дослідження таких великих систем. Тому необхідно спершу використовувати для нових рішень інший підхід. Єдиною альтернативою дослідження є математичне моделювання. Такий підхід не новий, дослідження розподілених систем за допомогою моделювання започаткували ще такі відомі вчені і лауреати премії Тьюрінга як Леслі Лампорт, Тоні Гоар, Робін Мілнер [14–16]).

1.2 Аналіз сучасних досліджень і актуальність досліджень РСД

Масштабуванню розподілених систем останніми роками і дослідженню їх продуктивності приділяється багато уваги. Проблема CAP-теореми, BASE та ACID моделі, що розглянуті у підрозділі 1.1, вивчаються вже понад 15 років.

За останні 10 років є понад 40 тисяч статей, що присвячені темі CAP-теореми і темі CAP-вимог. У 1999 році Ерік Брюер, відомий вчений, що досліджував можливості масштабування у РСД, опублікував CAP-теорему як гіпотезу [17]. У 2002 році американські науковці Сет Гілберт і Ненсі Лінч опублікували формальне доведення такого факту у розподілених сховищах даних [18]. У 2012 році Брюер опублікував ще одну статтю, де пояснення CAP-теореми відкориговано і надані рекомендації з забезпечення CAP-гарантій [19]. Нарешті, у 2017 році Брюер опублікував найвидатніше з досліджень, присвячене теоремі CAP [20], що сприяло розвитку Cloud Spanner – розробка міжнародної корпорації Google, де працює Ерік Брюер.

Є також статті з аналізом CAP-теореми учених Гілберта та Лінч [21], де розглядаються варіанти досягнення компромісу між AP (доступність, стійкість до розділень) або CP (узгодженість, стійкість до розділень) і CA (узгодженість, доступність), такі статті мають оглядовий і порівняльний характер, але не вирішують задачу досягнення компромісу між вимогами, оскільки вимоги можуть відрізнятися для кожної специфікації системи.

Отже, як вже згадано вище, нещодавно Google розробили розподілену базу даних Google Spanner [22]. Ця база – одне з найкращих сучасних рішень для забезпечення CAP-вимог, але забезпечує не сувору узгодженість, а, так би мовити, «зовнішню» узгодженість (або послідовну узгодженість, див. визначення у підрозділі 1.1), що означає, що репліка доступна у такому порядку, у якому різні її версії записані. Google Spanner забезпечує CP і майже забезпечує всі три гарантії CAP, але це рішення досить дороге у його підтримці, оскільки засновано на використанні високо доступного (атомного) годинника – інструменту, що має назву «True Time» [23], що забезпечує впорядкованість транзакцій і гарантує відсутність конфліктів, а також

гарантує узгоджене читання у розподіленій базі даних – правильний порядок запису транзакцій і, відповідно, правильний порядок їх читання. Але все ж таки, після першого запису репліки дані повинні ще розповсюдитися всією мережею, то ж можна зробити висновок, що сувора узгодженість все ще не задовольняється.

Стає зрозумілим, що проблему CAP-теореми досліджено точково з різних боків, і немає систематизованого алгоритму підтримки узгодженості, доступності та стійкості до розділень, який був би стверджений стандартом. Є різні практики, що застосовуються у тому чи іншому разі залежно від потреб програмного забезпечення. У роботі переслідується мета розробити універсальний метод прийняття рішень, на який можливо спиратись, будуючи мережу для розподіленого сховища даних, що дозволить мати конкретні критерії для формування сховища за умов різноманітних потреб конкретної розподіленої системи. Для цього необхідно дослідити розподілені сховища даних, і щоб не порушити і не втрутитись у його роботу та заради уникнення помилок у результатах дослідження, будуються математичні і імітаційні моделі для такого сховища. Таким чином, постановка задачі дослідження даної дисертаційної роботи є актуальною.

Аналіз, поданий вище у цьому підрозділі, показує, що нема структурованого стандартизованого рішення, що могло б послужити механізмом регулювання CAP-гарантій та механізмом прийняття рішень, що дасть можливість знайти баланс між гарантіями, залежачи від конкретної специфікації потреб системи РСД, що і стало підставою для даного дослідження.

1.3 Порівняльна характеристика реалізацій РСД з відкритим кодом

У цій дисертаційній роботі пропонуються удосконалені методи забезпечення CAP-вимог, які можуть бути інтегрованими з існуючими механізмами різних розподіленими баз даних. Дивлячись на таке, є логічним дослідити можливості певного внеску у існуючі рішення РСД, а саме проаналізувати сумісність архітектур запропонованих та існуючих змін, а

також ефективність обробки запитів на запит та читання. Для цього необхідно зробити оцінку архітектур СУБД з відкритим кодом, які надають можливість робити певні внески до їх коду.

Отже, серед рішень БД з відкритим кодом, які мають функцію розподілення бази даних, опишемо найбільш популярні і ті, що забезпечують необхідний функціонал розподілення даних.

- PostgreSQL

Розподілення даних. Оскільки PostgreSQL – реляційна СУБД, масштабування даних, конфігурація і налаштування такої БД дається непросто. Основною складністю є саме горизонтальне та вертикальне розбиття даних – через високі вимоги до доступності даних з’являється питання про ефективність їх отримання з різних серверів у разі, якщо потрібні дані з декількох таблиць, розділених по різних вузлах і пов’язаних між собою шляхом відношень (реляцій SQL [24]). Декілька років назад PostgreSQL представили механізми масштабування та розбиття даних, але все ж таки ця СУБД продовжує розвивати це рішення і підтримує невелику кількість вузлів (у найкращому випадку до 48 [25]), і при зберіганні великого об’єму даних по різних вузлах може бути неефективною, оскільки при кожному записі даних БД має вирахувати хеш для первинного ключа (складність запису даних у БД відома [26]).

Отже, у загальному випадку, коли від масштабування очікується не тільки висока доступність і ефективність при запитах на читання, а й більш ефективні записи до БД, вибір такого інструменту буде мати певні переваги. Масштабування з використанням цієї СУБД підходить, коли не планується розвивати кількість вузлів у мережі РСД до сотень вузлів і не планується багато даних до запису, інакше така інфраструктура ставатиме менш дієздатною, ніж інші, при дуже великих кількостях запитів, наявності дуже великої мережі.

Механізми реплікації. PostgreSQL підтримує реплікацію у такий спосіб, проілюстрований на Рис. 1.1.

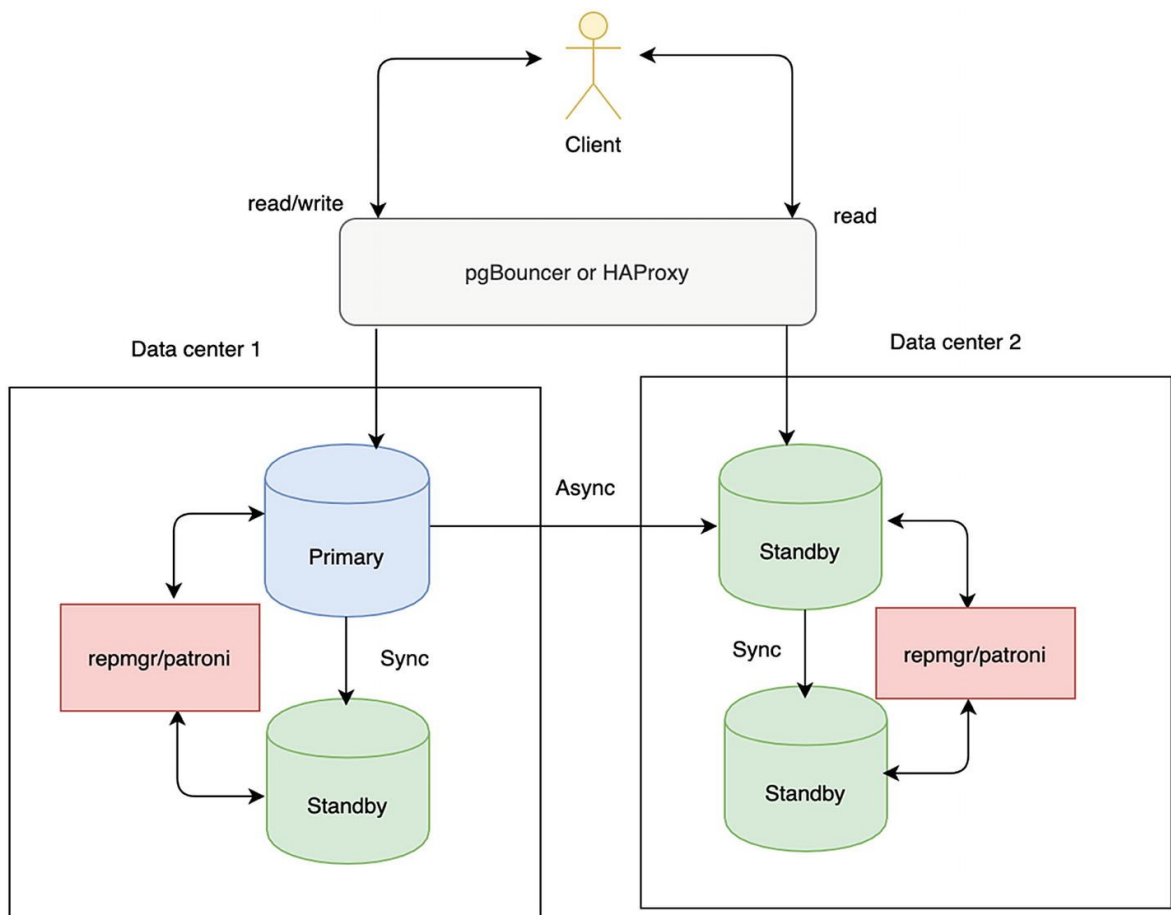


Рис. 1.1. Архітектура реплікації у PostgreSQL

Отже, коли користувач відправляє запит на запис через шлюз (pgBouncer на ілюстрації), починається процес реплікації і вузол-лідер (Primary) генерує репліку – копію записаної одиниці даних, яка асинхронно поширюється одразу всім вузлам-послідовникам – Node A, Node B, Node C (у термінах PostgreSQL вони називаються StandBy сервери [25]).

Такий підхід дозволяє швидше розповсюдити репліку, але виникає питання ефективності топології такої мережі у разі великої кількості вузлів – навантаження на первинний (Primary) вузол зростає пропорційно із зростанням вузлів-послідовників у мережі. У разі підтримки розповсюдження реплік епідемічним протоколом з'являються ті ж самі питання про час реплікації і очікування того, коли репліка розповсюдиться всією мережею.

Забезпечення гарантій CAP. Отже, описувана СУБД підтримує узгодженість та доступність з вимог CAP (що називається CA). Тобто, PostgreSQL забезпечує наразі розподілення даних на кластері, і підтримує

високу доступність (не для великих мереж); транзакції у БД працюють згідно протоколу двофазному коміту [10], що у разі неуспішного виконання транзакції, виконає відкат змін назад. Розповсюдження реплік відбувається асинхронно або синхронно залежно від виду використаної реплікації (ці механізми називаються «поточною реплікацією» та «синхронною реплікацією», відповідно [27]), що забезпечує узгодженість у цієї системі. Але якщо відбудеться розділення, в більшості випадків взаємодія з системою неможлива, якщо пріоритет у підтримки узгодженості. Якщо ж зробити систему стійкою до розділень, є висока ймовірність, що СУБД може відповідати різними репліками.

- MySQL.

Розподілення даних. Як і PostgreSQL, MySQL – реляційна СУБД, і аналогічно має лімітовані можливості до розподілення. Ця СУБД підтримує вертикальне і горизонтальне розбиття даних – одиницею даних може вважатися одна або кілька пов’язаних таблиць у вертикальному розбитті, або одиниця даних – вибрані рядки з таблиці або кількох за певним принципом у випадку горизонтального розбиття. Коли приходить новий запит на запис, репліки за цією одиницею даних розповсюджуються сховищем.

Механізми реплікації. Реплікація у MySQL реалізована згідно схеми, що подано на Рис. 1.2.

У цій СУБД реплікація працює за схожими принципами PostgreSQL. Як і у попередній СУБД, також підтримується стійкість до відмов: після відмови головного сервера роль лідера передається іншим серверам для подальшого функціонування сховища даних і дає системі можливість бути високо доступною.

Забезпечення гарантій CAP. Завдяки описаним вище механізмам, гнучкому налаштування у MySQL підтримується або узгодженість і доступність (CA), або узгодженість і стійкість до розділень (CP) [28]. Також MySQL надає змогу налаштувати доступність та стійкість до розділень (AP)

шляхом розподілення ролі головного сервера між різними вузлами та асинхронного способу реплікації.

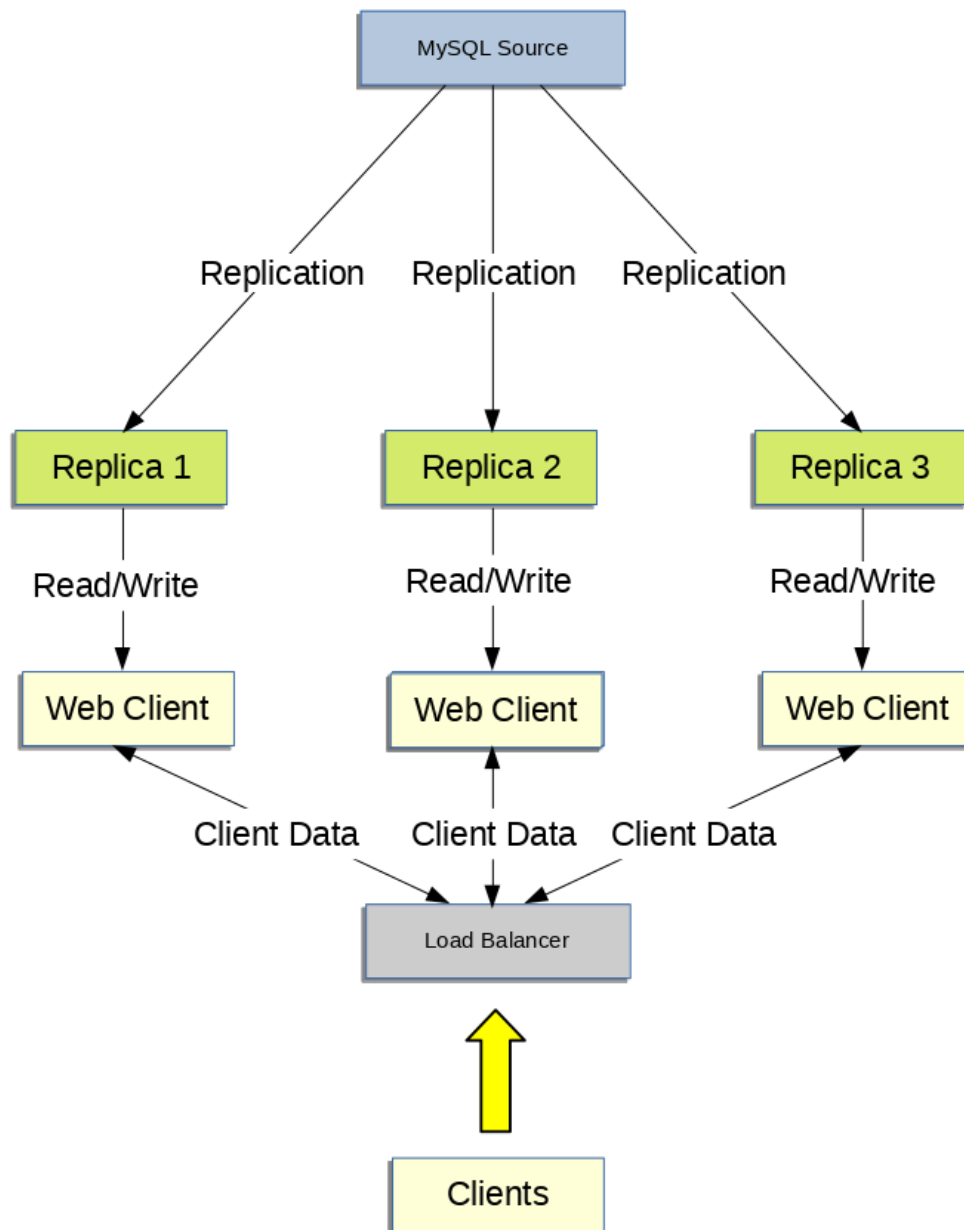


Рис. 1.2. Механізм реплікації у MySQL

- Google Spanner

У контексті високо доступних сучасних рішень реляційних баз даних не можна не згадати про Google Spanner. Це сучасна реляційна СУБД, яка гарантує високу доступність.

Розподілення даних. Розділення даних підтримується у Google Spanner, як і у будь-якій реляційній СУБД, що описані вище.

Механізми реплікації. Реплікація у Google Spanner відбувається на рівні файлів, на високому ж рівні дані організовані як рядки у сховищі. Кожний рядок копіюється, і така копія називається реплікою, яка розповсюджується мережею далі, в різні географічні локації. Оскільки Google Spanner підтримує глобальну мережу, гнучкість будування мережі з цією СУБД зростає. У процесі реплікації використовується алгоритм Paxos, що забезпечує узгоджений запис до бази даних: при кожному запиті на запис робиться вибір репліки, що буде записано. У термінах Google Spanner репліки мають три можливі ролі: репліка, яка перезаписує інші репліки, а також надає власні дані; репліка, яка тільки надає власні дані; репліка, яка приймає участь у голосуванні вибору репліки-лідера та у голосуванні значення, яке має бути записане у БД. Вибрана репліка після голосування матиме роль лідера і буде відповідати за запис даних. Між іншим, інші репліки можуть відповідати на запити на читання, якщо система не потребує забезпечення суворой узгодженості. У разі, якщо сувора узгодженість потрібна системі, поточна репліка звернеться до репліки-лідера з перевіркою, чи вона узгоджена з реплікою-лідером. Також, навіть якщо сувора узгодженість не потрібна, Google Spanner забезпечує послідовну узгодженість шляхом присвоювання кожній транзакції часової мітки з використанням годинника «True Time».

Забезпечення гарантій CAP. Розробники Google Spanner досягли найбільших успіхів у задоволенні гарантій CAP, але все ще неможливо гарантувати задоволення всіх компонентів CAP повністю – у будь-якому разі це надто відповідальне твердження, і ніхто з учених у цій темі не може гарантувати це. Google Spanner завжди гарантує узгодженість, тобто забезпечує або набір CP, або CA у контексті CAP-теореми [23]. Тобто, якщо розділення все ж таки трапляються у мережі, ця СУБД між доступністю і узгодженістю вибере узгодженість. Система може все ще залишитися високо доступною, але не на 100%.

- Mongo.

Розподілення даних. Mongo – система управління базою даних, яка зберігає дані у так званих «документах» [29]. У NoSQL СУБД розподілення даних набагато простіше, ніж у реляційній базі даних [30]. Оскільки Mongo дозволяє не вибудовувати складні реляції між таблицями (хоча при певному налаштуванні дозволяє створювати нескладні реляції між документами), розподілення даних проходить легшим шляхом, і отримання таких розподілених даних відбувається набагато легше. Це є вірним ще й тому, що мова для виконання запитів до БД NoSQL набагато простіша за звичні всім запити до реляційної БД. У контексті реплікації Mongo одиницею даних є певний набір даних («dataset», «replica set» [31]), який розподіляється на початку налаштування такої СУБД. Також ця СУБД підтримує горизонтальне та вертикальне розділення даних.

Механізми реплікації. Схему реплікації у Mongo подано на Рис. 1.3. Набір реплік розподілений на різні вузли бази даних в Mongo: на первинний та вторинні вузли (у інших варіаціях реплікації раніше такі вузли називаються лідером або головним вузлом та послідовниками). Загальний механізм реплікації майже нічим не відрізняється від інших, окрім того, що інструменти для налаштування реплікації набагато простіший, ніж у реляційних СУБД. Реплікація забезпечується епідемічними протоколами з сімейства Gossip [32].

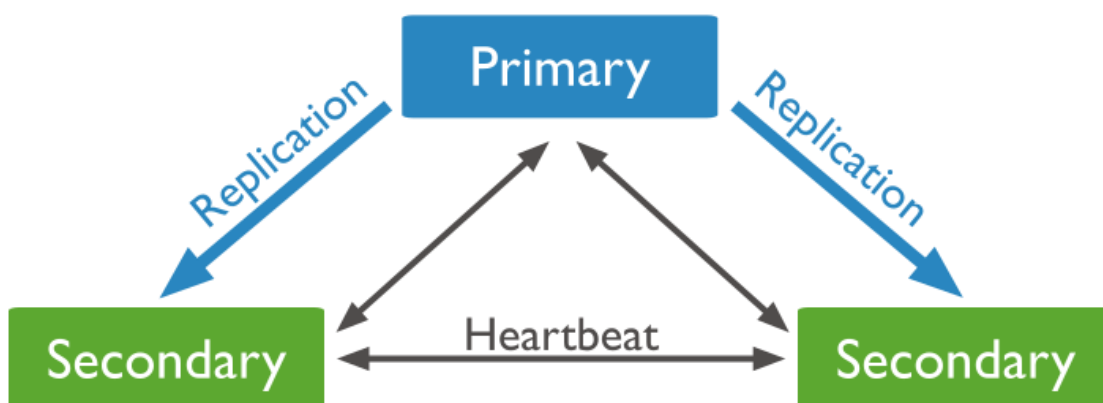


Рис. 1.3. Реплікація у MongoDB

У Mongo, як і у багатьох інших нереляційних СУБД, забезпечується стійкість до відмов – у разі падіння первинного вузла розпочинається процес

вибору нового лідера. Після відновлення попередній лідер працює у ролі вторинного вузла до тих пір, як його не виберуть лідером наступного разу.

Забезпечення гарантій CAP. Найчастіше налаштування MongoDB відповідає моделі BASE, і забезпечує високий рівень доступності та стійкість до розділень (AP у термінах CAP-теореми) – у цьому варіанті налаштування . Але налаштування за замовчуванням підтримує узгодженість і стійкість до розділень (CP), коли всі запити проходять через первинний вузол. Це робить систему не високо доступною, але узгодженою, тому що у такому випадку з'являється вузьке місце, і при великому навантаженні таке налаштування – неефективне.

- Cassandra.

Розподілення даних. Apache Cassandra – нереляційне рішення децентралізованого сховища даних [33], то ж розподілення даних має схожі загальні механізми, як і MongoDB.

Механізми реплікації. У процесі реплікації стратегія має знання про те, на яких вузлах зберігаються репліки. Всі репліки між собою рівнозначні: нема первинного або вторинного вузла або репліки-лідера [34]. Cassandra має дві стратегії реплікації: проста стратегія, не рекомендована для великої мережі в різних дата-центрах, та стратегія топології мережі [35]. На останній зупинимося більш детально, бо така стратегія більше підходить для великих мереж РСД у багатьох дата-центрах, які досліджуються у цій дисертаційній роботі. Ця стратегія дозволить визначити, скільки реплік буде зберігатися у кожному дата-центрі. Роблячи вибір, скільки реплік слід налаштувати в дата-центрі, двома основними критеріями є такі: можливість задовольнити запит на читання локально та наявність сценаріїв відмов. При обробці запитів Cassandra надає можливість налаштувати рівні узгодженості для балансування між узгодженістю та доступністю даних. Наприклад, найвищий рівень узгодженості спричиняє відповідь на читання тільки тоді, коли всі репліки узгоджені між собою. Це приводить до найвищого рівня узгодженості і найнижчого рівня доступності. Або репліки запит на читання може повернути

репліки, коли репліки з різних дата-центрів досягли кворуму. Є і інші налаштування для реплікації в Cassandra – є декілька таких налаштувань, які дозволяють управляти рівнем узгодженості і доступності на трьох рівнях (середній, мінімальний, тощо) [36].

Забезпечення гарантій CAP. Apache Cassandra у класичному налаштуванні задовольняє узгодженість та доступність, тобто є AP системою. Але можливо налаштувати систему так, щоб вона підтримувала будь-яку з комбінацій з двох гарантій з теореми CAP (CA, CP, AP).

Виконавши порівняльну характеристику механізмів реплікації та забезпечення CAP-вимог у цьому підрозділі, можна зробити висновок, що більшість популярних рішень для розподілених сховищ даних підтримують дві з трьох CAP-гарантій у різних комбінаціях залежно від налаштування. Ті рішення, що не увійшли до порівняння, мають схожі механізми реплікації та методи забезпечення CAP-гарантій.

Також, є можливість внеску майже у всі описані рішення, окрім Google Spanner.

1.4 Маршрутизація запитів у розподіленій системі даних

Важливим компонентом горизонтального масштабування будь-якого програмного забезпечення, яке працює з користувачем засобами комунікаційних протоколів, є балансування навантаження [37]. Тому важливо розглянути балансувальник навантаження як технологію, що використовується у мережах РСД, та методи балансування, що використовує ця технологія.

Балансувальник навантаження – це метод, який дозволяє розподіляти задачі між мережевими пристроями з метою оптимізації використання ресурсів, збереження часу відповіді на запити, горизонтального масштабування кластеру та забезпечення стійкості до відмов. На Рис. 1.4 подано загальний механізм роботи балансувальника навантаження: запит приходить спочатку на проміжний вузол і далі перенаправляється на один з вузлів в системі, який спроможний відповісти на запит клієнта.

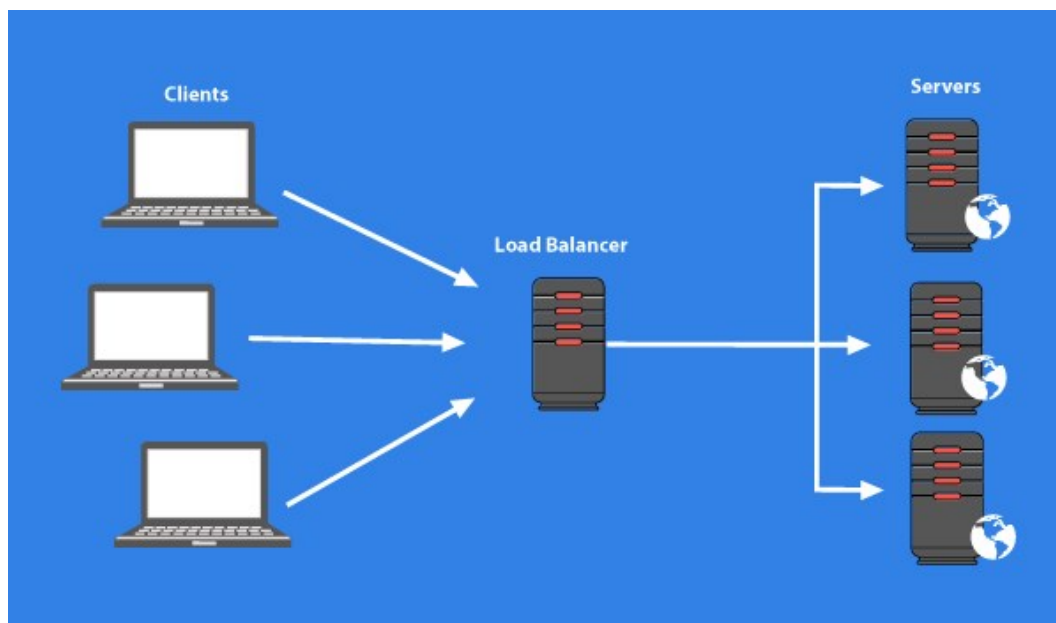


Рис. 1.4. Загальний механізм балансувальника навантаження

Також є балансувальники, які підтримують балансування на рівні протоколів БД.

Періодично балансувальник виконує так звану перевірку вузла («healthcheck» [38]), щоб вчасно мати знання про спроможність вузлів відповісти на запит. Балансування може здійснюватися за різними додатковими характеристиками: запит буде пересланий на найбільш незайнятий вузол або той, який за показниками процесора на даний момент часу має не завантажений процесор тощо. Такий механізм балансування забезпечується методом балансування, реалізованим у формі алгоритму, який налаштовується перед початком роботи системи. Класифікація алгоритмів балансування навантаження подана нижче. Ці алгоритми спершу розділяють на статичні і динамічні.

1. Статичне балансування [39]. Статичні методи балансування поводяться таким чином, що якщо вже якийсь вузол виділений для певного процесу, процес не може бути перерозподілений на інший вузол. Цей метод потребує менше взаємодії, а значить, зменшує час виконання операцій. Однак основна перешкода такого підходу у тому, що механізм не зважає на поточний стан системи при розподіленні задач на вузли. Це має значний вплив на продуктивність всієї розподіленої системи через велику амплітуду коливання

навантаження. Статичні методи балансування у свою чергу розділяються на різні алгоритми, що описані нижче.

- Алгоритм циклічного планування (Round Robin) [39]. В цьому алгоритмі навантаження розподіляється рівномірно на всі вузли мережі, де рівне навантаження призначається у кільцевому порядку без будь-якого пріоритету та повертається у кінці на перший вузол. Кожний вузол виконує свою роботу, яка була для нього виділена, незалежно від розташування інших вузлів. Алгоритм циклічного планування – простий у роботі, легко реалізовується. Цей алгоритм не потребує комунікації між процесами та дає кращу продуктивність для додатків спеціального призначення. Однак алгоритм не може дати очікуваний результат в загальному випадку і коли процеси потребують різної кількості часу для виконання.

- Алгоритм «Центральний менеджер» (Central Manager Algorithm). В цьому алгоритмі центральний вузол вибирає інший, який називають «послідовник» (follower) для того, щоб «послідовник» взяв навантаження на себе. Навантаження призначається тому вузлу, який має у цей момент часу найменше навантаження. Центральний вузол має знання про індекс навантаження всіх вузлів-послідовників, які приєднані к центральному. Кожного разу, як міняється навантаження, усі вузли отримують повідомлення про це від центрального вузла. Цей алгоритм потребує високий рівень комунікації між вузлами, який іноді може привести до стану «вузького місця» у системі (у термінах розподіленого програмного забезпечення та мереж називається bottleneck [40]): коли навантаження на вузол занадто велике, і система втрачає певний рівень високої доступності. Цей алгоритм має кращу продуктивність, коли динамічні активності створюються різними вузлами.

- Алгоритм Порогу (Threshold Algorithm) [41]. Відповідно до цього алгоритму, навантаження призначається одразу на етапі створенні вузла. Вузли вибираються локально без відправки повідомлень на інші вузли. Кожний вузол зберігає приватну копію навантаження системи. Навантаження характеризується трьома рівнями: недостатнє, середнє навантаження та

перенавантаження. Два параметри t_{under} , t_{upper} використовуються для опису цих рівней.

Від самого початку передбачається, що вузли мають недостатнє завантаження. Коли стан навантаження вузла перевищує межу на певному вузлі, цей вузол надсилає повідомлення щодо нового стану навантаження всім віддаленим вузлам, регулярно оновлюючи їх про поточне навантаження. Якщо стан локального вузла не є перезавантажений, то навантаження виділяється локально. В іншому випадку вибирається віддалений незавантажений вузол, якщо такий є. Цей алгоритм має низький рівень взаємодії між процесами та великою кількістю локальних розподілів процесів. Пізніше зменшуються накладні витрати на віддалений розподіл процесів та накладні витрати на доступ до віддаленої пам'яті, що призводить до поліпшення продуктивності.

- Рандомізований алгоритм. В цьому алгоритмі вузол вибирається випадково, без будь-якої інформації про поточне або попереднє навантаження на вузол. Оскільки алгоритм має статичний характер, він найкраще підходить, коли система має однаковий рівень навантаження на кожен вузол. Дає найкращу продуктивність для спеціальних додатків, які відповідають вимозі рівномірного завантаження. Кожен вузол зберігає власну кількість завантажень, тому не вимагається взаємодії між процесами. Але іноді це може спричинити перевантаження одного вузла у той час, коли інший вузол недостатньо навантажений.

2. Динамічне балансування. Такі алгоритми мають систему моніторингу за змінами в системі та перерозподіляють процеси відповідно до актуальних даних, які може надати моніторинг. Зазвичай цей алгоритм складається з трьох стратегій: стратегія передачі, стратегія розташування та інформаційна стратегія. Стратегія передачі вирішує, які задачі можуть передати дані на інші вузли для обробки. Стратегія розташування призначає віддалений вузол виконати завдання. Інформаційна стратегія – це інформаційний центр для алгоритма балансування навантаження. Ця стратегія є відповідальною за забезпечення ресурсів для стратегій розташування та передачі для кожного

вузла. Динамічні алгоритми можуть контролювати систему у трьох формах: централізованій, розподіленій та напіврозподіленій. В централізованій формі один центральний вузол в мережі призначається відповідальним за весь розподіл навантаження в мережі. В розподіленій мережі відповідальність розділена між всіма вузлами рівномірно. В напіврозподіленій мережа сегментується на кластери, де кожний кластер в сегменті – централізований. Балансування навантаження досягається за допомогою кооперації центральних вузлів всіх кластерів. Наразі є такі динамічні алгоритми балансування:

- Алгоритм центральної черги (Central Queue Algorithm). Цей алгоритм зберігає нові з'єднання та невиконані запити в циклічній черзі FIFO (метод управління чергою, де перший доданий елемент видаляється з черги першим [42]). Якщо в черзі немає доступних з'єднань, запит буферизується, доки не буде доступне нове з'єднання. Такий алгоритм потребує високого рівня комунікації між вузлами.

- Алгоритм локальної черги (Local Queue Algorithm). Цей алгоритм підтримує міграцію між процесами. Ідея полягає в обробці всіх процесів локально. Але коли сервер перенавантажений, всі нові запити на виконання, які надходять до нього, мігрують до іншого вузла, де заздалегідь визначено мінімальну кількість готових процесів. Коли сервер недостатньо завантажений, він запитує про діяльність віддалених вузлів. Віддалені вузли відправляють вузлу-менеджеру навантаження запити на нові процеси, вказуючи у запиті кількість виконуваних процесів. Менеджер навантаження порівнює кількість процесів у запиті і кількість локальних процесів, і у разі, якщо значення у запиті менше за локальне, деякі локальні процеси мігрують з вузла-менеджера до вузлів, що надіслали такий запит.

- Алгоритм найменшого з'єднання (Least Connection Algorithm). Це – стратегія розподілення навантаження на основі з'єднань, присутніх на вузлі. Балансувальник навантаження зберігає журнал повідомлень про з'єднання на кожному вузлі. Число повідомлень збільшується, коли новий зв'язок

встановлюється і зупиняється, коли підключення завершується або закінчується час. Спочатку виділяються вузли з найменшою кількістю з'єднань.

Така порівняльна характеристика алгоритмів балансування дає базис для методу балансування, що описаний у підрозділі 3.5 – балансування може виконуватися за поданим механізмом, але бути наслідуваним від одного з існуючих методів балансування.

1.5 Висновки за розділом

1. Зроблено детальний опис централізованих і децентралізованих сховищ, їх властивостей, обґрунтовані причини виникнення задачі, яка вирішується.

2. Виконано аналіз сучасних досліджень, які пов'язані з задачею, що вирішується, досліджено їх сучасний стан. Проаналізовано актуальність поставленої задачі в дисертаційній роботі.

3. Виконано порівняльну характеристику різних популярних рішень розподілених сховищ даних у контексті їх підходів до управління даними, досліджені можливості внеску до кожного з них, розглянуто механізми процесів реплікації у розподілених сховищах, надано загальну архітектуру обробки запитів. Проаналізовано, які гарантії CAP задовольняються у тому чи іншому рішенні з різними налаштуваннями системи.

4. Обґрунтовано використання балансування навантаження у розподілених системах, надано класифікацію методів балансування, проаналізовано статичні і динамічні методи балансування, обґрунтовано їх ефективність залежно від різних специфікацій розподілених систем, зокрема, систем РСД.

РОЗДІЛ 2

МАТЕМАТИЧНІ МОДЕЛІ В РОЗПОДІЛЕНИХ СХОВИЩАХ ДАНИХ

Даний розділ присвячений розробці математичних моделей процесів обробки даних у розподілених базах даних з метою аналізу розподілених баз даних та розвитку методів забезпечення CAP-гарантій на потрібному рівні.

Роль математичного моделювання у розподілених сховищах неможливо переоцінити. Дослідження розподілених сховищ даних за допомогою математичного моделювання започаткували ще такі відомі вчені і лауреати премії Тюрінга як Тоні Хоар, Леслі Лампорт, Робін Мілн.

Враховуючи складність об'єкту дослідження, запропоновано комплекс моделей для розподіленого сховища даних (РСД), який складається з декількох базових моделей: загальної моделі для опису архітектури РСД, моделі комп'ютерної мережі, моделі управління даними, і методу управління репліками, який описує, як забезпечується узгодженість даних врешті-решт. Також представлена відокремлена модель годинника для РСД, що матиме велике значення для вирішування конфліктів реплік при паралельній обробці запитів на різних вузлах системи.

2.1 Моделі логічних годинників у РСД.

Загально прийнятими є такі вимоги до алгоритмів, які забезпечують функціонування розподіленої системи [11]:

1. жодний з вузлів не має повної інформації про стан системи;
2. кожний вузол приймає рішення, використовуючи локальну інформацію;
3. збій вузла не спричиняє порушення функціонування алгоритму;
4. єдиний (глобальний) час відсутній.

В контексті дослідження функціонування РСД саме вимоги, подані у зазначених вище пунктах 1, 2 і 4, призводять до неузгодженості реплік, оскільки інформація за однією і тією ж одиницею даних на різних вузлах може відрізнятися.

Вирішення проблеми відсутності глобального системного часу є визначальним завданням для забезпечення послідовності виконання

транзакцій у порядку, що відповідає причинно-наслідковим відношенням і порушення якого призводить до некоректного відображення фактів предметної області різними репліками РСД, що може далі спричинити невірне функціонування розподіленої системи.

Для локалізованих систем проблема причинно-наслідкових відношень розв'язується прямолінійно – причина має відбуватися раніше, ніж її наслідок за часом глобального годинника. На відміну від цього, відсутність годинника глобального часу призводить до того, що проблема забезпечення причинно-наслідкових відношень стає важливою складовою при побудові РСД, тобто кожне РСД повинно мати механізм, що забезпечує порядок виконання транзакцій, який не суперечить об'єктивним причинно-наслідковим відношенням між системними подіями, а значить, ймовірність мати вірні результати транзакцій у системі збільшується.

Як приклад наведемо ситуацію отримання вузлом двох завдань виконання транзакцій запису, які отримані майже одночасно або з невеликим часовим інтервалом. У цьому випадку вузол має прийняти рішення, яку транзакцію виконувати, оскільки виконувати потрібно тільки більш пізню транзакцію, бо вона забезпечує актуальність репліки даних для цього вузла або виконати їх у порядку отримання; то ж результатом успішної обробки транзакції у будь-якому випадку повинна бути остання транзакція. Отже, взаємодія таких транзакцій при неправильному рішенні про їх порядок призводить до неузгодженості реплік на різних вузлах РСД. Тому є необхідність побудови моделі логічного часу, що забезпечує можливість ідентифікації правильної послідовності транзакцій для того, щоб забезпечити коректне врахування причинно-наслідкових відношень між подіями в процесі комунікації вузлів системи і запобігти певних ризиків неправильного функціонування РСД через неузгодженість даних.

Проблема логічного часу вперше була поставлена Леслі Лампортом у його роботі [43], в якій була сформульована модель логічних годинників, що знайшла подальший розвиток у роботах Шарля Андре і Фредеріка Малє [44].

Ця модель була досліджена з метою побудови аксіоматичної семантики мови CCSL (Clock Constraint Specification Language), яка є базовою мовою специфікації причинно-наслідкових обмежень стандартного профілю UML для MARTE (Modeling and Analyzing Real-Time Embedded systems) [45]. Важливим для наших цілей в цій роботі є аналог теореми Шпільрайна з теорії впорядкованих множин [46]. Для повноти викладення ми даємо доведення цієї теореми (Твердження 2.4), базуючись на так званих годинниках Лампорта [43], які дають програмний інструмент як алгоритм, що може замінити глобальний годинник у певних ситуаціях.

Модель логічних годинників постулює наявність скінченної множини C джерел подій, які звуться логічними годинниками. Вважається, що події від одного джерела $c \in C$ розрізняються тільки номером їх виникнення, тобто кожна подія i у цьому випадку повністю характеризується своїм джерелом і номером виникнення: $i = (c, n) \in C \times \mathbb{N}_+$, де $\mathbb{N}$ позначає множину натуральних чисел з нулем, а $\mathbb{N}_+$ – множину позитивних натуральних чисел. В цьому контексті елемент c яназивають джерелом i , а n – значенням годинникового лічильника.

Таким чином, множина всіх подій в системі логічних годинників, визначених C , ототожнюється з множиною $I(C) = C \times \mathbb{N}_+$. Також на множині подій визначаються два бінарних відношення: «виникає перед» та «виникають синхронно», для яких використовуються позначення $i < j$ (i виникає перед j) та $i \approx j$ (i та j виникають синхронно). Ці відношення мають задовольняти таким аксіомам:

1. відношення «виникає перед» є відношенням суворого порядку, тобто іррефлексивним і транзитивним відношенням;
2. відношення «виникає перед» задовольняє умові: для будь-яких $c \in C$, $n, m \in \mathbb{N}_+$, з $n < m$ випливає $(c, n) < (c, m)$;
3. для будь-якого $i \in I(C)$, множина $\{j \in I(C) \mid j < i\}$ є скінченною;

4. відношення «виникають синхронно» є відношенням еквівалентності, тобто рефлексивним, симетричним і транзитивним відношенням;
5. ці два відношення («виникає перед» та «виникають синхронно») пов'язані такою залежністю: для будь-яких $i, i', j, j' \in I(C)$, з $i < j$ і $i \approx i'$ та $j \approx j'$ випливає $i' < j'$.

Поряд з цими відношеннями розглядають також похідне відношення «спричиняє» (тобто є наслідком, позначається $i \ll j$ для i спричиняє j), яке визначається у такий спосіб: для будь-яких $i, j \in I(C)$, $i \ll j$ означає або $i < j$, або $i \approx j$.

Зазначимо, що аксіоми 1 – 4 виключають можливість одночасного виконання $i < j$ і $i \approx j$.

Очевидно, що відношення «спричиняє» є рефлексивним (тобто $i \ll i$ для будь-якої події $i \in I(C)$) і транзитивним (тобто $i \ll j$ та $j \ll k$ забезпечують $i \ll k$ для будь-яких $i, j, k \in I(C)$). Іншими словами, це відношення є квазіпорядком.

Отже, система логічних годинників визначається за допомогою відношення «спричиняє», яке відповідає таким трьом властивостям:

1. відношення «спричиняє» є квазіпорядком (тобто рефлексивним і транзитивним відношенням);
2. відношення «спричиняє» задовольняє такій умові: для будь-яких $c \in C$, $n, m \in \mathbb{N}_+$, з $n < m$ випливає $(c, n) \ll (c, m)$, але $(c, m) \ll (c, n)$ є невірним;
3. для будь-якого $i \in I(C)$, множина $\{j \in I(C) \mid j \ll i\}$ є скінченною.

Тобто ми маємо дві моделі системи логічних годинників: перша визначається множиною $I(C)$ та відношеннями «виникає перед» і «виникають синхронно», а друга – множиною $I(C)$ та відношенням «спричиняє».

Твердження 2.1. Наведені дві моделі є еквівалентними.

Доведення. Оскільки відповідні розмірковування є очевидними, вони тут не наводяться. Єдине, що потребує більш детального опису, – визначення відношень «виникає перед» і «виникають синхронно» в термінах відношення «спричиняє» для більшого розуміння, як ці відношення зіставляються:

- $i < j \triangleq (i \ll j) \wedge \neg(j < i)$;
- $i \approx j \triangleq (i \ll j) \wedge (j \ll i)$.

□

Леслі Лампорт визначив логічний годинник як $\mathbb{N}_+$ -значну функцію t на множині $I(C)$, яка задовольняє таким умовам:

1. для будь-яких $i, j \in I(C)$, $i < j$ забезпечує $t(i) < t(j)$;
2. для будь-яких $i, j \in I(C)$, $i \approx j$ забезпечує $t(i) = t(j)$.

Л. Лампорт запропонував конструкцію можливого логічного годинника, який зараз зазвичай називають годинником Лампорта [43, 47]. Для посилення на цей годинник будемо використовувати позначення lc (Lamport clock), що визначається у такий спосіб:

$$lc(i) = \max(n \in \mathbb{N}_+ \mid j_1 < \dots < j_n < i) + 1 \quad (2.1)$$

Коректність цього визначення забезпечується умовою 3 визначення моделі, що базується на двох визначених вище відношеннях.

Твердження 2.2. Для будь-якого логічного годинника t вірна нерівність $lc(i) \leq t(i)$ для будь-якого $i \in I(C)$.

Доведення. Дійсно, якщо $lc(i) = n$, тоді існують $j_1, \dots, j_{n-1}, i \in I(C)$, такі, що $j_1 < \dots < j_{n-1} < i$. Але визначення логічного годинника тоді забезпечує $t(j_1) < \dots < t(j_{n-1}) < t(i)$, що гарантує $lc(i) \leq t(i)$. □

Для більш точного логічного часу подій необхідно ввести допоміжне відношення на множині подій – «конкурують» (i та j конкурують позначається через $i \parallel j$) у такий спосіб: для будь-яких $i, j \in I(C)$, $i \parallel j$ означає, що $\neg(i < j) \wedge \neg(j < i) \wedge \neg(i \approx j)$.

Також введемо спеціальний клас логічних годинників, який характеризується такою властивістю: годинник t належить цьому класу, якщо

для будь-яких $i, j \in I(C)$, рівність $t(i) = t(j)$ забезпечує $i \approx j$. Позначимо цей клас через R .

Таке твердження є допоміжним.

Твердження 2.3. Для будь-яких $i, j \in I(C)$, таких, що $i \parallel j$, існує логічний годинник t , який задовольняє нерівності $t(i) < t(j)$.

Доведення. Наведемо формулу для такого годинника:

$$t(k) = \begin{cases} \text{lc}(i) + \text{lc}(k) - \text{lc}(j) + 1 & \text{якщо } k > j \\ \text{lc}(i) + 1 & \text{якщо } k \approx j \\ \text{lc}(k) & \text{інакше.} \end{cases}$$

Якщо побудована функція є логічним годинником, то доказуване припущення є вірним. Перевірка властивостей логічного годинника є тривіальним. $\square$

Твердження 2.4 (аналог теореми Шпільрайна [46]). Для будь-яких $i, j \in I(C)$ таких, що $i \parallel j$, існує логічний годинник $t \in R$, який задовольняє нерівності $t(i) < t(j)$.

Доведення. Безпосередньо випливає з теореми Шпільрайна [46] з використанням Твердження 2.3. $\square$

Твердження 2.5. Для будь-яких $i, j \in I(C)$, $i \parallel j$ виконується тоді і тільки тоді, коли існують логічні годинники $t_1, t_2 \in R$, що задовольняють нерівностям $t_1(i) < t_1(j)$ та $t_2(i) > t_2(j)$.

Доведення. Дійсно, якщо $i \parallel j$, то Твердження 2.4 гарантує існування потрібних логічних годинників.

Але навпаки, якщо такі годинники є, то завдяки $t_s(i) \neq t_s(j)$ ($s \in \{1, 2\}$) маємо $\neg(i \approx j)$. Крім того, $i < j$ забезпечило б $t_2(i) < t_2(j)$, а $j < i - t_1(j) < t_1(i)$, що суперечить припущенням, тобто $i \parallel j$. $\square$

Наведені вище результати дозволяють сформулювати висновок про необхідність створення специфічних для задач проектування РСД моделей та методів їх аналізу. Такий висновок з'являється на підставі відсутності глобального часу у таких системах, що є джерелом неузгодженості реплік – за

відсутності механізму глобального впорядкування двох конкуруючих транзакцій є ризик неправильного порядку запису у РСД. Вивчення ж цієї задачі у загальній теорії розподілених обчислень, як показує Твердження 2.5, призводить до необхідності аналізу усіх можливих еволюцій сховища даних, яке є не тільки нескінченним, але й незліченним, тобто не варто очікувати на існування алгоритмічних методів аналізу у загальному вигляді.

2.2 Загальна модель РСД

Для дослідження CAP-гарантій і методів їх забезпечення необхідно мати загальну модель розподіленого сховища, що відображає його структуру, внутрішні процеси та механізми функціонування. Побудові такої моделі присвячений цей підрозділ роботи.

РСД є мережею, кожний вузол якої зберігає копії певної підмножини одиниць даних, що зберігаються у сховищі, і комунікує з іншими вузлами через комунікаційні канали цієї мережі [48]. Отже, загальну модель РСД ми подаємо як кортеж з п'яти основних елементів:

$$(N, L, \partial, D, r), \text{ де} \quad (2.2)$$

N – скінченна множина вузлів у мережі РСД;

L – скінченна множина комунікаційних каналів;

$\partial: L \rightarrow 2^N$ – відображення, що асоціює пару суміжних вузлів мережі з комунікаційним каналом, що їх з'єднує;

D – множина одиниць даних, що зберігаються у сховищі;

$r: D \rightarrow 2^N$ – відображення, що асоціює одиницю даних з множиною вузлів, які зберігають копію цієї одиниці даних.

На Рис. 2.1 наведено діаграму, що ілюструє загальну структурну модель РСД.

Модель описує взаємодію клієнта і сховища як цілісної мережевої структури. Функціонування РСД відбувається у такий спосіб: клієнт відправляє запит до сховища до шлюзу розподіленої мережі, і далі запит

перенаправляється на деякий вузол мережі, що вибирається відповідно до алгоритму маршрутизації, що працює у мережі на момент запиту.

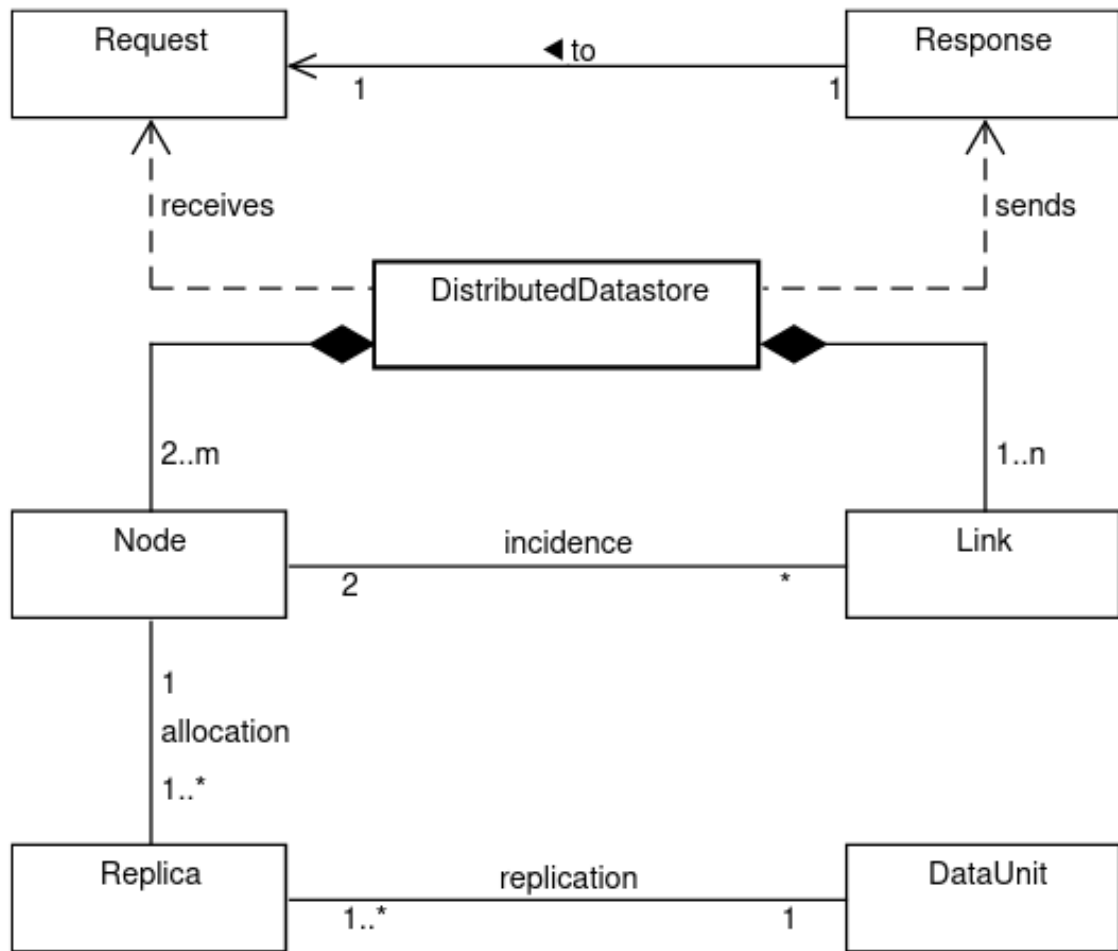


Рис. 2.1. Загальна модель РСД

Після цього відповідний вузол реагує на запит і запускає процес його обробки. Запит до РСД буває принципово двох видів: запит на отримання даних та запит на запис даних до РСД, а дані представляються множиною одиниць даних, копії яких називаються репліками. Кожний вузол зберігає певну підмножину таких реплік. Якщо одиниці даних, що запитується, вузол, на який надійшов запит, не зберігає, то він передає запит далі мережею.

Для передачі даних і запитів використовуються комунікаційні канали мережі, які пов'язують між собою пари вузлів.

Враховуючи складність досліджуваного об'єкту, подальший розвиток методів забезпечення CAP-гарантій потребує розробки більш детальних моделей РСД, що дозволить оперувати описаними властивостями сховищ

цього типу, а саме нам потрібні: модель мережі РСД, моделі централізованого СД і децентралізованого РСД.

2.3 Модель мережі РСД.

Модель мережі РСД у даній дисертаційній роботі є базисом для подальшого розвитку методів забезпечення CAP-гарантій для розподілених сховищ даних.

Відомим фактом є те, що кожна комп'ютерна мережа є набором вузлів, які пов'язані між собою через комунікаційні канали. Вона може бути представлена у вигляді графа [49]. Вузли у даному випадку є вершинами графу, а комунікаційні канали – ребрами.

Отже, математичною моделлю комп'ютерної мережі РСД є простий неорієнтований граф $G=(N,E)$, де елементи множини N – вузли, що відповідають серверам даних, елементи множини E – комунікаційним каналам. Кожне ребро $e \in E$ є двоелементною множиною вузлів, які відповідають двом серверам, що комунікують через відповідний канал, представлений цим ребром.

Зазначимо, що кожний елемент мережі може мати свої унікальні властивості.

Кожний вузол – це сервер, окремий комп'ютер або віртуальна машина, яка може мати певну пропускну здатність, приймаючи і відправляючи певну кількість пакетів, а також не захищений від певних збоїв.

Кожний комунікаційний канал також має певну пропускну здатність – максимальний дозволений розмір пакету, який канал здатен пропустити, середній час, за який пакет доставляється каналом, та інші властивості, що характеризуються числом і певною одиницею виміру. Такі властивості можуть об'єднуватися у загальне поняття – «ціну каналу» («link cost» [50]). З таких числових характеристик, які в програмній інженерії називають метриками, для подальшого дослідження мають найбільше значення саме:

- «ціна каналу» (час доставки пакету і інші складові цієї метрики);

- ймовірність доставки через канал (для поточного дослідження більш доцільно виділити цю характеристику окремо від «ціни каналу»);
- ймовірність того, що вузол здатен коректно оперувати без збоїв у момент часу, коли надходить запит.

Тому до моделі мережі додатково включено три функції:

$$p_a : N \rightarrow [0,1], \quad (2.3)$$

$$p_d : E \rightarrow [0,1], \quad (2.4)$$

$$c : E \rightarrow R_+, \quad (2.5)$$

перша з яких визначає ймовірність доступності відповідного вузла, а друга і третя – відповідно, ймовірність доставки повідомлення через відповідний комунікаційний канал та мережеву ціну цієї доставки.

Зрештою, маємо як модель мережі граф, поданий на Рис. 2.2,

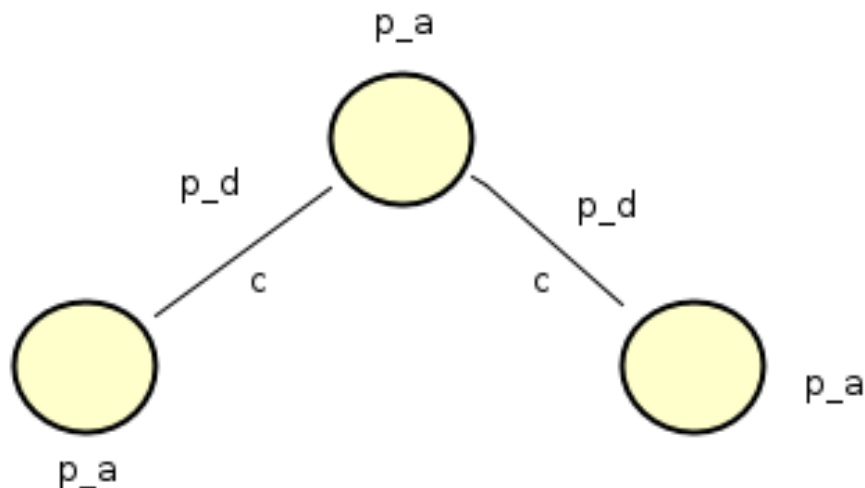


Рис. 2.2. Граф мережі РСД

де p_d – ймовірність доставки через відповідний комунікаційний канал і є атрибутом кожного ребра графу,

p_a – ймовірність того, що відповідний вузол здатний коректно оперувати і є атрибутом кожної вершини графу,

c – ціна доставки через відповідний комунікаційний канал і є атрибутом кожного ребра графу.

У цьому підрозділі також важливо нагадати, які топології розглядаються у цій дисертаційній роботі і чому. Для РСД мережі висока доступність має велике значення, тому вірним є твердження, що зв'язність графу РСД мережі має бути достатньо високою. Зв'язність графу характеризується кількістю сусідів для кожного вузла, що забезпечить варіативність шляхів доставки повідомлення для кожної пари вузлів. Отже, чим більш зв'язний граф, тим більше буде ймовірність доставки для кожної пари вузлів, а, значить, мережа РСД буде мати більш високий показник доступності. Рівень такої зв'язності відноситься і до так званого «коефіцієнту кластеризації», що позначає вузли у графі, що можуть групуватися разом, тобто поєднуватися у кластер, по всьому графу [51, 52]. У цій дисертаційній роботі розглядаються топології доступних мереж, де показник зв'язності для кожного вузла більше одиниці. Під такі характеристики підходять випадкові і регулярні графи, які описані у таких моделях Ердеша-Реньї, Воттса-Строгаца, Барабаші-Альберта [53–55]. Розглянемо ці моделі більш детально.

Основний механізм генерації випадкових графів полягає в обчисленні ймовірності зв'язності кожної пари вузлів за певною математичною формулою або формування зв'язків за певним алгоритмом. Такий підхід дозволяє згенерувати випадковий граф, кожний з вузлів якого буде мати певну ступінь зв'язності, обчислену за поданою у відповідній моделі формулою.

З застосуванням моделі Ердеша-Реньї згенерований граф буде випадковим, але не обов'язково буде слідувати принципам, які спостерігаються у багатьох безмасштабних мережах: коефіцієнт кластеризації буде низьким та не враховуватиме ймовірності утворення нових вузлів і нових зв'язків між ними і створеним графом. То ж така модель підходить для статичної мережі.

Модель Воттса-Строгаца більш сучасна і дозволяє побудувати безмасштабну динамічну мережу з високим коефіцієнтом кластеризації, що підвищує доступність у межах таких кластерів вузлів. Графи, згенеровані за цією моделлю, подано на Рис. 2.3.

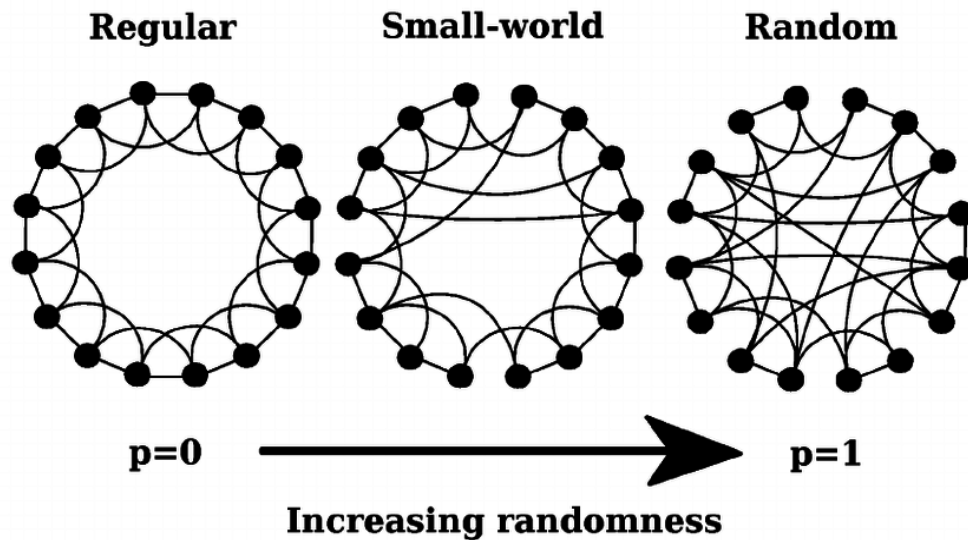


Рис. 2.3. Графи Воттса-Строгаца

Третя відома модель, модель Барабаші-Альберта дозволяє побудувати більш неоднорідну і більш наближену до реальності мережу, але може мати невеликий коефіцієнт кластеризації. Також вона передбачає можливість зростання мережі та додавання вузлів у кластери, де вузли мають високий ступінь зв'язності. Такий принцип називається принципом переважного приєднання [56] та працює і в соціальних мережах, і в деяких загальних мережах-інтернет. Такі граfi можуть бути корисними при формуванні тісно пов'язаних кластерів у деяких розподілених системах. Приклади графів, побудованих згідно моделі Барабаші-Альберта, подані на Рис. 2.4.

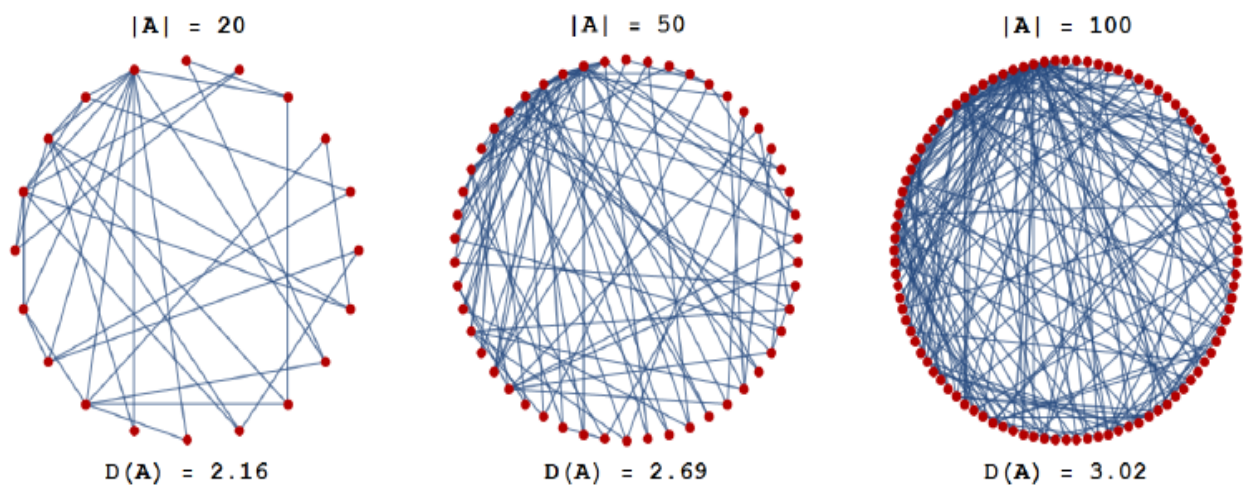


Рис. 2.4. Графи Барабаші-Альберта

2.4 Моделі централізованого СД

Другим важливим компонентом у РСД є дані. Для того, щоб мати уявлення, яку участь вони приймають у механізмах РСД, треба спочатку мати уявлення про структуру даних. Для цього будується початкова модель – модель централізованого СД (сховища даних) і визначаються його властивості. Опис процесів і механізмів дає можливість у подальшому дослідженні розвинути її до моделі децентралізованого РСД, які є актуальними і використовуваними знайшли широке застосування сучасність сучасних інформаційно-комунікаційних системах.

Для опису централізованої моделі управління даними фіксуються дві множини: K – зліченна множина ключів і V – домен (множина можливих) значень.

Станом даних називається скінченна підмножина

$$s \subset K \times V, \quad (2.6)$$

яка задовольняє умові

$$\forall k \in K, \forall v', v'' \in V, (k, v') \in s \rightarrow (k, v'') \in s \rightarrow v' = v'', \quad (2.7)$$

що забезпечує функціональність залежності значення від ключа.

Така скінченна множина s моделює сукупність одиниць даних, кожна з яких є пара (k, v) , де $k \in K$ є ключем, що посилається на значення $v \in V$, і відома в теоретичній інформатиці як «словник» або «асоціативний масив». Зазначимо: ця модель даних ідентифікує одиницю даних (k, v) з її ключем k .

Умова (2.7) входить у базове визначення «словника» [57].

Позначимо через $D(K, V)$ сім'ю всіх словників, ключі яких належать множині K , а значення - множині V . Зрозуміло, що ця сім'я містить порожній словник, тобто словник, що відповідає порожній підмножині множини $K \times V$. Для словників сім'ї $D(K, V)$ можна визначити оператори

$$write: K \times \bar{V} \rightarrow D(K, V) \rightarrow D(K, \bar{V}), \quad (2.8)$$

$$read: K \rightarrow D(K, V) \rightarrow \bar{V}, \quad (2.9)$$

у такий спосіб:

$$\begin{aligned} \text{write}(k, \text{none})s &= s \setminus (\{k\} \times V), \\ \text{write}(k, v)s &= \text{write}(k, \text{none}) \cup \{(k, v)\}, v \in V; \end{aligned} \quad (2.10)$$

$$\begin{aligned} \text{read}(k)s &= v, (k, v) \in s, \\ \text{read}(k)s &= \text{none}, (\{k\} \times V) \cap s = \emptyset. \end{aligned} \quad (2.11)$$

У цих визначеннях через $\bar{V}$ позначено множину значень, розширену елементом *none*.

Наведені вирази демонструють поведінкові аспекти словника, і, відповідно, поведінку даних у СД. Щоб упевнитися у коректності поведінки, доведемо, що оператори задовольняють певним умовам. У даному випадку необхідно вивести умови, які є базою аксіоматики для даної моделі, і у такий спосіб довести, що дані у СД мають задекларовані нижче поведінкові механізми.

Твердження 2.7. Умова атомарності. Після події успішного запису значення v до $D(K, V)$ за ключем k , результатом читання за цим ключем є записане значення:

$$\forall (k, v): \text{write}(k, v): \text{read}(k) = v. \quad (2.12)$$

Доведення. Доведемо від протилежного.

Нехай $\exists (k, v): \neg(\text{write}(k, v) \Rightarrow \text{read}(k))$.

Це можливо у декількох випадках:

- запис був неуспішним;
- не відбулася реєстрація транзакції (визначення транзакції [58]) (тобто факту успішного запису не було);
- запис відбувся за іншим ключем, що має те саме значення $k' = k$.

Першим випадком можна знехтувати, бо для даної моделі, як і для централізованого СД, згідно з принципом «атомності» в моделі ACID, якій має дотримуватись централізоване СД [11], має значення тільки факт закінченого, успішного запису. Друге припущення про запис за ключем $k' = k$ є протиріччям: у словнику кожний ключ обчислюється за допомогою хешової функції, що означає, що кожний ключ є унікальним, тобто вірна нерівність $\text{hash}(k) \neq \text{hash}(k')$ [57]. Як висновок, маємо, що початкове припущення не є

вірним, відповідно, виконується умова (2.12). $\square$

Твердження 2.8. Умова незалежності ключів. Подія запису значення v до $D(K, V)$ за ключем k не тягне за собою зміну значення за ключем k' , тож, після запису за одним ключем, значення за іншим ключем залишається незмінним.

$$\forall(k, k'): write(k, v) \wedge read(k') = v' \Rightarrow read(k') = v' \quad (2.13)$$

Доведення. Доведемо від протилежного. Нехай умова (2.13) невірна, і істинний такий вираз: $\exists(k, k'): write(k, v) \wedge read(k') = v' \Rightarrow read(k') = v''$, що буде означати, що запис за одним ключем дає у результаті запис і за іншим ключем. Випадки, у яких це може бути можливим такі: відбувся паралельний запис за ключем k та k' , де перша транзакція мала вплив на запис іншої; відбувся запис за ключем k' одразу після запису k ; значення за ключем k' після факту запису за ключем k однакові – $v' = v''$; відбувся запис за ключем k , який має такий самий хеш $k' = k''$. Перший випадок неможливий у централізованому СД, тому що згідно з принципом «ізоляції» моделі ACID, якої дотримується централізоване СД [11], транзакції незалежні одна від одної. Другий випадок не є наслідком, а лише двома послідовними подіями, які не можуть мати вплив одна на одну згідно з вже згаданим принципом ізоляції. Третій випадок не є наслідком, бо за будь-яким ключем k' можна прочитати значення v' , доки не відбувся запис $v' \neq v''$. І, нарешті, четвертий випадок неможливий за принципом унікальності ключів словника: $hash(k) \neq hash(k')$. Всі можливі припущення виявились хибними, отже, можна вважати, що умова (2.13) є вірною. $\square$

Тут і далі використовується позначення $\bar{V}$ для посилання на множину значень V , розширену елементом *none*.

Простим, але важливим з точки зору коректності моделі даних є таке твердження.

Твердження 2.9. Результатом застосування оператора $write(k, v')$ для будь-яких $k \in K, v' \in \bar{V}$ до будь-якого словника $s \in D(K, V)$ є словником.

Доведення. Припустимо, що це твердження невірне, тоді

1) або знайдуться $k_0 \in K, v_0 \in V, k \in K$ та $v' \neq v'' \in V$, такі, що $(k, v'), (k, v'') \in s \setminus (\{k_0\} \times V)$;

2) або знайдуться $k_0 \in K, v_0 \in V, k \in K$ та $v' \neq v'' \in V$, такі, що $(k, v'), (k, v'') \in (s \setminus (\{k_0\} \times V)) \cup \{(k_0, v_0)\}$.

У першому випадку маємо, що $(k, v'), (k, v'') \in s$, а значить $v' = v''$, що протирічить припущенню.

У другому випадку маємо, що або $k \neq k_0$ і тоді $(k, v'), (k, v'') \in s$, а значить, $v' = v''$, що веде до протиріччя, або $k = k_0$, і тоді $(s \setminus (\{k\} \times V)) \cap (\{k\} \times V) = \emptyset$, а значить знову є вірною рівність $v' = v''$, що також веде до протиріччя.

Таким чином, припущення невірне і твердження доведено. $\square$

2.5 Модель децентралізованого РСД

Для переходу до моделі управління розподіленими даними враховується, що у децентралізованому РСД одиниці даних розподілені по вузлах мережі як репліки. Це тягне за собою можливість неспівпадіння таких реплік між собою у деяких випадках, коли система дозволяє здійснювати операції запису з будь-якого вузла (хоча деякі бази даних дозволяють конфігурувати певні обмеження, де запис дозволяється тільки з конкретного вузла). Тобто, моделлю розподілу даних у сховищі є відображення $s: N \rightarrow D(K, \bar{V})$.

Однак, ця модель може бути зведена до словника у такий спосіб: ототожнимо відображення $s: N \rightarrow D(K, \bar{V})$ з множиною $S = \{(n, k, v) \in N \times K \times \bar{V} \mid (k, v) \in s(n)\}$.

Твердження 2.10. Множина s є словником з ключами з множини $N \times K$ і значеннями з множини $\bar{V}$. Навпаки, якщо $s \in D(N \times K \times \bar{V})$, тоді $\bar{s}: N \rightarrow D(K, \bar{V})$ є моделлю розподілу даних у РСД.

Доведення. Припустимо, що $(n, k, v'), (n, k, v'') \in s$, і тоді $(k, v'), (k, v'') \in \bar{s}(n)$. Враховуючи, що $\bar{s}(n)$ – словник, за властивостями цієї структури даних про унікальність ключів і значень за цими ключами $v' = v''$. Це протиріччя доводить першу частину твердження. Доведення другої частини є тривіальним. $\square$

Отже як модель децентралізованого РСД маємо відображення $s: N \rightarrow D(K, \bar{V})$, елементи якого можуть бути представлені парою $(n, D(k, v))$, де кожному вузлу відповідає складний об'єкт – словник даних з попередньої моделі, що буде означати, що кожний вузол зберігає таку структуру даних. $\square$

Враховуючи цей результат, далі використовуються словники з сім'ї $D(N \times K, \bar{V})$ як моделі розподілених сховищ даних. При цьому пара $(k, v) \in K \times \bar{V}$, яка задовольняє умові: $\exists n \in N, (n, k, v) \in s$, називається одиницею даних, що зберігає РСД у стані s . Дві такі одиниці даних, які мають спільний ключ $k \in K$, називаються репліками одна одної. Слід зазначити, що визначення операторів *write* і *read* не є тривіальним для РСД. Це, наприклад, можна зробити у такий спосіб:

$$\begin{aligned} \text{write}(k, \text{none})s &= s \setminus (\{(n, k)\} \times \bar{V}), \\ \text{write}(k, v)s &= (\text{write}(k, \text{none})s \cup \{(n, k, v)\}), \end{aligned} \quad (2.14)$$

$$\begin{aligned} \text{read}(k)s &= (n, k, v) \in s, \\ \text{read}(k)s &= \text{none}, (\{n, k\} \times \bar{V})s = \emptyset, \end{aligned} \quad (2.15)$$

де $n \in N$ вибирається довільно.

Проблемою цього визначення є залежність результату виконання операторів від вибору $n \in N$. Тому слід зазначити, що будь-яке визначення операцій, узгоджене з їх визначенням для централізованих сховищ даних, веде до неоднозначності, яка називається неузгодженістю (*inconsistency*). Для уникнення неузгодженості вводиться вимога, оформлена як таке твердження.

Припустимо, що поточним станом РСД є $s \in D(N \times K, \bar{V})$. Будемо казати, що S узгоджено за ключем $k \in K$, якщо для будь-яких вузлів

$n' \neq n'' \in N$ і значення $v \in V$ умови $(n', k, v) \in s$ і $(n'', k, v) \in s$ виконуються чи не виконуються одночасно. Стан S РСД називають узгодженим, якщо він є узгодженим за всіма ключами.

Позначимо множину узгоджених станів РСД через $D \subset (N \times K, \bar{V})$. Зрозуміло, що для узгодженого стану РСД результат виконання оператора *read* не залежить від вибору вузла мережі РСД. Опишемо можливі варіанти визначення оператора *write*, для яких узгодженість даних є інваріантом.

$$\begin{aligned} \text{write} : \{(n, k, v) : K \times \bar{V} \rightarrow D(K, V) \rightarrow D(K, V)\}, \\ \text{read} : \{(n, k) \mid (n, k, v) : N \times K \rightarrow D(K, V) \rightarrow \bar{V}\}. \end{aligned} \quad (2.16)$$

У централізованій моделі управління даних є одне джерело істини, тому виконується умова (2.7) у будь-який момент часу. Щоб дослідити виконання умови (2.7) на розподіленій моделі даних, введемо твердження абсолютної узгодженості у РСД. Репліки на вузлах узгоджені повністю, коли на будь-який запит на одиницю даних отримані значення співпадають:

$$\begin{aligned} C = \forall n \in N, \forall k \in K, \forall v', v'' \in \bar{V}, \\ s_n(k, v') \in s \wedge (k, v'') \in s \rightarrow v' = v''. \end{aligned} \quad (2.17)$$

Таким чином, отримано умову повної узгодженості РСД, але навіть й зараз це не можна назвати «єдиним джерелом істини», бо після будь-якого запису така умова перестане виконуватися.

Однак повна узгодженість у сучасних розподілених системах не має великої ваги, окрім систем, які повинні покладатися на стопроцентну узгодженість. Проте й для таких систем винаходять методи для забезпечення вимог до якості даних без виконання повної узгодженості. Однак, щоб мати можливість вимірювати узгодженість, треба визначити її верхню і нижню межі. Тобто, в цьому випадку верхньою межею узгодженості є умова (2.17), а нижньою – така:

$$\begin{aligned} I = \forall n', n'' \in N, n' \neq n'' \rightarrow \\ \forall k \in K, \forall v', v'' \in \bar{V}, \\ s_{n'}(k, v') \in s \wedge s_{n''}(k, v'') \in s \rightarrow v' \neq v''. \end{aligned} \quad (2.18)$$

Формула (2.18) демонструє умову повної неузгодженості, тобто коли на всіх вузлах мережі значення реплік – різні. Тобто, міра узгодженості може

змінюватися в залежності від кількості вузлів, які мають одну й ту ж саму репліку.

Підсумовуючи модель даних децентралізованого РСД, будемо називати ймовірністю стану узгодженості $\Pr(C)$ ймовірність виконання умови (2.17).

На основі визначених теоретичних елементів, опису розподіленої моделі даних та сформованої моделі РСД можна представити базовий метод розповсюдження даних з метою остаточного узгодження даних у РСД, що описує процес розповсюдження реплік, тобто є процесом реплікації.

Важливо підкреслити, що репліки розповсюджуються розподіленим сховищем епідемічним шляхом за допомогою протоколу Gossip [59]. Gossip – це ціла родина епідемічних протоколів, які базуються на спільних концептуальних засадах. Цей підхід представлений у багатьох варіаціях.

Найважливішим фактом є те, що пакети розповсюджуються мережею епідемічно, тобто між сусідніми вузлами: у сховищі кожна репліка одночасно поширюється від вузла до його сусідів. Цей процес називається реплікацією. Тригером реплікації є факт успішного виконання операції *write*. Ця операція у РСД – найскладніша з описаних.

За умови роботи у такий спосіб система «врешті-решт» знову стане узгодженою.

Вирішення конфліктів та забезпечення узгодженості на етапі паралельних запитів на запис на різні сервери забезпечується іншими методами, такими як алгоритми консенсусу та вирішення конфліктів за допомогою часових міток кожної репліки, що інтегруються у систему у вигляді окремих модулів-надбудов над методами реплікації. Метод, що описує процес реплікації, описаний нижче у формі алгоритму, поданого на Рис. 2.5.

Цей метод демонструє загальний підхід до поширення реплік. Тобто, коли запит на запис надійшов до мережі, і транзакція відбулася успішно, вузол, що записав інформацію у вигляді першої репліки, передає записану репліку сусідам.

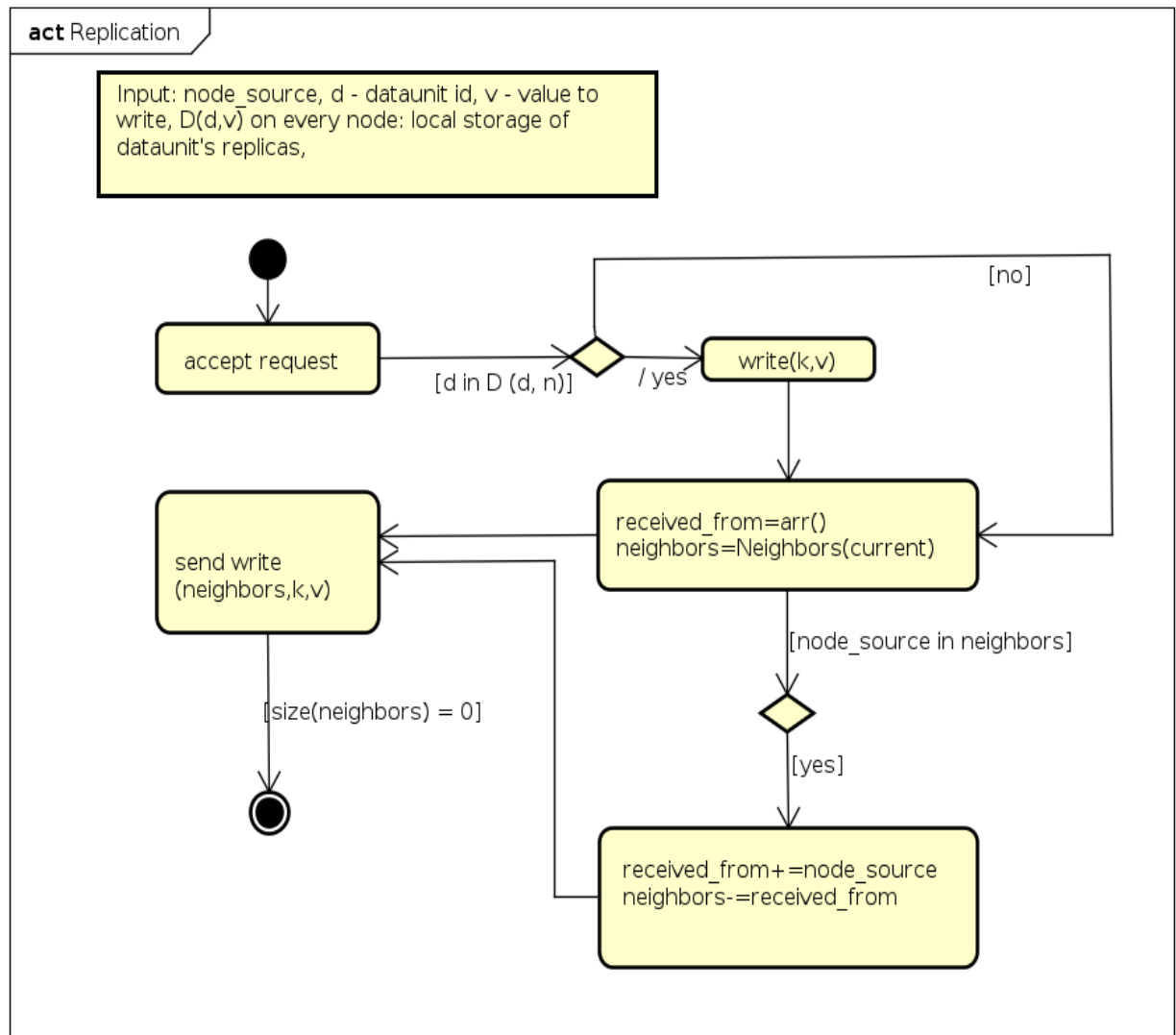


Рис. 2.5. Алгоритм процесу реплікації в РСД

Кожний вузол-сусід, який отримав репліку, записує її до свого локального сховища у разі, якщо така одиниця даних зберігається на цьому вузлу, і передає репліку далі. Доки одна репліка мандрує сховищем, система працює коректно. Але при паралельній обробці запитів на запис на багатьох вузлах, особливо коли розподілена мережа налічує сотні і тисячі вузлів, з'являється багато питань: як вирішувати конфлікти різних реплік, якщо реплікація відбувається з різних місць мережи, як вирішувати конфлікти, якщо репліки були записані одночасно тощо. Кожна репліка може бути позначена часовою міткою, але через неточність локального годинника і відсутність глобального годинника неможливо визначити точний час запису репліки.

Наразі, є дослідження, що пропонують вирішення цих проблем у певній мірі ([23, 60]).

У даній роботі пропонуються моделі годинників, які з застосуванням теорії Леслі Лампорта (див. підрозділ 2.1), допомагають вирішити ці проблеми шляхом побудови такої теоретичної бази, яка дозволяє зрозуміти, що кожному подію можна впорядкувати за часом її створення (і пропонує відповідний алгоритм), навіть у випадку, якщо є події, які відбулися паралельно і навіть мають одну і ту саму часову відмітку. Такі події впорядковуються за умови, що вони не порушують причинно-наслідкові зв'язки і не введення такої впорядкованості не впливає на подальше коректне функціонування системи.

Зазначимо, що на практиці використання розподілених сховищ даних не призводить до виникнення таких реплік, які були б створені у той самий час, і порушували причинно-наслідкові зв'язки.

Слід однак зазначити, що при цьому виникає таке питання щодо втрати системою описаної вище узгодженості, тобто порушення умова повної узгодженості (2.17), після запису репліки до розподіленого сховища даних. Оскільки мережа може налічувати сотні і тисячі вузлів і інтервал між транзакціями може бути невеликим, існує досить велика ймовірність, що система може швидко прямувати до повної неузгодженості (умова (2.18)).

У сучасних системах з великою розгалуженою мережею є нераціональним застосування таких абсолютних значень, оскільки у разі неповної узгодженості є тільки один варіант – чекати, коли система стане повністю узгодженою. У великій системі з розгалуженою мережею, численними запитами на запис, між якими може бути короткий інтервал, повна узгодженість може не відновитися ніколи. То ж дуже важливо мати механізми оцінки ефективності процесу відновлення узгодженості, тобто часу, що необхідний для такого процесу, та створення нових методів, які дозволять системі коректно оперувати даними незважаючи на те, що узгодженість врешті-решт може призводити до сценарію, за яким знайдуться неузгоджені репліки даних у будь-який момент часу [11].

2.6 Висновки за розділом

1. Наведено моделі логічних годинників РСД, які є теоретичним базисом для вирішення конфліктів реплік при паралельному розповсюдженні реплік розподіленим сховищем даних.
2. Побудовано загальну модель РСД, яка описує загальну поведінку і комунікацію між компонентами РСД і надає базу для подальших моделей і досліджень.
3. Розвинено модель мережі РСД, яка дозволяє розробити обчислювальні методи для оцінки узгодженості і процесів реплікації, і на цій основі сформулювати практичні результати у вигляді методів для забезпечення CAP-вимог.
4. Побудовано модель управління даними централізованого сховища даних, що описує структуру даних та їх властивості у сховищі.
5. Модель централізованого сховища даних розвинено до моделі децентралізованого РСД, що демонструє наближені до сучасності поведінку і властивості РСД.
6. Отримані наукові результати викладено у роботах [1, 5, 6].

РОЗДІЛ 3

ОБЧИСЛЮВАЛЬНІ МЕТОДИ ДЛЯ ЗАБЕЗПЕЧЕННЯ CAP-ГАРАНТІЙ

3.1 CAP-теорема як математичний факт

Як вже згадувалось у підрозділі 1.1, масштабування систем призводить до розподілення даних в РСД. Горизонтальне масштабування даних можна виконувати вздовж двох векторів [61]. Функціональне масштабування включає групування даних за функціями і поширення функціональних груп по базах даних. Розбиття даних в межах функціональних областей між декількома базами даних, або шардинг [62], додає другий вимір до горизонтального масштабування. Обидва підходи до горизонтального масштабування можуть застосовуватися одночасно. Таблиці, які мають різне призначення і не пов'язані функціонально між собою, можуть зберігатися на різних вузлах системи. Також вони можуть бути розподілені, якщо між ними немає логічних зв'язків, що означає, що і в майбутньому функціонального зв'язку між цими таблицями не передбачено. Кожна таблиця чи інший об'єкт, який розподілений мережею РСД, називається одиницею даних, а версії, які з'являються на кожному сервері після запису, називаються реплікою – копією одиниці даних. На етапі проектування розробник самостійно визначає, що з себе буде представляти кожна одиниця даних у РСД.

Існуючі підходи до розподілення даних та сам факт розподілення потребують виконання спеціальних властивостей для РСД. Є три основні властивості, які пов'язані між собою і можуть мати певний вплив одна на одну: узгодженість (consistency) – всі репліки у РСД співпадають між собою на всіх вузлах у будь-який момент, доступність (availability) – система, незважаючи на поточне навантаження, може коректно опрацювати запит до моменту, коли вона перестане чекати опрацювання і відповідь помилкою (помилка «тайм-ауту»), стійкість до розділення (partition tolerance) – незважаючи на мережеві розділення система здатна коректно функціонувати і завершувати операції у РСД. Як вже згадувалось, у світі комп'ютерних технологій є відомий

математичний факт, який доведений для певних моделей РСД, що називається теоремою CAP [18].

Теорема CAP. Ерік Брюер, професор з Каліфорнійського університету, Берклі і співзасновник та головний науковий співробітник Inktomi, зробив припущення, що веб-служби не можуть забезпечити всі три властивості одночасно (позначені аббревіатурою CAP): узгодженість, доступність, стійкість до розділень.

Зокрема, веб-додаток може підтримувати не більше двох таких властивостей з будь-яким дизайном бази даних. Очевидно, що будь-яка стратегія горизонтального масштабування ґрунтується на розбитті даних, тому при проектуванні РСД необхідно знаходити компроміс між узгодженістю та доступністю: якщо задля запобігання повної узгодженості потрібно заблокувати вузли, які неузгоджені між собою (тобто ще не мають нової репліки), таке рішення є непопулярним у РСД з великою мережею вузлів.

У даному розділі на основі побудованих математичних моделей у попередньому розділі формуються методи для обчислення і оцінки рівней CAP-вимог в умовах сучасного РСД.

3.2 Загальні метрики для оцінки CAP-гарантій у РСД

У інформатиці загальний підхід вимірювання рівня CAP-гарантій – бінарний: або узгодженість задовольняється повністю, або не задовольняється взагалі, так само з доступністю і стійкістю до розділень. Такий підхід можна вважати застарілим, тому що для сучасних систем ніколи може не виконуватися повна узгодженість чи доступність всіх вузлів, але необхідно мати уявлення про міру узгодженості та міру доступності і стійкості до розділень: на скільки задовольняється та чи інша гарантія. Такі величини можна оцінювати як величину від 0 до 1 або ж як кількісну величину. То ж, у даній роботі пропонується по-іншому поглянути на оцінку CAP-гарантій – пропонується вимірювати рівень CAP-вимог стохастичним шляхом, базуючись на прийнятих визначеннях CAP-вимог у підрозділі 3.1.

Метрика узгодженості. Узгодженість у РСД гарантує, що версії реплік на всіх вузлах співпадають у будь-який момент часу. Нагадаємо, що виокремлюють повну узгодженість, коли задовольняється ця умова, та узгодженість врешті-решт, коли врешті-решт через деякий період часу всі репліки в системі РСД будуть співпадати. Формальне визначення метрики для оцінювання рівня цієї гарантії РСД запропоновано нижче.

Нехай $\sigma_t \in \{0,1\}$ – випадкова величина, яка демонструє одну з таких подій у момент часу $t \geq 0$:

$\sigma_t = 0$ – відповідає події «існує одиниця даних, яка неузгоджена у момент часу t », тобто не виконується умова (2.17);

$\sigma_t = 1$ – відповідає події «всі одиниці даних узгоджені у момент часу t », тобто виконується умова (2.17).

Отже, метрикою узгодженості є ймовірність того що випадкова величина σ_t дорівнює 1, тобто

$$P(C) = \Pr(\sigma_t = 1). \quad (3.1)$$

Метрика доступності. Доступність полягає у тому, що кожен запит повинен отримати коректну відповідь і не відповісти помилкою тайм-ауту, тобто обробити запит за час, не більший від встановленого в конфігурації мережі. Таким чином, є логічним оцінювати доступність як кількісну величину часу. Формальне визначення метрики для оцінювання такої CAP-гарантії для РСД виглядає таким чином.

Нехай τ_t – часовий інтервал між подією отримання запиту у момент часу t та відповіддю на цей запит. Тоді середній час доставки визначається формулою середнього значення (математичного сподівання)

$$T(t) = E[\tau_t]. \quad (3.2)$$

Таким чином, міра доступності РСД буде залежати від кількості вузлів, які виконують запит за період часу, не більший за встановлений у конфігурації системи. Отже міра доступності може визначатися за формулою

$$A = \frac{|\{n \in N \mid T(t)_n < \text{timeout}\}|}{|N|}, \quad (3.3)$$

де N – множина вузлів РСД, а timeout – величина, налаштована в мережі РСД або на кожному вузлу такої мережі [63].

Метрика стійкості до розділень. Стійкість до розділень гарантує, що попри мережеві розділення на дві чи більше ізольовані частини, система спроможна коректно відповідати на запити і оперувати самостійно і незалежно, незважаючи на втрати даних внаслідок розділень.

Нехай $\zeta_t \in \{0,1\}$ – випадкова величина, яка демонструє одну з таких подій у момент часу $t \geq 0$:

$\zeta_t = 0$ – відповідає події «існує одиниця даних, недосяжна з хоча б одного вузла у момент часу t »;

$\zeta_t = 1$ – відповідає події «всі одиниці даних досяжні з будь-якого вузла у момент часу t ». Тоді метрика стійкості до розділень визначається формулою

$$P(t) = \Pr(\zeta_t = 1). \quad (3.4)$$

3.3 Метод обчислення рівня узгодженості

У попередньому підрозділі 3.2 визначено метрики для оцінки рівня CAP-гарантій для того, щоб мати змогу оцінити поточний стан РСД і відповідність CAP-вимогам у певній мірі. Такі метрики, зокрема, метрика узгодженості, надає теоретичний базис для запровадження обчислювального методу, що обчислює рівень узгодженості у РСД як стохастичну величину. Сформуємо таке твердження.

Твердження 3.1. Нехай ϵ РСД з множиною вузлів N . Нехай $N_d \subseteq N$ – підмножина, елементами якої є вузли, що зберігають репліки одиниці даних d , а $N_c \subseteq N_d$ – підмножина вузлів, що мають одну й ту саму версію репліки. Нехай ϵ випадкова подія, коли дві репліки тотожні (узгоджені) між собою за однією одиницею даних; репліки взяті з двох різних вузлів, які є випадковою вибіркою з підмножини N_d ; досліджувана одиниця даних представлена на

розбитті $N = N_1, \dots, N_k$. Будемо називатимемо повною узгодженістю послідовність подій, де на кожній виборці вузлів з розбиття N репліки узгоджені між собою, а повною неузгодженістю – зворотну величину (формули (2.17), (2.18), відповідно). Отже, міра повної неузгодженості визначається такою формулою:

$$I(d) = 1 - \sum_{k=1}^K \frac{N_k(Y_k - 1)}{N(N - 1)}. \quad (3.5)$$

Доведення. По-перше, розглянемо ймовірність випадкової події, коли два випадково взяті вузли мають однакові версії реплік, що означає узгодженість цих вузлів між собою. Розбиваючи цю подію на декілька складових, отримаємо вибір першого вузла випадково з множини N_d та вибір другого вузла з множини N_d з врахуванням того, що вже вибраний попередній. Такі складові основної події залежні між собою. Таким чином, зважаючи на наявні дані для поточної задачі і використовуючи теоретичну базу комбінаторики, можна обчислити ймовірність узгодженості двох вузлів між собою:

$$p_c = \frac{n_c}{|N_d|} \cdot \frac{n_c - 1}{|N_d| - 1} = \frac{n_c \cdot (n_c - 1)}{|N_d| \cdot (|N_d| - 1)}. \quad (3.6)$$

Тепер необхідно обчислити ймовірність загальної узгодженості або загальної неузгодженості вузлів на всьому РСД. Для цього необхідно провести описані нижче вибірки і обчислити ймовірність того, що в кожній виборці два вузли є узгодженими. Для наочності теоретичного доведення розглядається система РСД з п'яти вузлів. З огляду на те що, репліки за однією одиницею даних можуть відрізнятися між собою у різних комбінаціях (однакові репліки на двох, трьох вузлах, три різні репліки на трьох вузлах тощо), такі комбінації розглядаються у вигляді числового розбиття [64]

$$\begin{aligned} &5, & 4+1, & 3+2, \\ &3+1+1, & 2+2+1, & 2+1+1+1, \\ &1+1+1+1+1, \end{aligned}$$

тобто випадок, коли всі 5 вузлів узгоджені між собою, 4 узгоджені між собою і 1 – окремо, і так далі до розбиття, де всі вузли неузгоджені між собою, тобто мають п'ять різних реплік. У кожній з таких комбінацій два вузли будуть узгодженими між собою, якщо вибрані з однієї підмножини, наприклад, у разі розбиття 3+2, кожний з двох вибраних вузлів належить до підмножини з трьома або двома вузлами узгодженими між собою вузлами.

Отже, в описуваному експерименті беруться два вузли з кожної підмножини розбиття і обчислюється ймовірність узгодженості між вузлами для кожного розбиття. Для кожної підмножини розбиття такі події є незалежними, тобто загальна ймовірність узгодженості для розбиття 3+2 буде сумою ймовірностей

$$C(3,2) = p_{c_3} + p_{c_2}, \quad (3.7)$$

і відповідно, ймовірність неузгодженості як протилежна подія має вигляд

$$I(3,2) = 1 - (p_{c_3} + p_{c_2}). \quad (3.8)$$

Отже, з наведених вище пояснень виходить, що узагальнена формула для будь-якого розбиття така:

$$I(d) = 1 - \sum_{k=1}^K p_{i_k}. \quad (3.9)$$

Таким чином, застосовуючи подані вище формули (3.1)–(3.9), отримано такі результати:

$$I(5) = 0$$

$$I(4+1) = \frac{2}{5}$$

$$I(3+2) = \frac{3}{5}$$

$$I(3+1+1) = \frac{7}{10}$$

$$I(2+2+1) = \frac{4}{5}$$

$$I(2+1+1+1) = \frac{9}{10}$$

$$I(1+1+1+1+1) = 1.$$

Грунтуючись на виведених формулах (3.6)–(3.9), отримано загальну формулу ймовірності того, що взяті вузли знаходяться в одній підмножині, для будь-якого розбиття, яка подана як

$$I(d) = 1 - \sum_{k=1}^K \frac{N_k(Y_k - 1)}{N(N-1)}, \quad (3.10)$$

що є загальною мірою неузгодженості для розглянутого РСД. $\square$

Для отримання практичних результатів проведений експеримент з використанням програмних моделей, опис яких надано у підрозділі 4.2.

В цьому експериментальному дослідженні проведено численну кількість ітерацій для РСД з мережею зі 100 вузлів та отримано практичні величини для кожної ітерації, які знаходяться близько величини, яка порахована згідно з формулою (3.10), з невеликою похибкою (Рис. 3.1, Рис. 3.2 та додаток В).

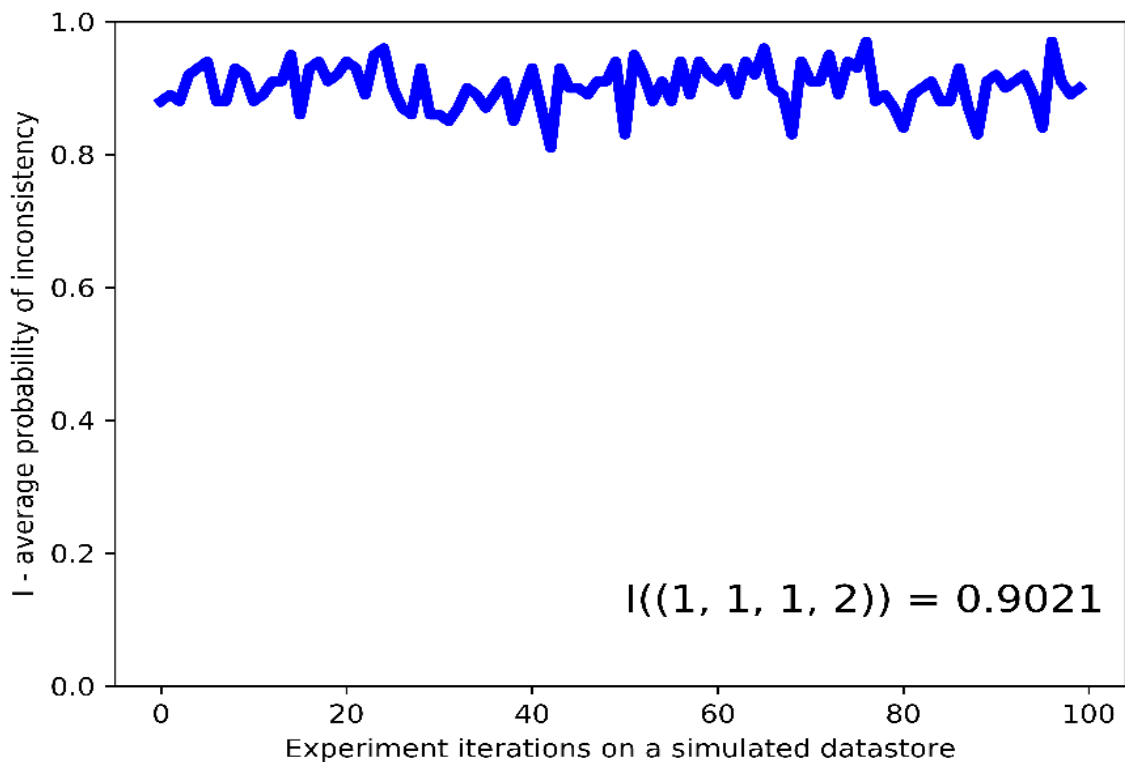


Рис. 3.1. Експеримент узгоджених розділень на РСД з п'ятьма вузлами

Оскільки практичні результати відрізняються від теоретичної оцінки з дозволеною похибкою, ймовірна формула (3.10) міри узгодженості у РСД є математично вірною.

Оцінка рівня узгодженості дозволить проводити моніторинг на вже працюючому для кінцевого користувача РСД та запобігти певних ризиків на

етапі проектування системи, показуючи, як за допомогою експериментів над РСД з певними характеристиками отримати рівень узгодженості, що є актуальним за поточним технічним завданням або специфікацією системи.

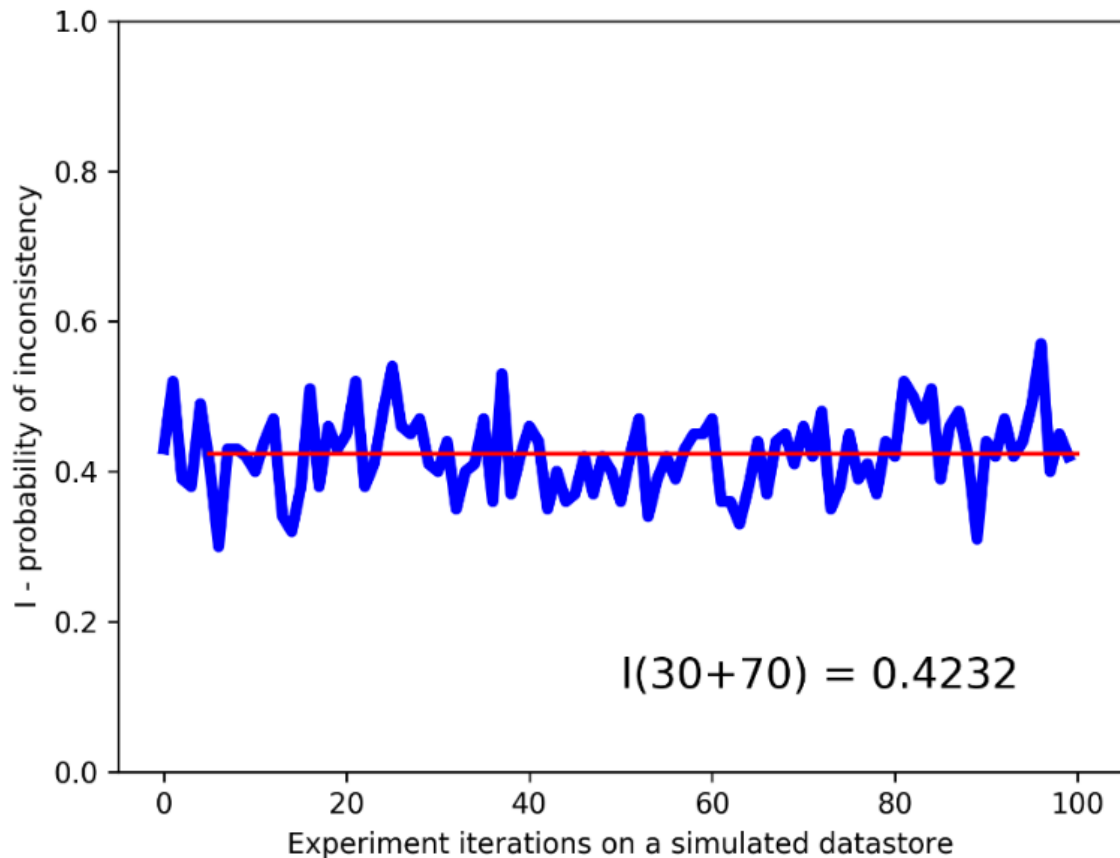


Рис. 3.2. Експеримент узгоджених розділень на РСД із 100 вузлами

3.4 Метод обчислення ціни відновлення узгодженості

Розглядаючи процеси реплікації у розподілених сховищах, необхідно згадати, що у РСД репліки поширюються епідемічним шляхом (за це відповідає у РСД сімейство протоколів Gossip [65]): кожна репліка передається від сусіда сусідам, доки репліка не пошириться всією мережею (див. ілюстрацію такого механізму на Рис. 3.3).

Розглянемо спрощений варіант надходження запитів на запис, коли в системі перебуває не більше однієї репліки, доки репліка не розповсюдиться всім сховищем. У такому разі після події кожної успішної транзакції система

РСД, яка до запису і початку реплікації, була узгодженою, перестає задовольняти цій властивості до моменту, коли процес реплікації завершиться.

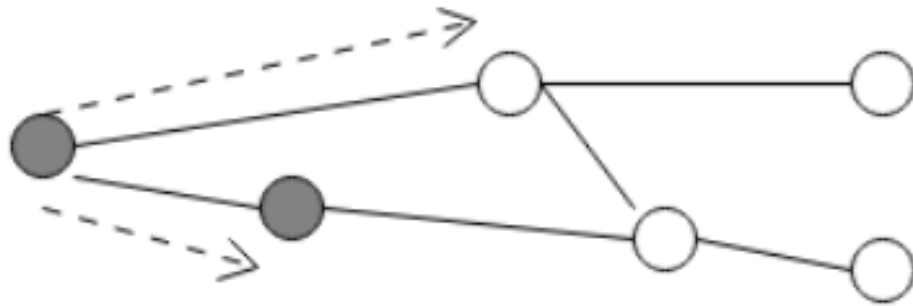


Рис. 3.3. Ілюстрація розповсюдження реплік епідемічними протоколами

Якщо розглядати більш реальні випадки, цілком можлива ситуація, коли у системі під час процесу реплікації розповсюджується більше однієї репліки однієї і тієї ж одиниці даних. Це означає, що процес реплікації навіть за однією одиницею даних цілком можливо буде неперервним, і РСД ніколи не задовольнятиме гарантії суворої узгодженості, особливо у великих мережах РСД, до якої одночасно можуть бути під'єднаними сотні користувачів, або ж багато розподілених процесів можуть покладатися на дані, які зберігаються.

І в системах РСД, які не потребують стовідсоткового гарантування суворої узгодженості, і в РСД, в яких запити на запис не є частими, все ж є ключовим розуміння часового інтервалу, який займає процес реплікації у середньому, або певних критеріїв, які безпосередньо впливають на цей інтервал [66]. Це дозволить оцінювати теоретично час реплікації та покладатися на такі кількісні величини, програмуючи тайм-аут у програмах – величину, яка буде відповідати за те, скільки приблизно треба чекати результату успішного запису всім сховищем, щоб покладатися на записане значення, мати можливість моніторингу такої системи, а також визначати верхню межу середнього часу, який може займати процес реплікації для подальшого проектування процесів реплікації.

Проміжним результатом в процесі такої оцінки є таке твердження.

Твердження 3.2. Нехай у РСД немає втрат даних, та всі зв'язки – стійкі до розділень. Припустимо, що надійшло одне повідомлення, яке є реплікою. Тоді ціна відновлення повної узгодженості буде не більше за діаметр графу, який представляє топологію мережі сховищ.

Доведення. Оскільки повідомлення розповсюджується від сусіда сусідам паралельно згідно епідемічним протоколам [65], через комунікаційні канали, які з'єднують кожні два вузли (див. загальну модель РСД у підрозділі 2.2), і у кожного каналу є ціна доставки повідомлення (такий атрибут введений в моделі мережі у підрозділі 2.3), то ціною розповсюдження репліки є максимальна ціна, яку мають канали, суміжні з вузлами-сусідами. Користуючись моделлю мережі РСД, побудованою у підрозділі 2.3, подано модель мережі РСД у вигляді неорієнтованого зваженого графу [67], в якому вагою кожного ребра є ціна комунікаційного каналу, до якої входить багато параметрів включно з інтервалом часу від прийняття повідомлення до його передачі іншому суміжному вузлу (див. Рис. 3.4).

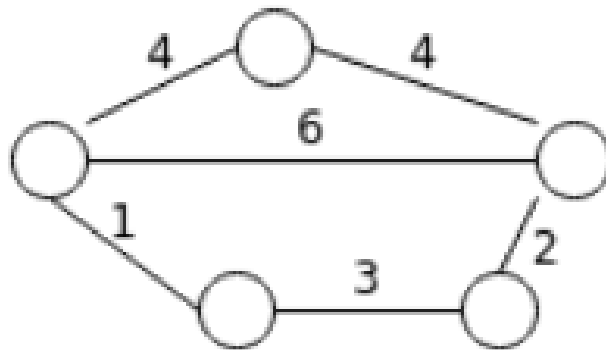


Рис. 3.4. Ілюстрація зваженого графу мережі РСД

Очевидним є те, що у РСД без втрат даних для кожної пари вузлів повідомлення першим прийде до вузла призначення найкоротшим шляхом [68]. Посилаючись на базову термінологію зважених графів в доменній області мереж, найкоротшим шляхом у зваженому графі від однієї вершини до іншої за базовими принципами теорії графів є шлях, сума ваг ребер якого є мінімальною. У найгіршому випадку, реплікація почнеться з вузла, найбільш віддаленого від усіх інших. Оскільки передача даних

впродовж реплікації відбувається для кожного вузла, процес реплікації завершиться, коли репліки розповсюдяться всією мережею, включаючи найвіддаленіший вузол. Найкоротший шлях до найвіддаленішого вузла, тобто «найдовший найкоротший шлях» за базовою термінологією є діаметром графу [67]. Тобто вірність даного твердження витікає з базових визначень компонентів графа. $\square$

Щоб підтримати теоретичний результат практичним експериментом, проведено симуляцію РСД з потрібним процесом розповсюдження одного повідомлення, використовуючи програмну модель, подану у підрозділі 4.3. Один з результатів цього експериментального дослідження подано на Рис. 3.5, інші результати – у додатку Г. Графік демонструє, що для кожного експерименту над РСД з мережею різного діаметру (вісь Y) кількість часових слотів (вісь X) не перебільшує діаметру графу, що представляє РСД. Кожна координата на графіку є одиничною ітерацією експерименту – розповсюдження одного повідомлення сховищем з відповідним діаметром.

Наведене твердження – проміжний результат оцінки реплікації даних. У реальності будь-яка мережа не може бути досліджувана без врахування втрат даних та збоїв вузлів. Також, щоб таке дослідження можна було зробити у контексті стійкості мережі до розділень і доступності, як досліджуваних САР-вимог у цій роботі, це твердження отримало певний розвиток, що наведено нижче.

Користуючись побудованою моделлю мережі у підрозділі 2.3, розглянемо процес реплікації з врахуванням таких доповнених характеристик РСД: p_d – ймовірність доставки комунікаційним каналом, і p_a – ймовірність коректного функціонування вузла. Якщо обидві величини таких ймовірностей дорівнюватимуть 1, це буде гарантією того, що репліка буде доставлена до відповідного пункту призначення. Також важливою величиною є «ціна комунікаційного каналу», яка у наведених нижче математичних формулах фігурує як c (скорочення від «link cost») – метрика, яка поєднує багато величин, досліджуваних у мережі, включаючи час доставки повідомлення [50].

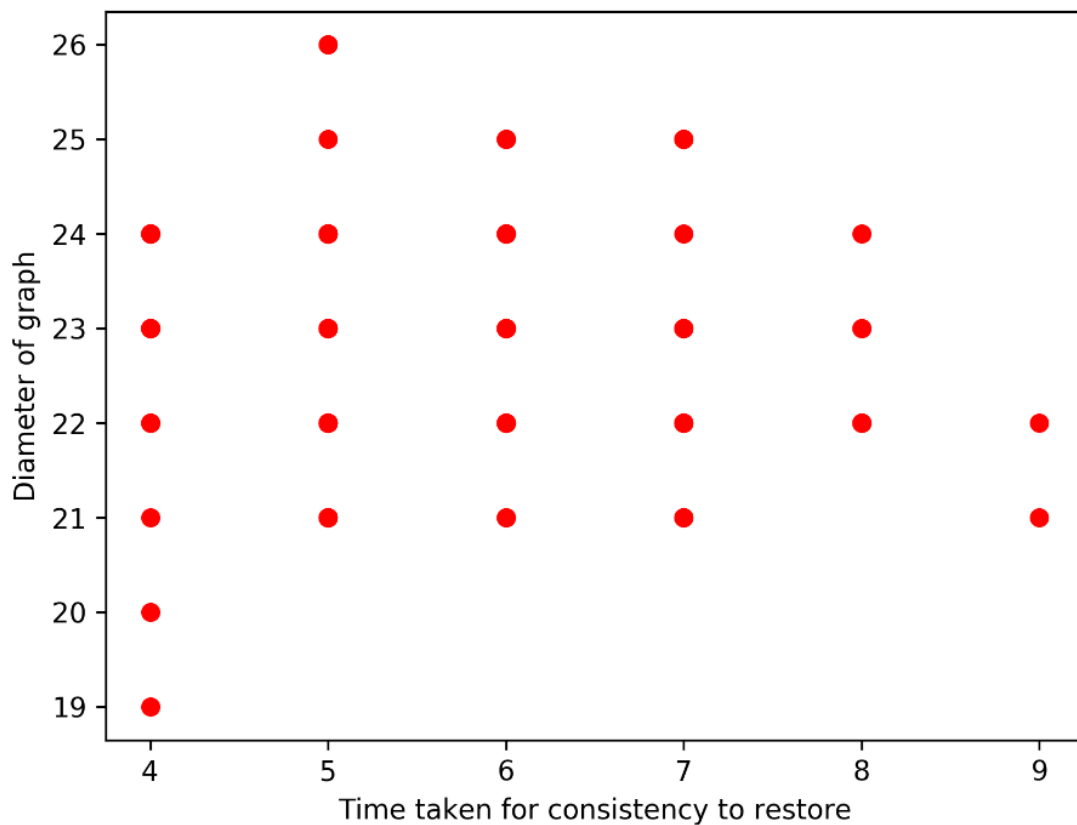


Рис. 3.5. Залежність ціни відновлення узгодженості до діаметра графа

Досліджувати час доставки як окрему характеристику каналу у мережі зі складним механізмом доставки не є раціональним, тому що такі результати дослідження навряд будуть коректними без врахування інших характеристик мережі, які цілком можуть бути пов'язані з часом доставки також.

Отже, обчислимо ціну відновлення узгодженості, тобто ціну процесу реплікації, звужуючи дослідження до оцінки розповсюдження однієї репліки після її первинного запису у РСД. Оскільки репліки можуть розповсюджуватися паралельно, дослідження багатьох реплік не має впливати на час розповсюдження репліки.

То ж, для поточного дослідження згідно побудованій моделі мережі у підрозділі 2.3 компоненти графа, що представляє мережу РСД, мають певні атрибути, а саме:

P_d – атрибут для кожного ребра k у неорієнтованому графі G , ймовірність доставки на комунікаційному каналі, що відповідає ребру k . Ця

величина має значення від 0 до 1, досліджується і обчислюється завчасно до того, як проектувати РСД, згідно метриці «ціна комунікаційного каналу», яка згадувалась раніше;

p_a – атрибут для кожної вершини i у неорієнтованому графі G , ймовірність того, що вузол, що відповідає вершині i , має значення від 0 до 1 та досліджується та обчислюється завчасно до того, як виконувати такі операції;

c – атрибут для кожного ребра k у неорієнтованому графі G , має кількісне значення в інтервалі позитивних натуральних чисел. Це метрика, яка складається з багатьох характеристик, досліджуваних у мережі: ємність («capacity»), пропускна здатність («bandwidth»), оцінка втрат даних (у поточному обчисленні вже описаний вище атрибут p_d), а також досліджуваний час доставки («latency», «delay») та інші характеристики, які інженер конкретної мережі вважатиме за потрібне включити до оцінки.

Важливо зазначити, що, оскільки обчислюється ціна доставки за умови втрат даних, у подальших формулах більш раціонально використовувати таку характеристику як втрати даних окремо, оскільки ця величина буде приймати участь у окремих формулах, без інших характеристик, що входять до складових «ціни комунікаційного каналу».

Отже, запропонуємо таке твердження і доведемо, що воно є вірним, користуючись базисом теорії графів та теорії ймовірності.

Твердження 3.3. Нехай є довільний неорієнтований граф G , множина його шляхів $P = P_1, \dots, P_k, \dots, P_n$. Середні ціни доставки для кожної пари i, j визначаються такою формулою:

$$\tau_{ij} = \sum_{k=1}^n c_k \cdot \left(1 - \frac{p_k}{p(\text{delivery})}\right). \quad (3.11)$$

Доведення. Візьмемо будь-яку пару вершин (вузлів) i, j . Шлях від i до j $P_k \in P = \{P_1, \dots, P_k, \dots, P_n\}$, де P – множина всіх шляхів від i до j у графі G . Обчислимо ймовірність доставки по будь-якому з таких шляхів.

Будь-який шляху графу складається з n вершин і ребер, які з'єднують дві вершини між собою: $P_k = \{e_{i,i+1}, e_{i+1,i+2}, \dots, e_{j-1,j}\}$. Оскільки доставка по ребру $e_{i+1,i+2}$ неможлива без доставки ребром $e_{i,i+1}$, логічним висновком є такий, що події доставки кожним ребром залежні між собою: доставка шляхом P_k відбудеться тільки тоді, коли вона відбудеться і ребром $e_{i,i+1}$, і ребрами $e_{i+1,i+2}, \dots, e_{j-1,j}$, тож ймовірність доставки будь-яким шляхом з множини P визначається формулою

$$p_k = p_{e_{i,i+1}} \cdot \dots \cdot p_{e_{k,k+1}} \cdot \dots \cdot p_{e_{j-1,j}} = \prod_{k=i+1}^j p_{e_{i,k}}. \quad (3.12)$$

Але доставка також не відбудеться, якщо вузол не зможе прийняти повідомлення, оскільки вузол є посередником, що передає повідомлення далі іншим ребром. Враховуючи ймовірність його коректного функціонування p_a для кожного вузла від i до j , тепер доставка шляхом P_k визначається формулою

$$p_k = p_{e_{i,i+1}} \cdot p_{a_{i+1}} \cdot \dots \cdot p_{e_{j-1,j}} \cdot p_{a_j} = \prod_{k=i+1}^j p_{e_{i,k}} \cdot p_{a_{k+1}}. \quad (3.13)$$

Отже, маємо доставку будь-яким шляхом з множини P у загальному випадку.

Зрозуміло, що ймовірність того, що повідомлення не буде доставлене – зворотна величина, тобто ймовірність втрати даних на будь-якому шляху P_k :

$$p_{dataloss}(k) = 1 - p_k = 1 - \prod_{k=i+1}^j p_{e_{i,k}} \cdot p_{a_{k+1}}. \quad (3.14)$$

Оскільки повідомлення може бути доставлене через будь-який інший шлях, якщо на якомусь P_k відбулася втрата даних, то необхідно обчислити ймовірність такої ж втрати даних всіма іншими шляхами з множини P . Кількість шляхів у цій множині залежить від зв'язності графа [67].

Отже, оскільки така подія загальної втрати даних станеться тільки тоді, коли доставка репліки не відбудеться жодним шляхом, формула загальної втрати даних від i до j така:

$$p_{data\,loss}(i, j) = \prod_{k=1}^n p_{data\,loss}(k) = \prod_{k=1}^n \left(1 - \prod_{m=i+1}^j p_{e_{i,m}} \cdot p_{a_{m+1}} \right). \quad (3.15)$$

То ж доставка відбудеться у протилежному випадку – коли подія загальної втрати даних не відбудеться. Тоді зрозуміло, що ймовірність доставки загалом від i до j

$$p_{delivery}(i, j) = 1 - \prod_{k=1}^n p_{data\,loss}(k) = 1 - \prod_{k=1}^n \left(1 - \prod_{m=i+1}^j p_{e_{i,m}} \cdot p_{a_{m+1}} \right). \quad (3.16)$$

Після таких обчислень можна сформулювати такі дві події:

А) доставка відбулась будь-яким шляхом P_k ,

В) доставка відбулась загалом від i до j .

Подія А неможлива без того, щоб відбулася подія В, оскільки доставка якимось шляхом відбудеться тільки тоді, коли вона можлива взагалі. Отже, маємо фіксований ймовірнісний простір $\Omega = \{A, B\}$. Виходячи з визначення умовної ймовірності

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad (3.17)$$

з наведеного випливає, що формула доставки шляхом P_k дорівнює

$$p(i, j) = \frac{P_k}{p_{delivery}(i, j)}, \quad (3.18)$$

а отже,

$$p(i, j) = \frac{\prod_{k=i+1}^j p_{e_{i,k}} \cdot p_{a_{k+1}}}{1 - \prod_{k=1}^n \left(1 - \prod_{m=i+1}^j p_{e_{i,m}} \cdot p_{a_{m+1}} \right)}. \quad (3.19)$$

Повертаючись до кінцевого припущення про середню ціну доставки і спираючись на те, що у підрозділі 3.2 пропонувалося кількісні величини середнього часу (метрика доступності) оцінювати як математичне середнє у формулі (3.2), та посилаючись на визначення математичного середнього, де ми усереднюємо значення ціни ймовірністю втрати даних, маємо

$$\tau_{ij} = \sum_{k=1}^n c_k \cdot (1 - p(i, j)) = 1 - \frac{\prod_{k=i+1}^j p_{e_{i,k}} \cdot p_{a_{k+1}}}{1 - \prod_{k=1}^n \left(1 - \prod_{m=i+1}^j p_{e_{i,m}} \cdot p_{a_{m+1}} \right)}, \quad (3.20)$$

а значить, формула

$$\tau_{ij} = \sum_{k=1}^n c_k \cdot \left(1 - \frac{p_k}{p(\text{delivery})} \right) \quad (3.21)$$

є вірною.

Обчислення для кожної пари дасть як результат квадратну симетричну матрицю T

$$T = \begin{pmatrix} \tau_{11} & \cdots & \tau_{1n} \\ \vdots & \ddots & \vdots \\ \tau_{n1} & \cdots & \tau_{nn} \end{pmatrix}, \quad (3.22)$$

де кожний елемент обчислюється за формулою (3.21). Це математичне доведення ґрунтується на базі теорії ймовірності [69–70]. $\square$

Максимальне зі всіх значень матриці і буде загальною ціною реплікації, у цьому разі це не обов'язково буде відстань між найвіддаленішими вузлами, тому що втрати даних на будь-яких вузлах і комунікаційних каналах корегують ситуацію, і найшвидша, і найповільніша доставка може виникнути на будь-якому шляху для будь-якої пари (i, j) .

3.5 Метод контрольованого балансування узгодженості

У підрозділі 3.3 наводились формули для обчислення рівня узгодженості у РСД. Це може сприяти розвитку системи моніторингу узгодженості у працюючому РСД, але також може стати основою для методів забезпечення CAP-гарантій: коли відома така величина як середній рівень узгодженості для конкретної системи, є знання, чи потрібно застосовувати певні методи, що допоможуть забезпечити узгодженість у більшій мірі.

Наразі у комп'ютерних науках нема механізму оцінки рівня узгодженості як шкали або ймовірнісної міри: наскільки задовольняється узгодженість у тому чи іншому РСД. Це може залежати від частоти надходжень запит на запис, які спричиняють неузгодженість, і цілком

можлива ситуація, коли система ніколи не буде узгоджена повністю, особливо у розподілених сховищах з великою мережею.

То ж, маючи метрики для оцінювання поточного рівня узгодженості і можливість оцінювати узгодженість не бінарним, а статистичним шляхом, з'являється підстава для методу моніторингу та регулювання рівня узгодженості у РСД.

Наразі таке рішення не реалізовано у розподілених сховищах. У цій дисертаційній роботі наводяться методи, доведені до алгоритмічної реалізації, протестовані за допомогою програмної симуляції, та проведено теоретичну та практичну оцінку просторової та часової складності таких реалізацій.

Отже, справа в тому, що наразі у процесі відповіді на читання зазвичай вузли відповідають доступною реплікою, і цілком можливо, що ця репліка буде неузгоджена з більшістю інших, які вже розповсюдилися мережею. Але оскільки процес реплікації відбувається епідемічним шляхом, є висока ймовірність, що система не втратить доступність, якщо намагатися відповідати з угоджених між собою вузлів – тих, які вже отримали нову репліку. Таке рішення можна прийняти на будь-якому етапі реплікації залежно від того, який рівень доступності потрібен за технічною специфікацією системи чи рішення користувача РСД – скільки вузлів повинні отримати репліку, щоб саме вони вважалися узгодженими між собою (більшість вузлів, третя частина мережі і т.п.).

Для розробки таких методів було проаналізовано найбільш використовувані рішення баз даних з відкритим кодом (MongoDB, PostgreSQL, Google Spanner), які мають функцію розподілення даних на кластери. Було визначено, що наразі використовуються інші компромісні рішення, найпопулярніше з них таке: згідно рівням ізоляції транзакції [71] дозволяється читати нову версію репліки на різних етапах виконання транзакції, або ж на різних етапах реплікації, як у Google Spanner. Досліджені можливості інтеграції і раціональної співпраці з існуючими механізмами розглянутих баз даних з функцією розподілення.

Отже, почнемо із структури компонентів, які має система РСД у розробленому методі контрольованої узгодженості, дамо більш детальний опис частини, яка відповідає за обробку запитів.

На Рис. 3.6 демонструється діаграма компонентів розподіленого сховища. На діаграмі наявні основні компоненти реалізації методу. Поговоримо окремо про кожний.

Шлюз (gateway). Розподілена система для користувача – чорний ящик і повинна такою бути. Робота з розподіленою базою даних починається з запиту у РСД у певному форматі, узгодженому з конкретним рішенням СУБД, яка використовується. Кінцевому користувач не повинно бути відомо, на який сервер приходить запит, як він обробляється, які операції проводяться для того, щоб обробити його запит. Користувача цікавить тільки час відповіді і коректність поверненої відповіді. Шлюзом може виступати як окремий сервер, який перенаправляє запити у внутрішню систему за найпростішим алгоритмом, так і так званий балансувальник навантаження, який за певним алгоритмом переправляє запит на той чи інший вузол. Також шлюзом може бути вузол самої системи – тоді він буде мати відповідний мережевий інтерфейс, щоб мати змогу приймати запити.

У сучасному світі технологій алгоритми балансувальників розмножилися і можуть працювати за різними принципами. Навіть ті веб-сервери, які раніше просто переправляли запити за найпростішими алгоритмами, розвинули підходи до розподілення запитів між серверами (наприклад, Nginx, Apache [72, 73]). Також, у деяких рішеннях підтримується балансування у БД. Більш детальний аналіз існуючих механізмів, які теоретично або практично сумісні з інтерфейсами баз даних, надано у підрозділі 1.4.

Вузол (node). Це ядро внутрішнього механізму всієї системи РСД, компонент розподіленої мережі, сервер, який приймає запити у РСД від сервера-шлюзу.

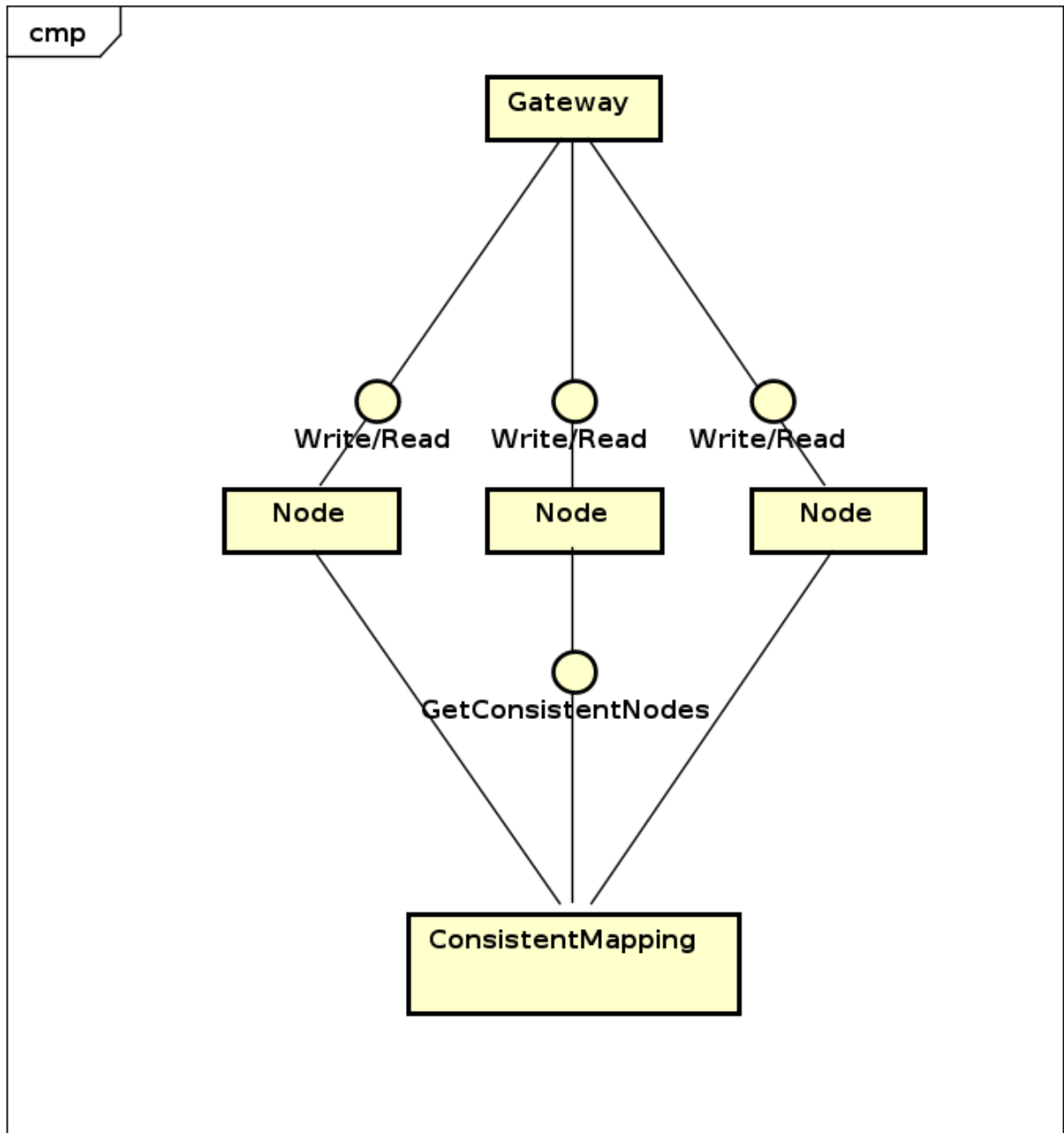


Рис. 3.6. Діаграма компонентів контрольованого балансування узгодженості

Вузол виконує основну роботу у системі РСД – займається обробкою запиту, який реалізований за певним алгоритмом залежно від використовуваної СУБД. У багатьох СУБД основний принцип обробки однаковий, але є істотні відмінності у певних деталях, що можуть впливати на вибір того чи іншого рішення розподіленої бази даних [74, 75].

У цьому підрозділі пропонуються модифіковані методи обробки запитів і процесів реплікації, які дадуть змогу забезпечувати узгодженість і доступність на рівні згідно потреб конкретної системи РСД шляхом контролю

балансування між доступністю і узгодженістю, і посприяє тому, що отримати нові репліки, які вже можна вважати узгодженими, раніше, ніж завершиться процес реплікації.

Наступний компонент є елементом новизни, оскільки в пропонуваному алгоритмі він є основним сполучником у процесі забезпечення балансування узгодженості.

Узгоджене відображення (Consistent Mapping). У підрозділі 2.5 побудовано модель управління даними в РСД. Доведено, що структура даних може представлятися «відображенням» (у комп'ютерних науках «словником» або «хеш таблицею»). У базах даних структура частіше за все є спеціальною таблицею, але досі відповідає заданій моделі і зберігає її властивості, доведені у тому ж підрозділі. Для того, щоб система мала глобальне знання про те, які вузли узгоджені між собою, створюється окремий компонент – відображення узгоджених вузлів, яка описується як $D \rightarrow N$, тобто множина пар $(dataunit, nodes)$, що складаються з ключа та значення. Ключом у такій парі є $dataunit \in D$ – одиниця даних, подана певним об'єктом, один з атрибутів якого – часова мітка, а другий атрибут – ідентифікатор одиниці даних, який однозначно ідентифікує і репліку – її копію (див. загальну модель РСД у 2.2). Значенням у згаданій вище парі $nodes$ є масив ідентифікаторів вузлів, які мають одну й ту саму репліку, тобто є узгодженими між собою. Цей компонент як узгоджене відображення може зберігатися у будь-якому високо доступному місці, бути централізованою хеш-таблицею (яку також називають у комп'ютерних науках словником [57]) на кластері вузлів, або ж певні вузли у системі РСД можуть виконувати роль такого сховища. Підхід призначення вузлам мережі певних ролей не новий, такий підхід використовує і відомий протокол консенсусу Paxos (і інші протоколи з сімейства протоколів Paxos [76] та інші протоколи консенсусу [77, 78]), де вузли в мережі можуть виконувати різні ролі, бути відповідальними за певні рішення згідно протоколу. У останньому випадку це може бути реалізовано як вузли, що сусідні між собою

у мережі РСД, так і як окремі групи кластерів, які забезпечують узгодженість на певну частину мережі.

Останнє рішення може бути корисним, коли РСД мережа умовно поділена на різні зони за принципом геолокації серверів, частоти запитування певних типів одиниць даних і за іншими принципами, які можуть бути раціональними для технічного завдання конкретного програмного забезпечення. Наприклад, найбільш популярні хмарні провайдери (Google Cloud Platform [79], Amazon Web Services [80], Microsoft Azure [81]) реалізують функцію поділення на регіони і доступні зони за різними принципами (найчастіше це географічне положення конкретного дата-центру, де знаходяться сервери провайдера) [82, 83]. Таке розподілення цілком виправдане: наприклад, є логічним висновок, що частина розподіленої мережі, яка знаходиться в Австралії, рідше потрібна розробнику з України, ніж дата-центр поблизу України – в Чехії. Звісно, до даних, які зберігаються у більш віддалених куточках світу, теж має бути доступ, але пріоритетом все ще мають бути більш близькі сервери. За таким принципом можуть призначатися і ролі певним серверам або кластерам серверів, залежно від функціоналу, який має реалізувати програмне забезпечення, що використовує таку розподілену систему. У РСД можливість такої гнучкої конфігурації системи є найбільш раціональним рішенням. У цій дисертаційній роботі досліджено декілька варіацій реалізації методу контрольованого балансування узгодженості для того, щоб зробити попередню оцінку ефективності. На базі оцінок таких реалізацій на етапі проектування і налаштування РСД для відповідного програмного забезпечення архітектор бази даних і розробник програмного забезпечення матимуть змогу прийняти рішення, яка конфігурація РСД з інтегрованим методом контрольованого балансування узгодженості більше підходить для поточного технічного завдання.

Отже, спершу розглянемо сам метод контрольованого балансування узгодженості. Такий метод складається з двох фаз: перша – для запису даних, яка ініціює процес, який наповнює «узгоджене відображення», описане вище

і тим самим дозволяє контролювати узгодженість на заданому рівні на етапі запиту на читання; друга фаза – для відповіді на запит на читання, де здійснюється власно балансування між узгодженими вузлами, якщо виконуються певні умови. Перша частина цього методу, подана нижче, ініціює процес розповсюдження даних, тобто реплікацію.

Метод відповіді запиту на читання.

Загальна мета алгоритму – записати нове значення у систему РСД.

У алгоритмі використовується словник даних, властивості якого описані у моделі управління даними децентралізованого РСД у підрозділі 2.5. Алгоритм подано у формі псевдокоду на Рис. 3.7.

Вхід:

`nodesource` – вузол, з якого надійшов запит
`d` – одиниця даних, `v` – нове значення, `N` – список вузлів
`D(d, v): D(d)=v` – відображення даних, що зберігаються
`CD(D, N):CD(d)=arr(N)` – узгоджене відображення

Repeat

```

    receivedfrom = empty array
    neighbors = neighbors(nodecurrent)
    consistent_nodes = CD(d)
    if k not in D (d, v):
        send write(neighbors, d, v, nodesource); continue

    write(d, v)
    if nodesource in neighbors and d.ts > CD.dataunit.ts:
        CD(d) = arr(N) + nodecurrent
    if nodesource not in neighbors and d.ts < CD.dataunit.ts:
        CD(d) = nodecurrent
    add nodesource to receivedfrom
    neighbors -= receivedfrom
    send write(neighbors, d, v)
until size of neighbors is 0

```

Рис. 3.7. Алгоритм обробки запиту на запис у формі псевдокоду

Основний механізм алгоритму такий: коли вузол отримав нове значення до запису, після факту успішного запису у локальне сховище, вузол визначає, звідки надійшло значення. Якщо значення надійшло з внутрішньої системи, це

означає, що вузол має доповнити своїм ідентифікатором список узгоджених вузлів за відповідною одиницею даних у узгодженому відображенні, у разі, якщо часова мітка (яка відповідає за версію репліки) більша за часову мітку в відповідному значенні одиниці даних.

Якщо ж значення надійшло від клієнта, відповідний список узгоджених вузлів спустошується і наповнюється поточним вузлом у разі, якщо в узгодженому відображенні нема більш нової репліки за такою одиницею даних. Це перевіряється шляхом порівняння часових міток в одиниці даних, що зберігається. Часові мітки репліки та одиниці даних можуть визначатися згідно моделі годинників, що описана у підрозділі 2.1.

Таким чином, компонент системи, що зберігає узгоджене відображення, буде мати єдине істинне значення репліки у момент часу t .

Даний метод також подано у формі діаграми активності для наочності на Рис. 3.8. Така форма має бути більш зрозуміла на етапі проектування, коли попередня схема буде корисною на етапі реалізації програмного коду.

Цей метод дозволить мати у системі знання про єдину істинну версію репліки на даний момент часу. Таке відображення буде зберігати список ідентифікаторів вузлів, які мають одну й ту саму версію репліки, найновішу на даний момент.

Саме таке сховище і буде узгодженим відображенням, опис якого подано вище.

У кожній конкретній системі РСД на етапі проектування і вибору рішення для узгодженого відображення необхідно зробити потрібні заміри, скільки часу займає та чи інша операція взаємодії з таким рішенням і вибрати те, яке найбільш підходить під специфікацію системи.

Існує багато реалізацій такого сховища, де кожному ключу відповідає певне значення, наприклад, так рішення як Redis, Memcache та ін.

Вибір таких рішень та оцінка їх операцій вставлення та читання виконуються окремо на етапі вибору «key-value» сховища на етапі інтеграції цього алгоритму у відповідне рішення РСД.

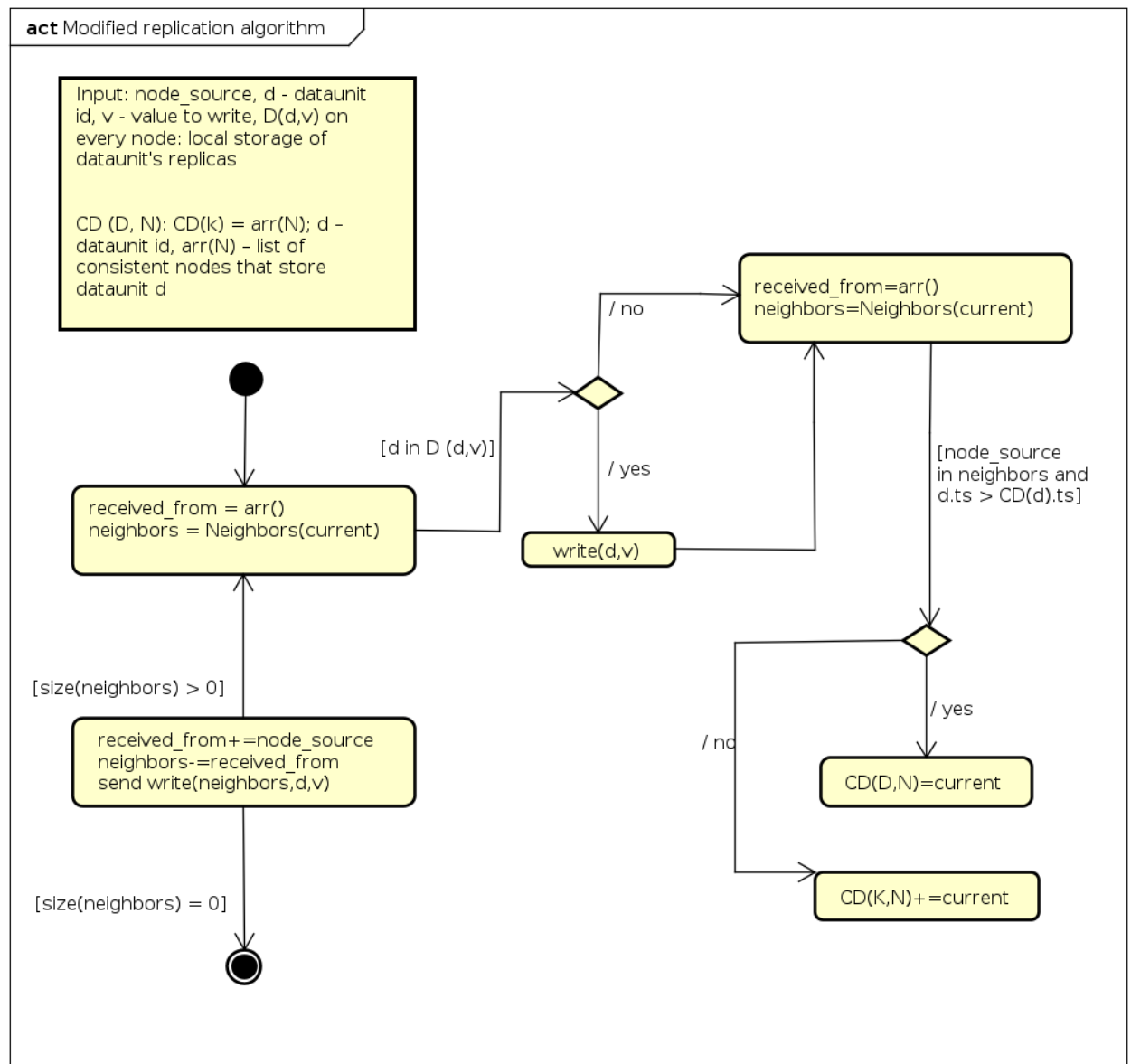


Рис. 3.8. Удосконалений алгоритм процесу реплікації. Діаграма активності

Друга фаза методу контрольованого балансування узгодженості повинна бути інтегрована з методом відповіді на запит на читання.

Ця частина методу подана у формі псевдокоду на Рис. 3.9 та у формі діаграми активності на Рис. 3.10 для повного розуміння механізму роботи такого алгоритму.

Отже, механізм алгоритму такий: у систему перед початком її роботи додається величина у вигляді сконфігурованого значення, що відповідає за мінімальну кількість вузлів, які мають бути доступними у будь-якому разі.

Вхід:

$node_{source}$ – вузол, з якого надійшов запит

d – одиниця даних, v – нове значення, N – список вузлів

$D(d, v)$: $D(d)=v$ – відображення даних, що зберігаються

$CD(D, N)$: $CD(d)=arr(N)$ – узгоджене відображення

$threshold$ – значення порогу – мінімальна кількість вузлів, яка повинна бути завжди доступною, конфігурується у РСД.

Repeat

$received_{from}$ = empty array

$neighbors$ = $Neighbors (node_{current})$

$nodes_{consistent}$ = $CD(d) \rightarrow arr(n)$

if d in $nodes_{consistent}$:

 return read (d) $\rightarrow v$; continue

if length($nodes_{consistent}$) > threshold:

 send read($node_{consistent}$)

else if d in $D (d)$:

 return read(d) $\rightarrow v$

else:

 send read($neighbors$)

 add $node_{source}$ to $received_{from}$

$neighbors -= received_{from}$

until size of $neighbors$ is 0

Рис. 3.9. Алгоритм відповіді на запит на читання у формі псевдокоду

Це поріг, за який непотрібно переходити для збереження раціонального навантаження на вузли у системі. У іншому разі у системі РСД знизиться рівень доступності, і вже не можна буде мережу назвати високо-доступною (більше інформації про високо-доступні (Highly Available) системи див. [13]).

Також, за рахунок роботи першої фази методу контрольованого балансування узгодженості, маємо можливість отримати список узгоджених вузлів.

Список узгоджених вузлів отримується в компоненті узгодженого відображення, роль якого може виконувати будь-який вузол у системі (підсистемі) або окремий сервер (або невеликий кластер) глобального сховища, де кожній одиниці даних відповідає список вузлів, які зберігають узгоджені між собою репліки.

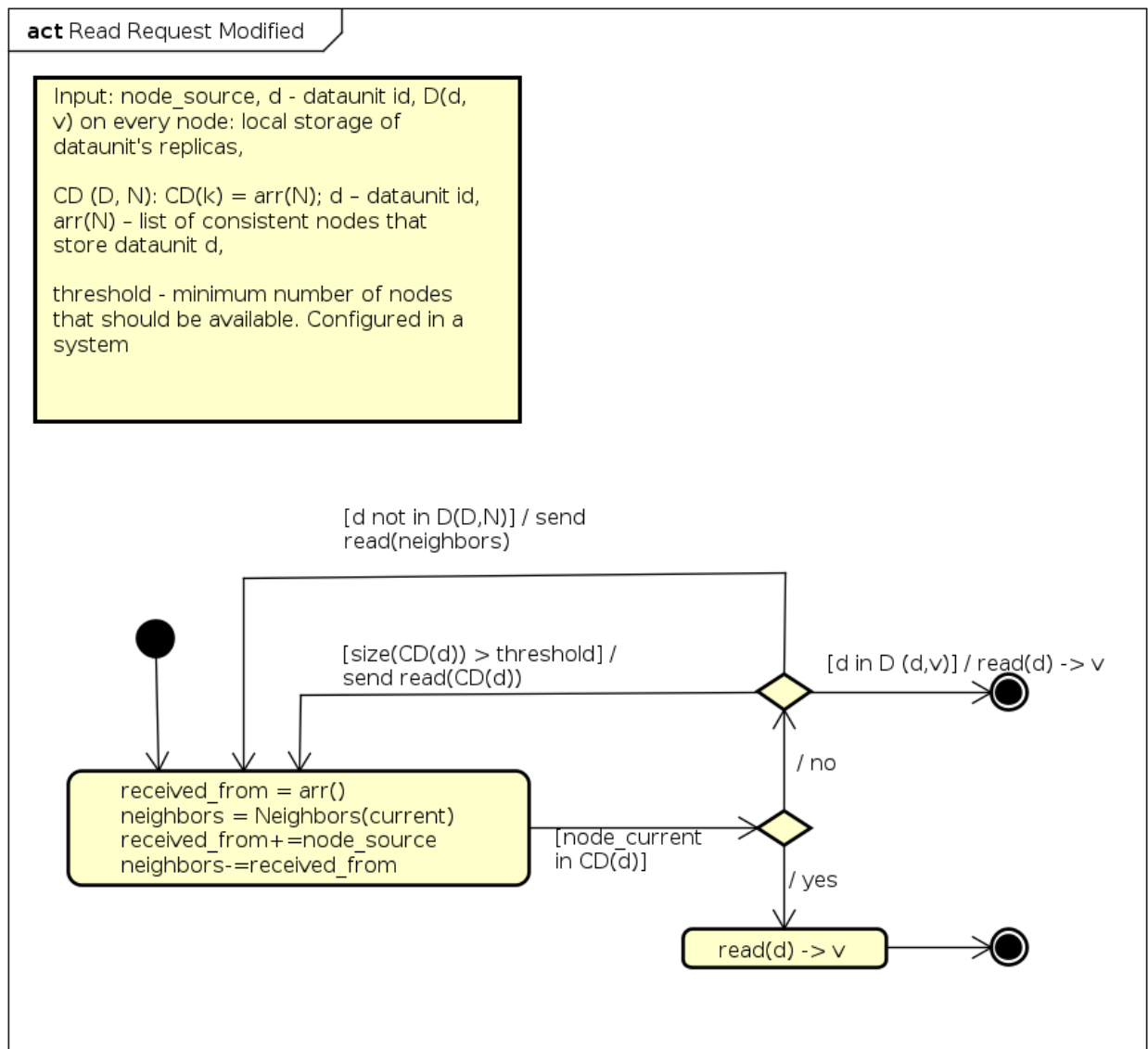


Рис. 3.10. Алгоритм процесу відповіді на запит на читання

Отже, маючи у системі можливість отримати список узгоджених вузлів, кожний вузол, який отримує запит, може звернутися до описаного компоненту, що надає таку інформацію, і відповісти клієнту узгодженою реплікою з умовою, якщо кількість таких узгоджених вузлів не нижче встановленого в налаштуваннях РСД порогу. У іншому разі вузол може відповісти будь-якою доступною реплікою, а саме – відповісти самостійно, якщо він має таку репліку, або переслати запит іншому доступному вузлу.

Таким чином система робить спробу відповідати узгодженим значенням, але якщо це знижує бажаний рівень доступності – відповідати реплікою будь-якої версії.

Основний принцип ефективності цього методі полягає у тому, що найчастіше наповнення таблиці відбувається набагато швидше, ніж надходять запити на запис, бо репліки розповсюджуються мережею епідемічним шляхом, швидкість нарощування списку узгоджених вузлів також залежить від міри того, наскільки близько вузли, які зберігають одні й ті ж самі одиниці даних, розташовані один від одного. Чим менше між ними відстань, тим швидше відновлюється узгодженість, але треба брати до уваги й те, що це буде впливати на саму швидкість відповіді на запит на читання, якщо знання про те, яку одиницю даних зберігає вузол, має тільки сам вузол, а не сервер шлюзу. А оскільки велика частина систем повинна бути настільки децентралізована, наскільки це можливо, скоріше за все, швидкість відповіді на читання впаде у частинах мережі, де вузли, що зберігають запитувані одиниці даних, розташовані більш віддалено один від одного. Тому потрібно бути обережними з такими рішеннями, але тенденція розповсюдження реплік є зрозумілою: чим більше вузлів мають запитувану одиницю даних, тим швидше наповнюється сховище узгодженого відображення, та більша ймовірність отримати відповідь на запит раніше.

Швидке розповсюдження реплік дає зрозуміти, що для багатьох систем метод контрольованого балансування узгодженості має бути ефективний, за винятком таких випадків, коли запити на запис надходять частіше, ніж наповнюється узгоджене відображення, а саме неперервно виконується умова $t(\text{write}(d, v)) - t(\text{write}(d, v')) \leq t(C(d) = n + 1) - t(C(d) = n)$, тобто інтервал, за який кількість узгоджених вузлів стає більше, менше за інтервал між запитами на запис. Наданий метод буде неефективний до того часу, доки не почне порушуватися ця умова, але в даному випадку, якщо необхідна повна узгодженість, у РСД все ще може використовуватися такий метод.

З іншого боку, важко уявити систему, у якій запити на запис на одну й ту саму одиницю надходять настільки часто.

То ж можна зробити висновок, що для більшості випадків наданий метод має бути ефективним з цього боку.

Щоб перевірити ефективність методу контрольованого балансування узгодженості, наведено декілька варіантів архітектури РСД з наданим інтегрованим методом, і проведено експериментальне дослідження, яке підтверджує вірність описаного принципу ефективності: за допомогою реалізованих програмних моделей (опис програмних моделей оцінки методу буде подано нижче у підрозділі 4.4) виміряно кількість вузлів, які узгоджені між собою в умовах різної кількості і частоти надходження запитів на запис, згенеровані графіки відповідно кожному варіанту побудови архітектури.

Отже, нижче описано повний аналіз різних архітектур РСД з модифікованими методами реплікації та відповідей на запити користувача (запити на читання і на запис).

- 1) *Роль балансування між узгодженими вузлами виконує шлюз (gateway), описаний вище.*

На Рис. 3.11 подано діаграму класів програмної моделі, яка реалізує метод контрольованого балансування узгодженості, де роль передачі запитів узгодженим вузлам виконує шлюз.

Шлюзом може виступати як балансувальник навантаження, який приймає запити від клієнта і потім за поданими алгоритмами для методу контрольованого балансування узгодженості пересилає запити відповідним вузлам.

Це рішення має суттєвий недолік: балансувальники навантаження можуть працювати з базами даних на рівні протоколів TCP, але не мають вбудованої функції розбору запиту і отримання інформації про те, яка одиниця даних запитується.

Це суттєво має знизити ефективність такої архітектури – не можна призначати балансувальнику навантаження роль, для якої він не призначений (найкращі практики проектування будь-яких програмних систем зібрані у [84]). У такому випадку шлюз буде витрачати додатковий час на отримання ідентифікатору одиниці даних, яка запитується.

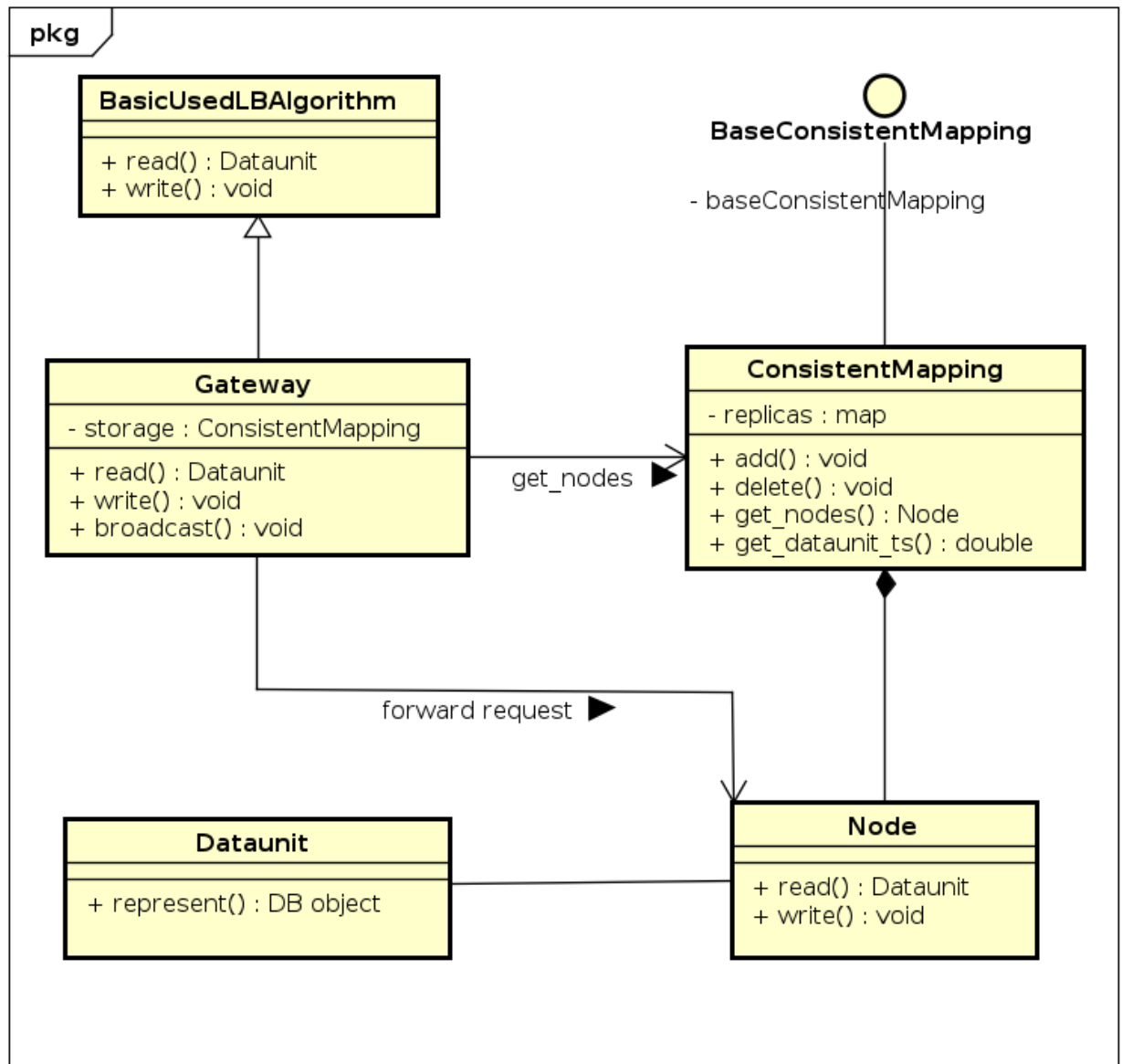


Рис. 3.11. Діаграма класів РСД з методом балансування узгодженості.
Мережевий шлюз у ролі балансування узгодженості

Розглянемо інший варіант, коли шлюзом є вже сервер з мережі РСД, який має всі функції, які мають звичайні сервери бази даних, але також він має спеціальний мережевий інтерфейс, який може спілкуватися з клієнтом і мережевий інтерфейс для спілкування з внутрішньою системою розподіленого сховища даних.

Отже, у такому випадку цей шлюз або знаходиться поруч з вузлом, який відповідає за зберігання узгодженого відображення, або сам зберігає узгоджене відображення, залежно від спроектованої системи (похибка

отримання списку узгоджених вузлів у цьому разі невелика, оскільки такий сервер є найближчим сусідом).

Такий сервер-шлюз, отримавши запит, буде одразу розуміти, на які вузли пересилати запити на читання і за певним алгоритмом буде надсилати запит. Алгоритм такого балансування може бути кастомізованою реалізацією будь-якого алгоритму балансувальника навантаження, який підходить, або за найпростішим алгоритмом Round-Robin (більш детальний опис алгоритмів балансувальника навантаження можна знайти у підрозділі 1.4).

При цьому процес реплікації запиту на запис все ще забезпечується на інших вузлах системи, оскільки інакший спосіб неможливий. Єдине, за що може відповідати шлюз (у разі, якщо він відповідає за зберігання узгодженого відображення) – комунікація з вузлами і модифікація такого відображення впродовж реплікації. Однак такий підхід не рекомендований для внутрішньої мережі, розмір якої – більше 10 вузлів. Згідно з найкращими практиками побудування архітектури різні важливі функції мають виконуватися на різних вузлах, інакше такі рішення погано масштабуються [61].

Взаємодія компонентів такого рішення і те, які саме функції вони виконують, подані на діаграмі на Рис. 3.12.

Механізм обробки запиту на запис не змінюється для різних реалізацій алгоритму, бо у кожному варіанті відповідальність за наповнення об'єкту узгодженого відображення через певний його інтерфейс лежить на всіх вузлах системи РСД, бо тільки вузол, що записав репліку, знає всю потрібну інформацію для наповнення відображення.

Проводячи експерименти на реалізованих програмних моделях згідно з описаними вище принципами і схемами, отримано графіки, які демонструють, що процес реплікації за наданими алгоритмами працює настільки ефективно, наскільки це потрібно, щоб підтримувати рівень узгодженості, потрібний системі, в залежності від заданого порогу кількості мінімального рівня доступності.

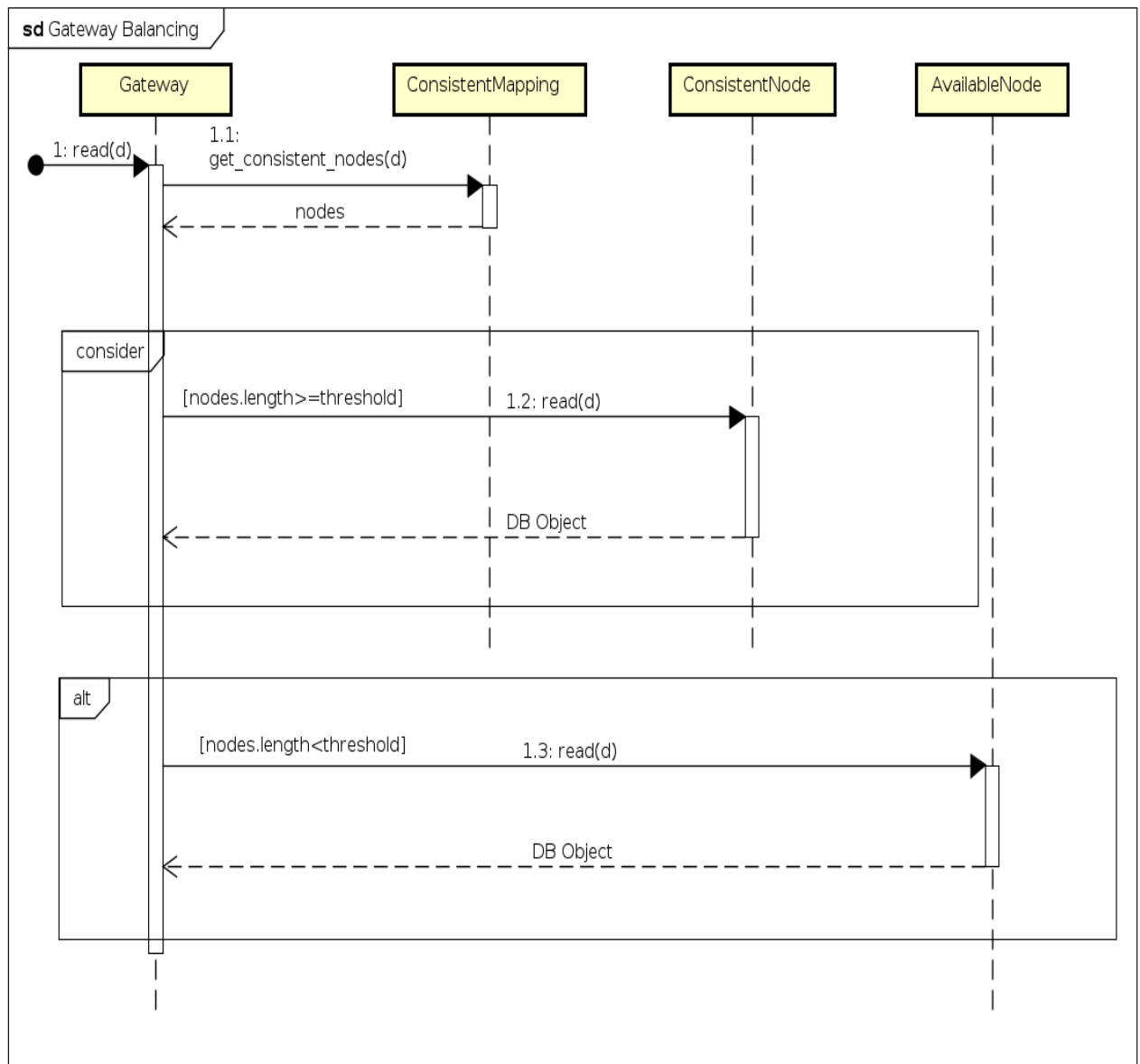


Рис. 3.12. Діаграма послідовності відповіді на запит на читання.

Мережевий шлюз як балансувальник між узгодженими вузлами

Проведено експерименти для різних варіантів використання РСД: коли більше запитів на запис, ніж на читання (200:100, 400:200 (див. Рис. 3.14, Рис. 3.15 відповідно)), навпаки – більше запитів на читання, ніж на запис (200:100) (див. Рис. 3.13).

Червоною прямою лінією позначено визначений системою поріг (наприклад, у поточному експерименті для системи РСД узгодженість є важливішою за доступність і мінімальний рівень доступності визначений в 30 вузлів), ця величина може коливатися залежно від конфігурації системи РСД при реальному її використанні. Ця величина також залежить від пріоритету

узгодженості чи доступності: якщо потрібно, щоб були доступні 70 відсотків вузлів, то у системі перед її використанням налаштовується відповідний поріг для цього методу.

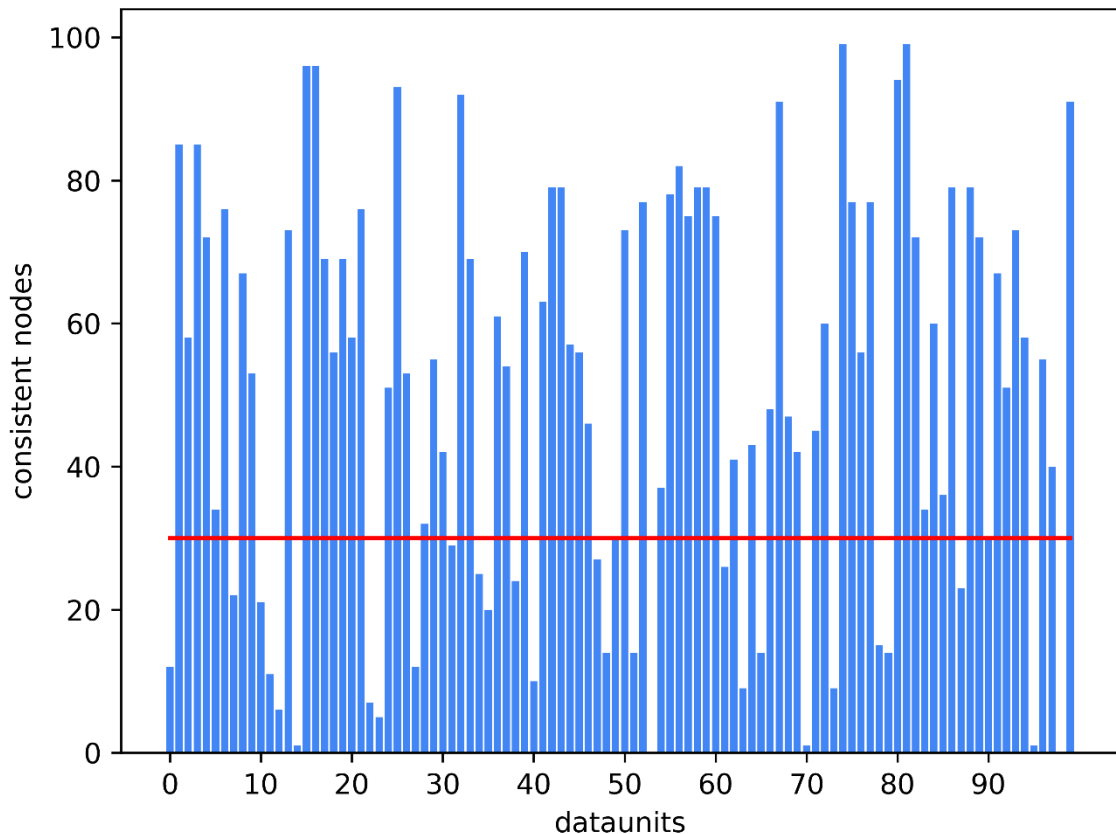


Рис. 3.13. Демонстрація кількості узгоджених вузлів впродовж роботи РСД з інтегрованим методом балансування на сервері-шлюзі з переважаннями запитами на читання

Такий експеримент демонструє, що для певних потреб програмного забезпечення така архітектура РСД може бути вигідною і дешевшою у використанні, тобто якщо роль балансування (другої фази обговорюваного методу) відводиться єдиному центральному вузлу (або невеликому кластеру), який виконує роль шлюзу (або головного, у розподілених базах даних такий вузол називається лідером («leader» [85])).

Але у разі великої мережі (або підмережі), використання одного головного вузла може бути неефективним через управління великою кількістю

вузлів-послідовників (у термінології розподілених баз даних такі вузли називаються послідовниками («follower» [85])).

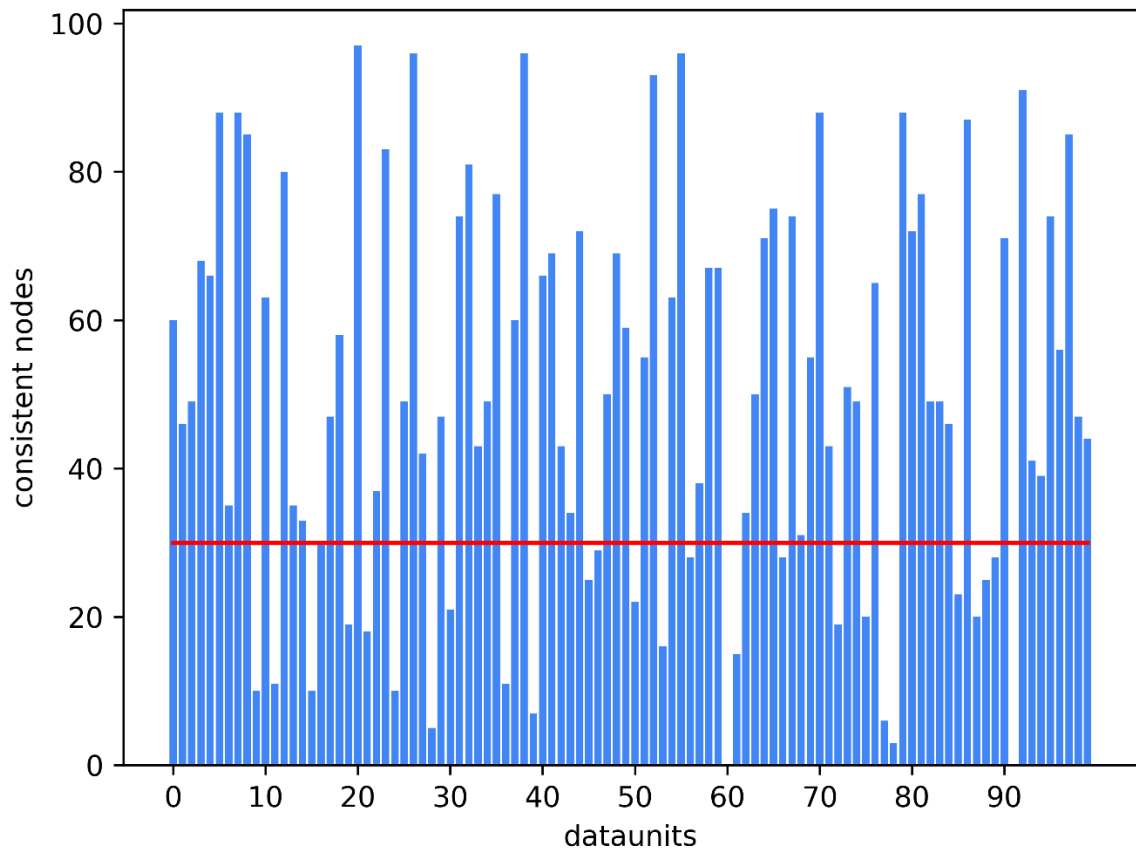


Рис. 3.14. Демонстрація кількості узгоджених вузлів впродовж роботи РСД з інтегрованим методом балансування на сервері-шлюзі з переважаннями запитами на запис

Тому з огляду на конкретні потреби масштабування програмного забезпечення, наприклад, якщо мережа має більше сотні вузлів, є доречним розділяти таку мережу на інші підмережі меншого розміру для застосування такого варіанту алгоритму.

Оцінка ефективності цього варіанту реалізації і всіх відповідних операцій проводиться у підрозділі 4.4, надані відповідні графіки оцінки ефективності за розробленими програмними моделями у Додатку Е до дисертації.

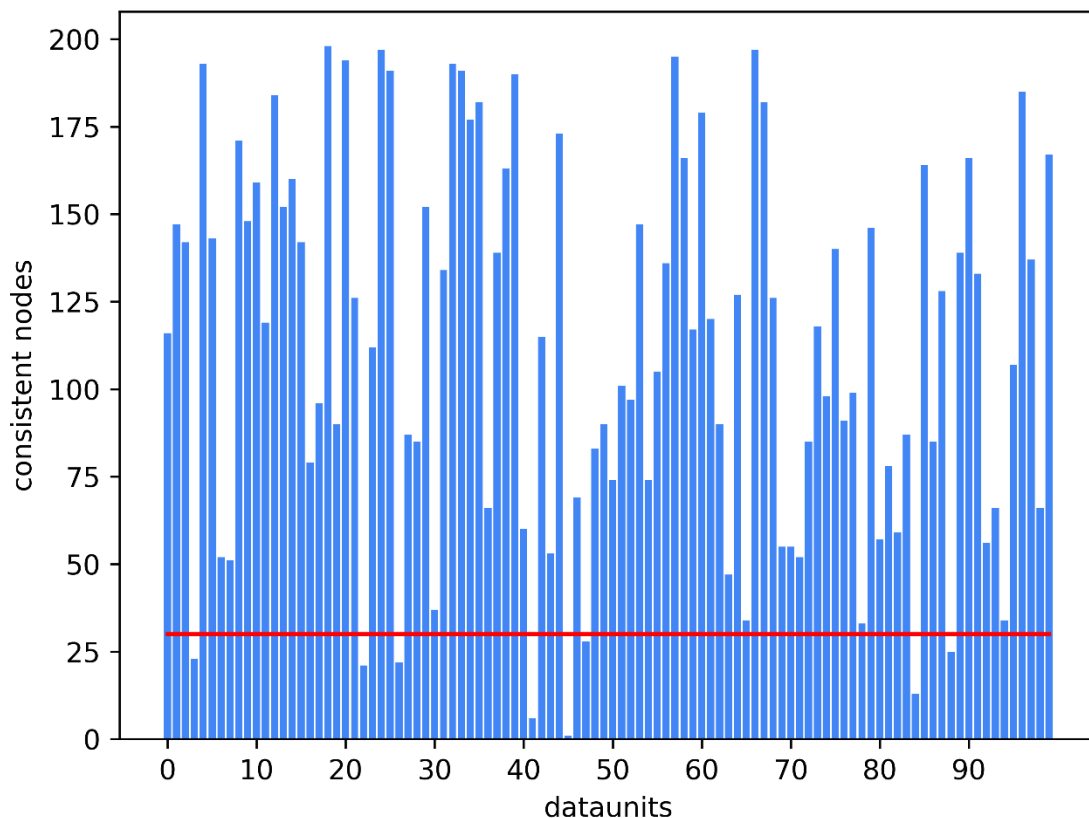


Рис. 3.15. Демонстрація кількості узгоджених вузлів впродовж роботи РСД з інтегрованим методом балансування на сервері-шлюзі з переважаючими запитами на запис

2) *Роль балансування між узгодженими вузлами виконує будь-який вузол в мережі.*

У цьому варіанті реалізації поданого методу роль балансування належить кожному вузлу, а роль сховища компонента узгодженого відображення виконує окремий високо-доступний вузол (або невеликий), що має достатньо ресурсів, щоб відповідати на запити всієї мережі. Оцінка ефективності окремих операцій над таким сховищем оцінюється у підрозділі Теоретичні та практичні оцінки ефективності методу контрольованого балансування узгодженості.

Діаграма класів, подана на Рис. 3.16 та діаграма активності, подана на Рис. 3.17, демонструють інший варіант взаємодії описаних раніше компонентів між собою. У цій реалізації кожний вузол, отримуючи запит на

читання, відправляє запит до узгодженого відображення на отримання списку узгоджених вузлів, і далі виконує операції згідно з алгоритмом, поданим на Рис. 3.10, а саме: перевіряє, чи має він свій власний ідентифікатор у списку узгоджених вузлів. Якщо так, вузол відповідає одразу, у іншому випадку вузол перевіряє, чи довжина отриманого списку не менше встановленого порогу мінімальної кількості узгоджених вузлів і тоді пересилає запит на узгоджені вузли (за допомогою додавання у об'єкт запиту отриманого списку), і далі запит пересилається до відповідного вузла за алгоритмом пошуку найкоротшого шляху [86, 87]).

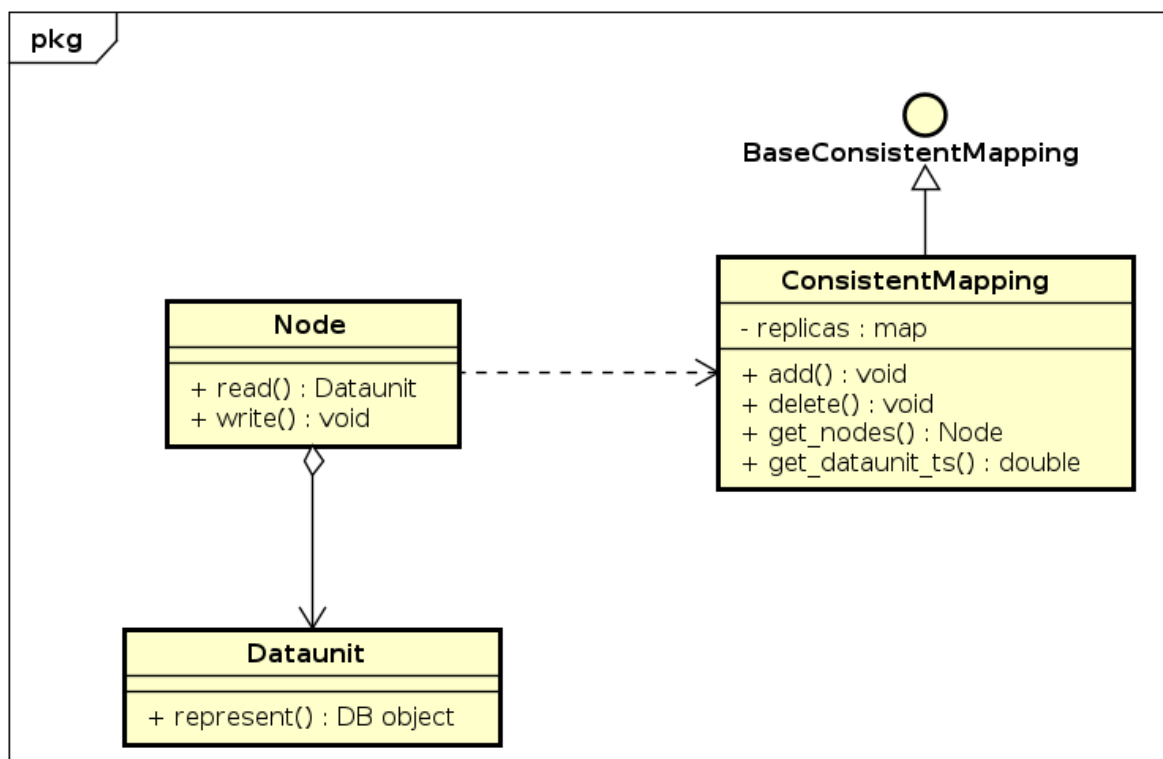


Рис. 3.16. Діаграма класів балансування узгодженості на вузлі

Такий алгоритм є ефективним у разі, якщо ресурси узгодженого відображення задовольняють його навантаження всієї мережею. У протилежному випадку або необхідно розділяти мережу на підмережі, і для кожної підмережі створювати узгоджене відображення, як у попередній реалізації, або додавати більше ресурсів до узгодженого відображення (більше оперативної пам'яті, кількості дозволених з'єднань с сервером, процесор тощо), або переносити вузол узгодженого відображення в іншу мережу, а на

місце вузла узгодженого відображення додавати вузол з підсиленими ресурсами.

Такий підхід може бути ефективний в великих мережах в тисячі вузлів, єдиною умовою є те, що ресурси сховища узгодженого відображення мають відповідати навантаженню на це сховище.

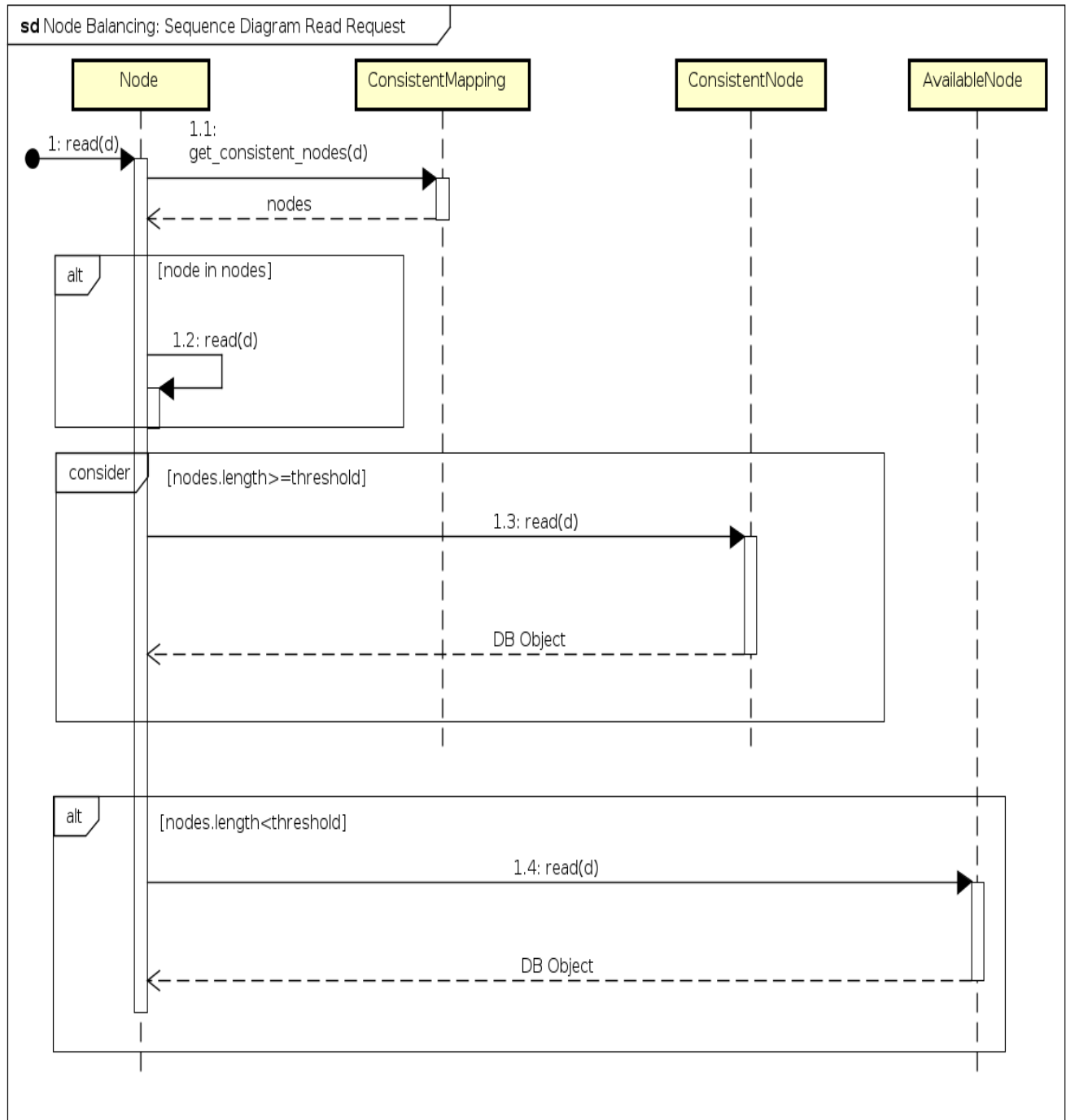


Рис. 3.17. Діаграма активності балансування на вузлі

Але таким самим чином можна використовувати і попереднє рішення з такою умовою.

Вибір між цим чи попереднім рішенням може залежати від того, скільки ресурсів має проєктувальник РСД і скільки ресурсів і яке навантаження на всю систему РСД планується у майбутньому. Ролі впродовж використання РСД і збільшення навантаження можуть перепризначатися, а вузли можуть міняти свої характеристики, і мережа може міняти кількість вузлів.

Оцінка підтримки рівня доступності РСД подана на графіках на Рис. 3.18 та Рис. 3.19. Графіки отримані за допомогою експериментального дослідження, програмна модель якого розроблена за наведеними вище діаграмами класів та діаграми активності.

Проведені експерименти для сховищ від 100 і більше вузлів.

Всі графіки, подані у цьому підрозділі та у Додатку Е дисертації, отримані для РСД у різних комбінаціях станів навантаження на РСД: багато операцій на запис і багато операцій на читання, мало операцій на запис і багато – на читання, і навпаки, багато операцій на запис і мало – на читання.

Роль таких експериментальних досліджень полягає у тому, щоб оцінити ефективність методу при всіх можливих конфігураціях поведінки розподіленого сховища даних для прогнозування певної поведінки у системі РСД з різними рівнями навантаження на систему.

Є зрозумілим, що залежно від кількості запитів, і від того, яких запитів (на читання чи на запис) більше, поведінка методу буде дуже сильно мінятися. Тому заміри кількості доступних вузлів (фактично, вимірювання рівня доступності) є необхідними. Заміри робляться згідно метрики рівня доступності, поданої у підрозділі 3.2.

Розглянуто дві реалізації методу контрольованого балансування узгодженості як програмні моделі, що є базою для подальшої реалізації і інтеграції у існуючі алгоритми, що реалізують процеси реплікації та обробляють запити на запис та читання. Поданий у цьому розділі метод доведений до двох різних алгоритмічних реалізацій, що включають псевдокод, діаграми класів, діаграми активності та діаграми послідовності.

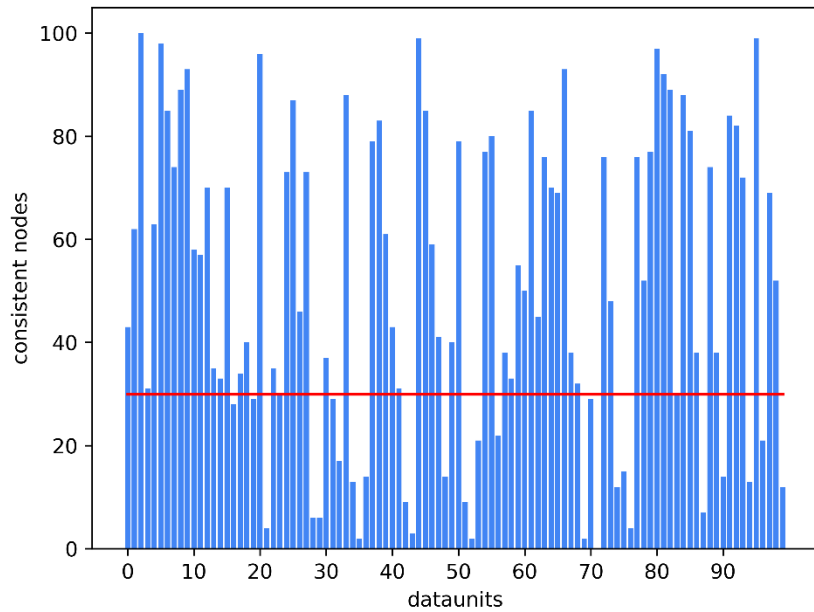


Рис. 3.18. Демонстрація кількості узгоджених вузлів впродовж роботи РСД з інтегрованим методом балансування на внутрішніх вузлах з переважаючими запитами на читання

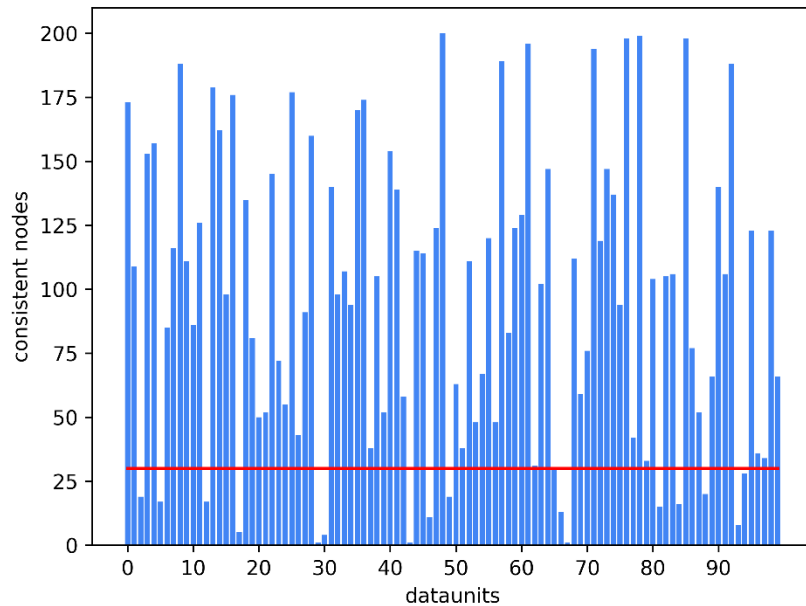


Рис. 3.19. Демонстрація кількості узгоджених вузлів впродовж роботи РСД з інтегрованим методом балансування на внутрішніх вузлах з переважаючими запитами на запис

Актуальність і потрібність цього методу оцінена у цьому підрозділі і подані відповідні графіки (включаючи графіки у Додатку Е), які демонструють

те, що описана концепція може бути ефективною – залежно від кількості вузлів, які мають бути доступними, можна все ще досягти на якомусь етапі збільшення рівня узгодженості у РСД.

Оскільки подані діаграми містять повну інформацію про взаємодію компонентів та об'єктів, розподілення ролей, механізм роботи в різних умовах налаштування і станів системи РСД, для остаточної реалізації достатньо наявних побудованих моделей.

Продемонстрована у підрозділі 4.4 складність таких алгоритмів дає підставу казати про ефективність розроблених методів і можливість їх подальшого впровадження в існуюче програмне забезпечення РСД.

Такий процес – найбільш трудомісткий для цієї мети: повинна бути проведена комунікація між розробниками популярних рішень розподілених баз даних з відкритим кодом, впроваджені зміни мають бути протестовані на середовищах з різними налаштуваннями, проведені інтеграційні і функціональні тести, і врешті-решт впроваджені зміни мають бути затверджені всіма головними розробниками різних компонентів окремого рішення розподіленої бази даних. На даний момент для оцінки ефективності розроблено симуляції згідно поданим програмним моделям, щоб мати підставу для подальшого масштабного впровадження у такі рішення.

3.6 Метод для прийняття рішень при проектуванні РСД

У РСД, як і у будь-якому програмному забезпеченні, велику роль відіграє проектувальне рішення, декомпозиція задач для цього рішення, і вірне їх планування, що дає час на проектування РСД з ефективною архітектурою і можливість більш гнучкого управління ризиками, з якими можуть зустрітися розробники архітектури і користувачі самого РСД.

Для того, щоб спланувати налаштування РСД, яке буде оптимальним для конкретного програмного забезпечення, згідно з відомими парадигмами життєвого циклу розробки програмного забезпечення (або «SDLC model» в [88, 89]) необхідно:

- 1) побудувати первісні моделі вимог до програмного забезпечення;

- 2) проаналізувати моделі вимог на сумісність, розширюваність та гнучкість;
- 3) спроектувати архітектуру проекту: побудувати програмні моделі згідно моделі вимог;
- 4) перевірити архітектуру на розширюваність і гнучкість для управління впровадження нового функціоналу згідно новим вимогам;
- 5) скласти тестові плани для інтеграційного, функціонального тестування і провести повноцінну процедуру тестування готового програмного забезпечення;
- 6) розгортання інфраструктури, запланованої на третьому етапі, завантаження коду програмного забезпечення на розгорнену інфраструктуру;
- 7) підтримка проекту – виправлення помилок, які з'явилися, інтеграція нового функціоналу.

Рішення про використання першої реалізації РСД треба планувати ще з другого етапу – аналізу моделей вимог до програмного забезпечення, оскільки деякі вимоги до програми впливають на вимоги до самого розподіленого сховища даних.

З огляду на такі вимоги до процесів розробки необхідно приймати рішення про використання РСД, його реалізації, налаштування ще на перших етапах розробки задля раціонального управління ризиками. У випадку неправильної архітектури програмне забезпечення буде неефективним, і чим далі підуть такі процеси згідно життєвого циклу програми, тим дорожчим стане управління ризиками і виправлення помилок. Вирішення такої проблеми потягне за собою необхідність перероблювати архітектуру з нуля, що є найдорожчим для будь-якого програмного продукту.

Саме тому проектування будь-якої архітектури є вкрай важливим етапом. Практичні результати, сформовані у цьому розділі, надають можливості більш гнучкого управління ризиками на етапі проектування розподіленої бази даних, які описані нижче у цьому підрозділі.

1) Наявність методів для обчислення рівня середньої ціни доставки дають можливість оцінки ефективності сформованої топології мережі РСД на етапі проектування.

2) Загальні метрики для оцінки рівня CAP-гарантій, подані у підрозділі 3.2 (а саме, метрики для оцінки рівня доступності та стійкості до розділень, що подані у формулах (3.3) та (3.4) відповідно), можуть приймати участь у методах моніторингу системи РСД. Умова доступності системи є зрозумілою і стандартною для загальних конфігурацій РСД. Для умови стійкості до розділення необхідно сформулювати умову стійкості мережі до розділень, тобто моніторинг збоїв комунікаційних каналів, і перевірка системи на факт того, що збій комунікаційних каналів, тобто часткове розділення мережі на 2 або більше ізольованих секцій, не був причиною збою вузла. Така метрика може змінюватися залежно від налаштувань мережі і вузлів мережі.

3) Метрики для оцінки рівня узгодженості, подані у підрозділі 3.3, можуть також бути компонентом методу моніторингу узгодженості на етапі тестування і підтримки програмного забезпечення і своєчасного реагування на операції, які спричиняють недостатній рівень узгодженості, що призводить до некоректної роботи системи.

4) Метод для обчислення ціни відновлення узгодженості під час реплікації у РСД, описаний у підрозділі 3.4, дає змогу оцінити ефективність побудованої мережі і відрегулювати властивості та модифікувати ресурси кожного вузла або комунікаційного каналу мережі, додати більше зв'язків між вузлами або, навпаки, прибрати їх, залежно від поточних потреб. Таке динамічне регулювання мережі особливо актуально у використанні хмарних технологій, де можна резервувати будь-які ресурси і потім відмовлятися від них, не втрачаючи зайві кошти. Такий метод дозволяє зробити такі дії до остаточного запуску РСД до широкого використання.

5) Метод контрольованого балансування узгодженості, який описаний у підрозділі 3.5, є компонентом РСД, який може бути вирішальним у методах прийняття рішень на етапі проектування РСД, у разі, якщо він буде

використаний (такий інтегрований метод з існуючими рішеннями РСД буде необов'язковим функціоналом для підтримки зворотної сумісності у програмному забезпеченні РСД, що у інформатиці має назву «opt-in» [90].

Отже, на базі описаних вище практичних результатів, сформований цілісний компонент методу прийняття рішень на етапі проектування РСД для управління ризиками некоректної поведінки системи розподіленого сховища, запобігання конфліктів реплік, що можуть привести до некоректних результатів розподіленого обчислення, надання користувачу некоректної інформації тощо. Цей компонент поданий у формі опису процесу розробки, текстового набору інструкцій для проектування РСД.

Оскільки надані вище практичні результати фактично є інструментарієм для методу прийняття рішень, маємо такі вхідні об'єкти до методу прийняття рішень:

G – граф, що представляє поточну спроектовану мережу РСД;

N – список вузлів мережі;

R – список вимог до РСД, які сформовані на етапі будування моделі вимог і її аналізу. Може включати такі вимоги як відношення рівня узгодженості до рівня доступності як кількісна величина у процентах або значенні від 0 до 1; бажана ціна доставки для кожної пари вузлів; бажана ціна всього процесу реплікації;

T – підпрограма, яка приймає на вхід граф мережі РСД G із вхідними параметрами початкової ймовірності доставки для кожного комунікаційного каналу, ціни доставки по комунікаційному каналі, та ймовірність доступності кожного вузла, та як результат повертає матрицю цін доставки для кожної пари вузлів, згідно формули (3.21), поданої у підрозділі 3.4;

C – підпрограма, яка обчислює рівень узгодженості за формулою (3.10), поданою у підрозділі 3.3;

A – підпрограма, яка обчислює рівень доступності за формулою (3.3), поданою у підрозділі 3.2;

P – підпрограма, яка задає умови коректного функціонування вузла для поточного стану системи, виконує процес моніторингу мережі і процес реагування на розділення мережі – за допомогою інформації з ізольованих серверів (шляхом періодичної перевірки вузлів, тобто реалізації принципу «heartbeat» або «healthcheck» [38])). На базі такої інформації така підпрограма може прийняти рішення, чи стійкі до розділення ізольовані від іншої частини мережі вузли, які були суміжні з комунікаційними каналами, на яких відбулись збої, і чи продовжуватимуть такі вузли функціонувати коректно.

Маючи такі змінні на базі наданих практичних результатів, сформовано сам метод прийняття рішень у вигляді точних текстових інструкцій.

Крок 1. Впорядкувати список вимог R за пріоритетом від найбільшого до найменшого і почати ітерації над цим списком.

Крок 2. Якщо поточна вимога – це список вузлів, які мають віддавати певні типи даних швидше, ніж інші, відрегулювати їх ресурси згідно потребам. Після задоволення результату перейти до наступної вимоги. Ввести зрегульовані характеристики до таблиці прийнятих рішень.

Крок 3. Вимога бажаної ціни відновлення узгодженості. Якщо поточна вимога – це бажана ціна (максимальна) для відновлення узгодженості або бажана ціна доставки для кожної пари, потрібно запустити підпрограму T , подавши на вхід граф мережі РСД G , описаний у вхідних параметрах вище. Отримавши матрицю цін доставки для кожної пари, зробити її аналіз і прийняти рішення, яка пара повинна мати меншу ціну доставки, а яка може мати й більшу ціну, або рішення, чи влаштовує обчислена ціна реплікації. Відрегулювати характеристики мережі згідно вимогам, і скорегувати ті характеристики, які будуть не гірше, ніж ті, що задовольняють попередню вимогу. Ввести зрегульовані характеристики до таблиці прийнятих рішень.

Крок 4. Перевірка на конфлікти вимог. Якщо отримані характеристики задовольняють вимогу, виконати попередній крок на видозміненій системі РСД. Якщо отримані характеристики не задовольняють попередню вимогу,

зробити вибір між конфліктуючими вимогами або відрегулювати ресурси мережі таким чином, щоб дії для покриття вимог не конфліктували між собою.

Крок 5. Відношення рівня узгодженості до доступності. У разі такої вимоги застосувати метод контрольованого балансування узгодженості (у вигляді вбудованої опції у програмному рішенні РСД, яке використовується). Встановити поріг – бажаний рівень мінімальної доступності вузлів у системі. Необхідно брати до уваги таке: чим менше буде такий поріг, тим більше буде задовольнятися узгодженість і навпаки. Необхідно протестувати поточні налаштування встановленого порогу доступності, та повторювати налаштування і тестування до факту успішного покриття вимоги. Виконати Крок 4 – перевірки попередньої вимоги. Ввести зрегульовані характеристики до таблиці прийнятих рішень.

Крок 6. Бажаний рівень доступності. Оцінити рівень доступності за допомогою підпрограми *A*. У разі, якщо не влаштовує, відрегулювати ресурс заданого серверу, виконати Крок 4 – перевірки попередньої вимоги, і ввести зрегульовані характеристики до таблиці прийнятих рішень.

Крок 7. Бажаний рівень стійкості до розділень. Виконати крок, подібний Кроку 6, однак за допомогою підпрограми *P*.

Таким кроком закінчуються інструкції для підвищення ефективності роботи системи РСД. Ці інструкції можуть розширюватися у разі, якщо вимог до системи стало більше.

Кроки 5, 6, 7 можливо виконувати як на етапі проектування, так і на етапі використання системи з метою її подальшої підтримки.

3.7 Висновки за розділом

У даному розділі подано практичні результати цієї дисертаційної роботи на основі наукових результатів, поданих у попередньому розділі.

1. Сформовано статистичні загальні метрики оцінки рівня САР-вимог у розподілених сховищах даних.

2. За рахунок використання сформованих метрик розроблено метод обчислення рівня узгодженості у розподіленому сховищі даних.

3. За рахунок моделі мережі, побудованої у підрозділі 2.3 і метрики оцінки рівня доступності, розроблено метод обчислення ціни процесу реплікації, який дасть можливість зрозуміти, чи матиме система у якийсь час повну узгодженість, що буде залежати від частоти і кількості записів на запис.

4. На основі проаналізованих рішень балансування у підрозділі 1.4 та описаного базового методу реплікації у підрозділі 2.5, розроблено і доведено до алгоритмічної реалізації метод контрольованого балансування узгодженості, що має бути інтегрований до методів відповіді на запит на запис, на читання і до методів реплікації.

5. На основі поданих у даному розділі практичних результатів сформовані інструкції для проектування, розробки і моніторингу процесів у системі РСД для запобігання ризиків, що можуть спричинити некоректну поведінку системи і згодом привести до помилок, які дорого виправляти.

6. Отримані наукові та практичні результати викладені у роботах [2–4,7–8].

РОЗДІЛ 4

ОПИС ПРОГРАМНИХ МОДЕЛЕЙ ДЛЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Опис предметної області та інструментарію дослідження

Для проведення експериментальних досліджень у РСД в дисертаційній роботі використовувались програмні симуляційні моделі, представлені інструментарієм UML-діаграм, які є стандартним інструментом для проектування будь-якого комплексного програмного рішення ([91–93]).

Складено діаграми компонентів, класів, активності та послідовності ([92]) для кожного з поданих алгоритмів для методу контрольованого балансування узгодженості. Ці діаграми виконують такі ролі у написанні програмного коду:

- діаграма компонентів дає розуміння загальної структури системи і взаємодії основних частин програмованої системи;
- діаграма класів деталізує взаємодію компонентів, розбиває їх на менші частини, описує функціонал, які надають інтерфейси і класи, що поєднуються разом у цілісну структуровану програму;
- діаграма активності описує конкретні операції, які мають бути реалізовані на певному етапі і послідовність та умови їх виконання;
- діаграма послідовності – це діаграма активності з іншого погляду на систему, демонструє, як взаємодіють класи системи і на яких конкретних операціях код в одному класі звертається до певного інтерфейсу (інтерфейсів).

У цьому розділі складено діаграми для програмної реалізації для інших експериментальних досліджень, які потрібні для практичного доведення тверджень, сформованих у попередньому розділі, а саме реалізація експерименту для оцінки рівня узгодженості у РСД, реалізація розповсюдження однієї репліки розподіленим сховищем на РСД з мережею в 100 і більше вузлів.

Також подано результати практичних експериментів над всіма варіантами реалізацій методу контрольованого балансування узгодженості

(див. підрозділ 3.5), подано оцінку ефективності таких реалізацій і зроблено відповідні висновки.

Програмний код експериментальних досліджень реалізовано за допомогою мови програмування Python [94], використано такі бібліотеки: NetworkX – для побудови графів мережі [95], matplotlib [96] – для побудови числових графіків для підтвердження виведених математичних формул та демонстрації потрібності загалом методу контрольованого балансування узгодженості, Python lib – asyncio [97] – для перевірки ефективності методу контрольованого балансування узгодженості та підтвердження висновків, зроблених у підрозділі 3.5.

У наступних підрозділах подано детальний опис симуляційних і програмних моделей для кожного експериментального дослідження.

4.2 Опис програмної моделі для оцінки рівня узгодженості

У підрозділі 3.3 подано математичне твердження 3.1, де продемонстровано формулу (3.10), яка обчислює рівень узгодженості у РСД у поточний момент часу t .

Програмну модель цього експерименту подано на Рис. 4.1.

Суть програмного експерименту полягає у доведенні правильності ймовірнісної формули неузгодженості для РСД, тобто отриманні числового набору значень, на яке має чіткий вплив теоретичне значення, і можна прослідкувати близькість отриманих практичних значень до теоретичного.

Механізм програмного експерименту такий: для кожного експерименту визначається кількість розділень між вузлами – ця величина отримується з заданого файлу конфігурації поточного експериментального дослідження.

Розділяючи множину чисел – ідентифікаторів кожного вузла, визначається кількість вузлів у кожному розділенні, наприклад, у першому – 2, у другому – 5, а у третьому – 3. Це буде означати, що ці підмножини матимуть різні репліки, але в кожній одній підмножині вузли будуть мати однакові між собою версії реплік.

Далі для кожного розділення визначаються підмножини вузлів, перемішується загальна множина вузлів для того, щоб упевнитися, що з загальної множини вузлів два вузли будуть вибиратися абсолютно випадково. Таке перемішування проводиться при кожній виборці.

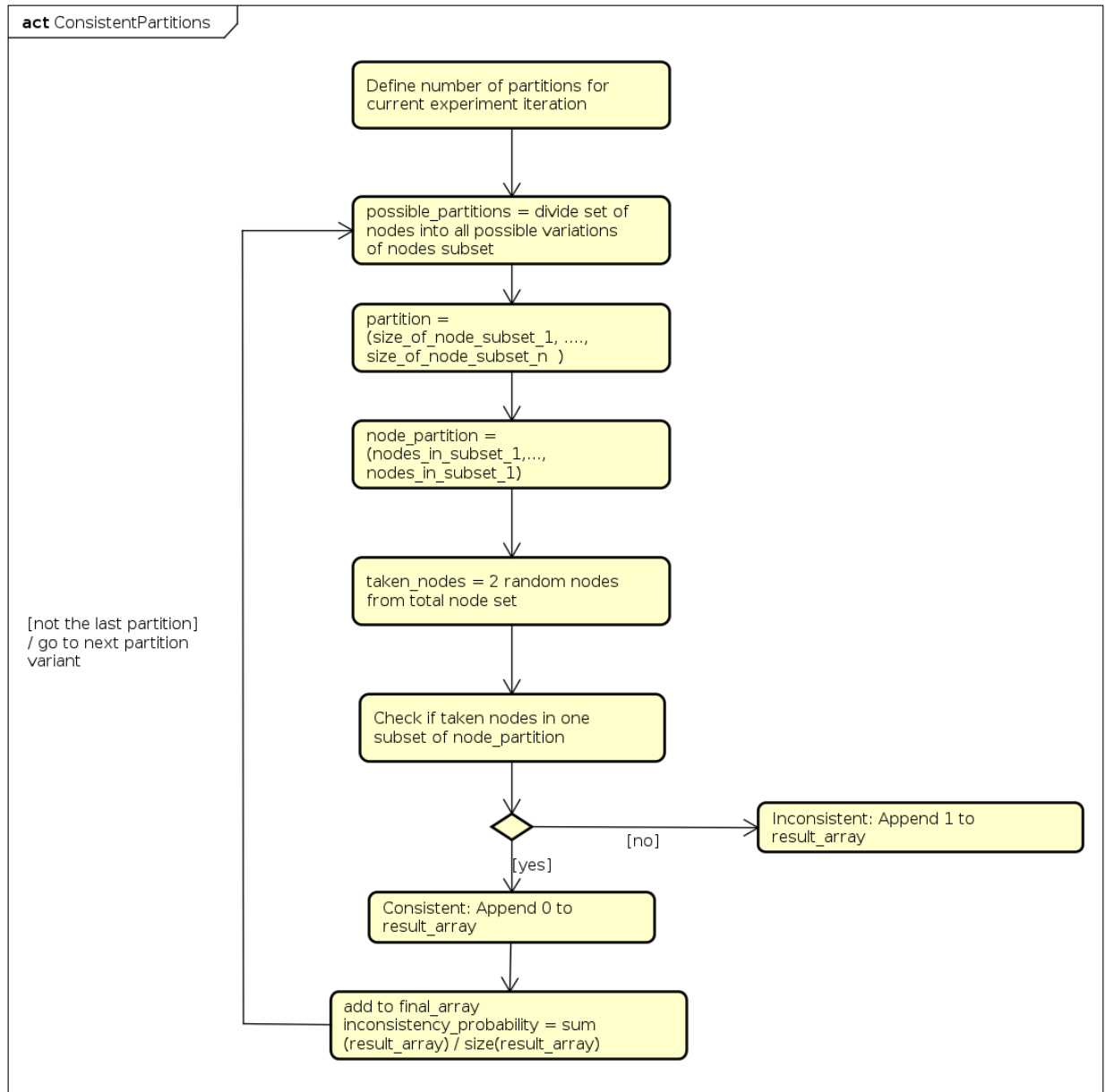


Рис. 4.1. Діаграма активності експерименту про рівень узгодженості у РСД

Отже, для кожної загальної множини вузлів (після перемішування для забезпечення випадковості) для кожного варіанту розділення множини на підмножини випадково вибираються два вузли, і виконується перевірка, чи ці два вузли належать одній підмножині, чи ні. У позитивному результаті вузли вважаються узгодженими між собою, таке значення вважається за 0, оскільки

у експерименті обчислюється неузгодженість, у іншому випадку результат такої перевірки є одиницею.

Такі значення накопичуються в окремий масив, і після цього середнє арифметичне такого масиву заноситься у значення на графіку. Таким чином проводяться численні ітерації експерименту, у кількості 100 і більше ітерацій.

Отже, коли графік заповнюється такими обчисленими значеннями, отримана множина величин наноситься на графік, і як результат, маємо графік, де вісь абсцис виконує роль кількості експериментів, а вісь ординат виконує роль значення оцінки неузгодженості, яке може приймати теоретичне та практичне ймовірнісне значення – від 0 до 1.

Приклади отриманих графіків подано на Рис. 4.2–Рис. 4.4.

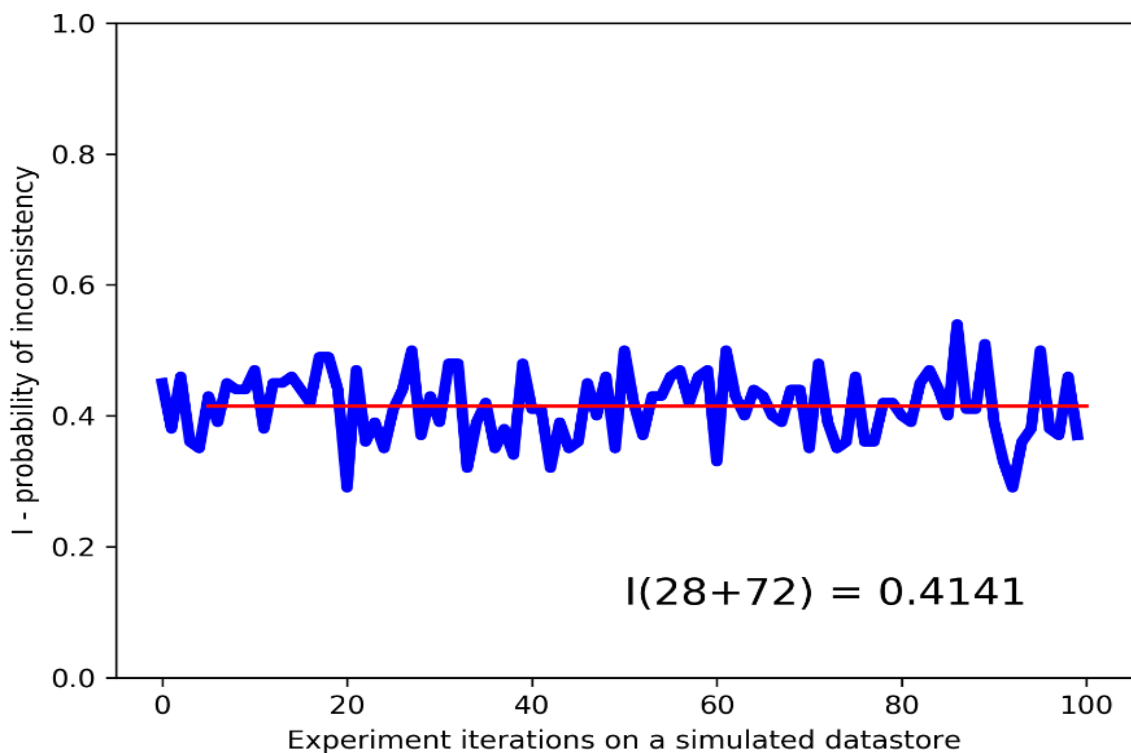


Рис. 4.2. Результати експерименту узгоджених розділень. Демонстрація рівня неузгодженості на двох розділеннях (28 і 72 вузлів)

Отримані графіки демонструють вірність формули (3.5), виведеної у підрозділі 3.3: практичні значення, позначені синьою, хвилястою лінією на графіку, знаходяться близько від теоретичної оцінки, позначеної червоною

прямою лінією на графіку. Експерименти проведено, як для невеликої мережі, так і для мережі на 100 і більше вузлів.

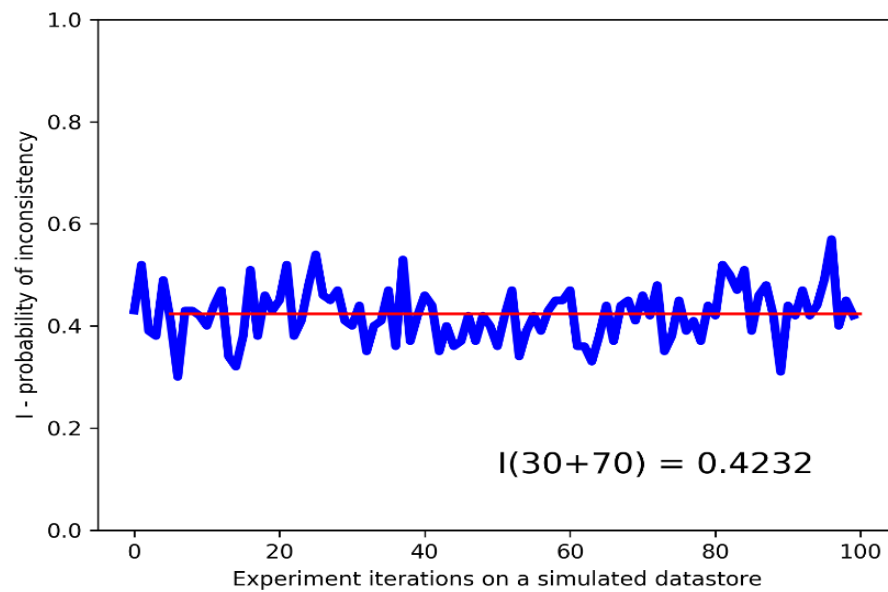


Рис. 4.3. Результати експерименту узгоджених розділень. Демонстрація рівня неузгодженості на двох розділеннях (30 і 70 вузлів)

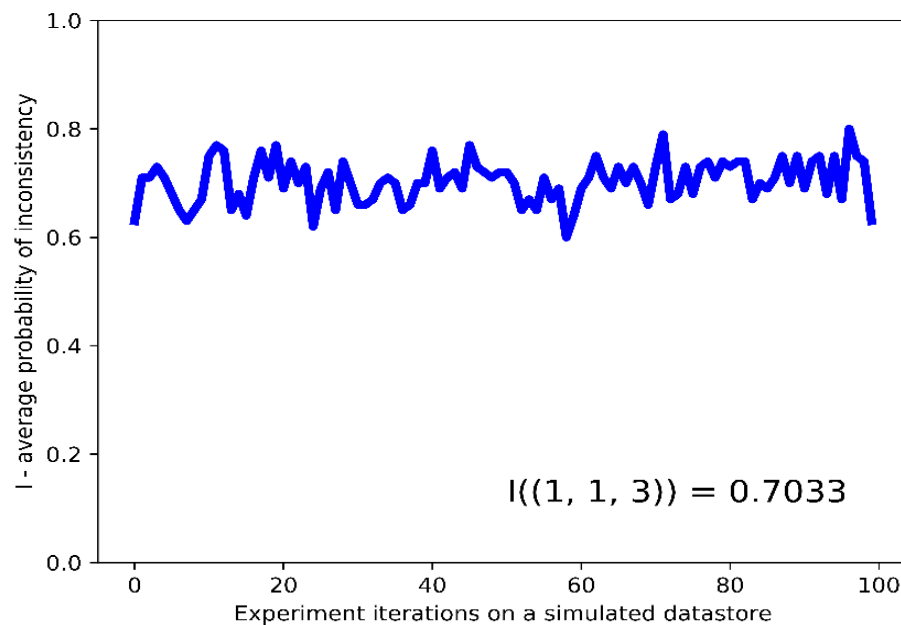


Рис. 4.4. Результати експерименту узгоджених розділень. Демонстрація рівня неузгодженості на трьох розділеннях (1, 1 і 3 вузла)

Решта графіків подана у додатку В дисертаційної роботи.

Такий програмний експеримент обґрунтовується тим, що реальну систему РСД неможливо дослідити таким чином: якщо зупинити систему на певному її стані, це спричинить і зупинку комп'ютерних процесів, що не можна допустити на системі, яка тестується або працює для користувача, а також дані реплік можуть не відповідати реальному стану, бо зупинка процесів може привести і до неправильного відображення стану реплік залежно від способу використання РСД.

4.3 Опис програмної моделі для оцінки процесів реплікації

У підрозділі 3.4 проведено оцінку процесів реплікації у різних умовах мережі РСД на основі початкових характеристик: ціни, ймовірності доставки комунікаційних каналів, ймовірності коректного функціонування вузла.

Спочатку шляхом логічних припущень і використання базису теорії графів та ймовірності доведено твердження 3.2.

Для практичної перевірки твердження проведено програмний експеримент на реалізованій симуляції розподіленого сховища даних, який подано випадковим регулярним зваженим графом мережі, де вага ребер позначає ціни відповідних комунікаційних каналів. А саме, запускається процес реплікації для однієї репліки для наочності результатів, які отримуються. Оскільки у запущеному процесі є можливість виміряти кількість часових слотів, у метриці ціни комунікаційного каналу опускаються інші характеристики, і досліджується тільки час доставки відповідним каналом. У запущеному процесі на графі із зв'язністю більше 1 (що означає, що є більше одного шляху доставки повідомлення для кожної пари вузлів) вимірюється сума ваг ребер пройденого шляху, оскільки реальний час доставки неможливо дослідити на симуляції сховища на локальному сервері.

Отже, за наданими характеристиками проводиться перевірка, що дійсно повідомлення буде швидше доставлено тим шляхом, який має меншу суму цін, і згодом демонструється правильність твердження, що у гіршому випадку час реплікації буде дорівнювати діаметру графа – максимальному з найкоротших

шляхів – шляху, яке пройде репліка між найвіддаленішими між собою вузлами.

Для більш детального опису експерименту подана діаграма класів на Рис. 4.5, що демонструє структуру симульованого розподіленого сховища і взаємодію між структурними елементами цієї діаграми.

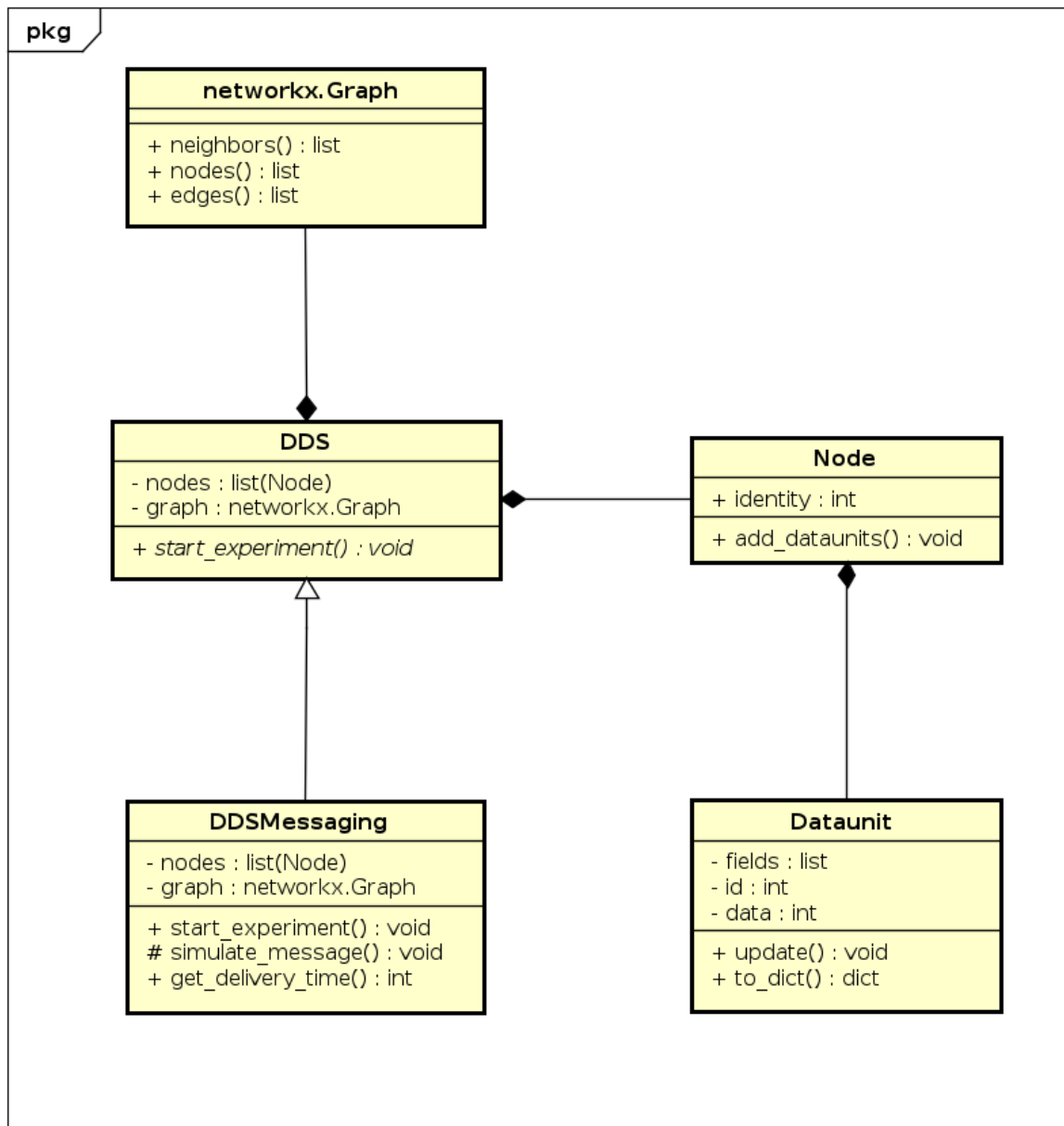


Рис. 4.5. Діаграма класів симуляції реплікації в РСД

Отже, класи, подані на діаграмі класів на Рис. 4.5, описуються так:

- DDS – (Distributed Datastore) базовий клас розподіленого сховища, дозволяє проводити різні експерименти над РСД, для чого потрібно визначити новий клас, успадкований від поточного.

- `DDSMessaging` – клас, оснований на визначенні `DDS`, успадковує його атрибути та метод запуску експерименту. Цей клас запускає конкретний експеримент проходження однієї репліки сховищем і вимірює необхідні характеристики для підтвердження зроблених теоретичних припущень.

- `networkx.Graph` – об'єкт графу, побудований методами бібліотеки для побудування графів `networkx`. Надає широкий функціонал для роботи з мережевими графами.

- `Node` – об'єкт вузла, який має свій ідентифікатор і має функцію додавання одиниць даних. Таким чином у класі `DDSMessaging` одиниці даних розподіляються по списку об'єктів `Node`. Цей об'єкт є основним із всіх структурних елементів, якими маніпулює клас `DDSMessaging` у процесі розповсюдження однієї репліки.

- `Dataunit` – об'єкт одиниці даних, що розподіляється вузлами мережі на початку запуску експерименту. Має функцію оновлення самого себе та функцію репрезентації об'єкта за допомогою метода `to_dict`. По факту представляє об'єкт і самої репліки. У даному експерименті оскільки системою розповсюджується в кожний момент часу одна репліка, не має потреби у створюванні об'єкта репліки.

Для детального опису поведінки кожного класу і послідовності виконання операцій при проведенні експерименту подано діаграму послідовності на Рис. 4.6, що описує поведінковий механізм даного експерименту.

Тобто, на початку симуляції ініціалізується граф, множина вузлів, та одиниці даних розподіляються по вузлах.

Далі алгоритм такий: повторюючи операції для кожного вузла, за допомогою бібліотеки `networkx` обчислюються сусіди поточного вузла, обчислюються шляхи від сусідів до інших вузлів, виключаючи шлях від сусіда до поточного вузла, отримується максимальна ціна серед отриманих шляхів поширення реплік на вузли і додається до списку обчислених цін. Також нарощується множина опрацьованих вузлів – тих, що отримали репліку.

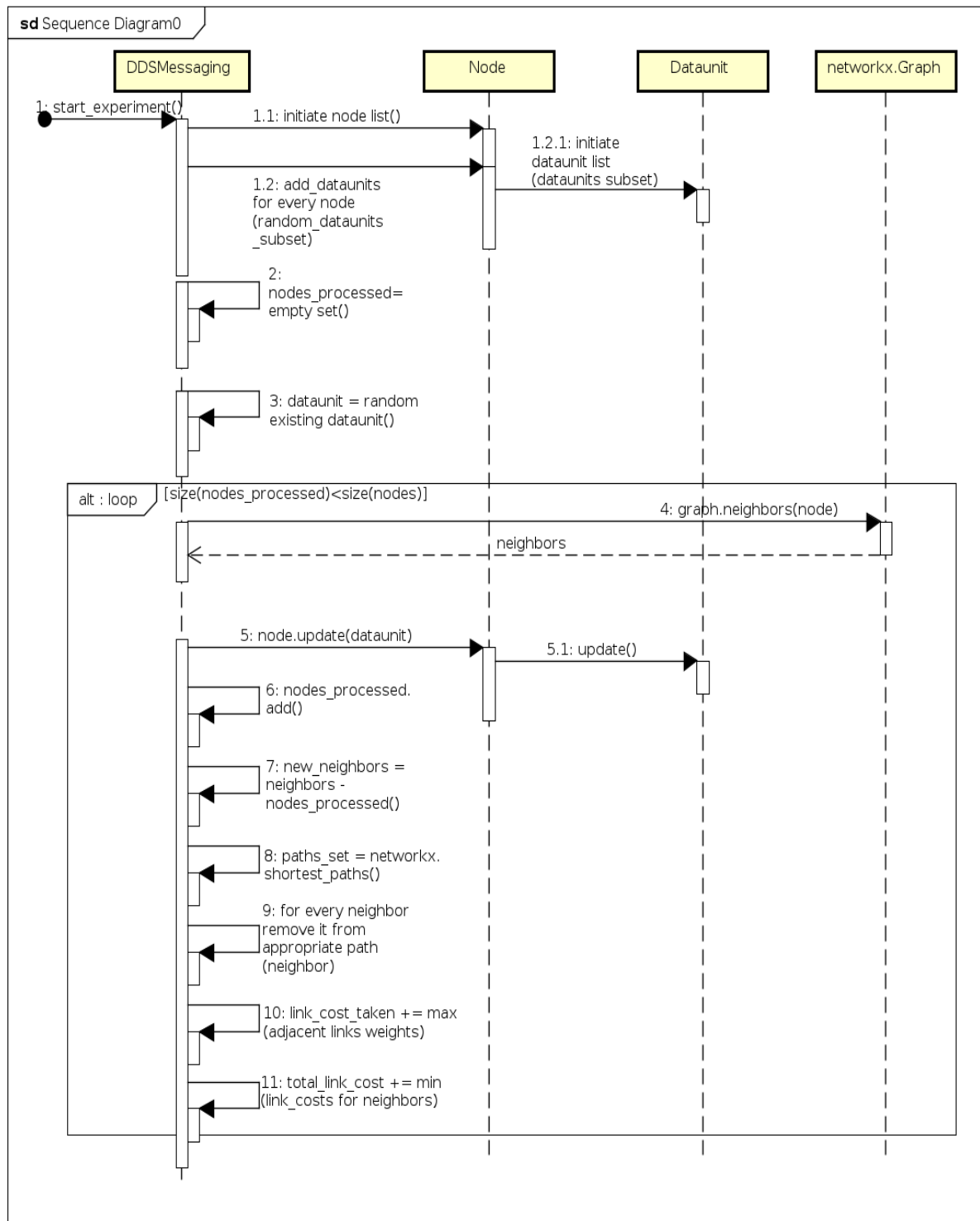


Рис. 4.6. Діаграма послідовності симуляції процесу розповсюдження однієї репліки

Після таких дій програма продовжує повторюватися для інших вузлів, перевіряючи, чи не є вони вже в множині опрацьованих вузлів.

Після проведеного експерименту обчислена ціна реплікації додається у множину результатів експерименту. Далі експеримент повторює симуляцію

ще стільки разів, скільки задано в файлі конфігурації експериментів. Отримані результати для 100 і більше експериментів на РСД з 50 до 100 вузлів в мережі. Результати експерименту подано на Рис. 4.7, Рис. 4.8 та у додатку Г дисертації.

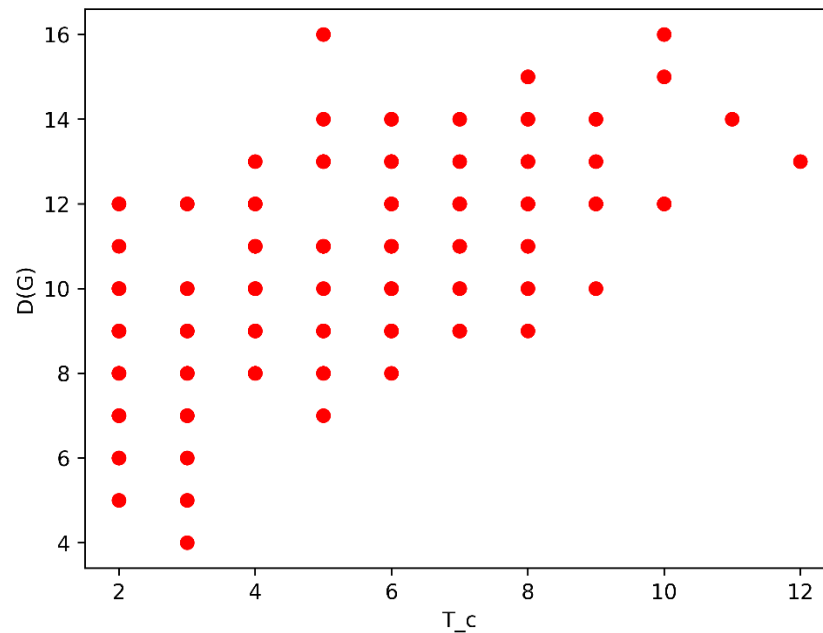


Рис. 4.7. Результати симуляції процесу реплікації з однією реплікою в мережі в 100 вузлів. 200 ітерацій реплікації

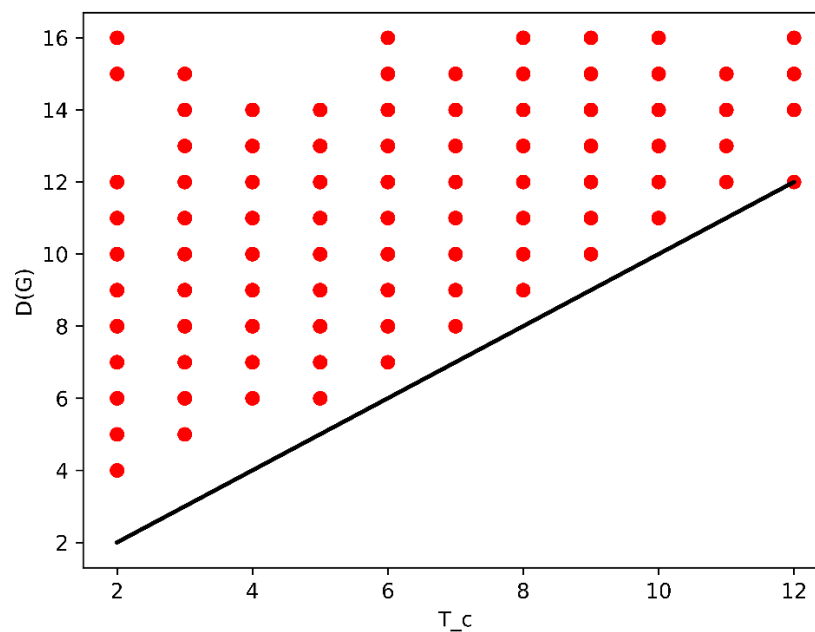


Рис. 4.8. Результати симуляції процесу реплікації з однією реплікою в мережі в 50 вузлів. 1000 ітерацій реплікації

Таким чином, маючи результати такої симуляції, вірним буде зробити висновок, що Твердження 3.2 доведено не тільки за допомогою математичного виведення, але й практичним шляхом.

Також, у додатку Д подано програмну модель, яка обчислює матрицю доставки для кожної пари вершин у графі мережи РСД з втратами даних та збоями на вузлах згідно з формулою (3.20), поданої у твердженні 3.3 та доведеної математично. Елемент обчисленої матриці для пари найвіддаленіших один від одного вузлів є ціною розповсюдження реплік сховищем, тобто ціною відновлення узгодженості. Така величина допоможе зрозуміти верхню межу ціни процесу реплікації і скорегувати характеристики мережи РСД завчасно.

4.4 Теоретичні та практичні оцінки ефективності методу контрольованого балансування узгодженості

У підрозділі 3.5 описаний метод контрольованого балансування узгодженості для регулювання САР-вимог узгодженості і доступності для забезпечення потреб системи РСД згідно з специфікацією.

В даному підрозділі дано оцінку ефективності такого методу згідно загальним метрикам оцінки алгоритмів: часової та просторової складності [98, 99]. Також додані метрики ефективності з огляду на специфіку методу – те, наскільки метод підтримує збалансованість між узгодженістю та доступністю, як вимог розподіленого сховища даних, на належному рівні.

Ефективність спочатку оцінена теоретично на базі складених діаграм активності та послідовності і написаного на їх основі коду.

1) Оцінка фази обробки запиту на читання.

1.1) Для варіації методу у разі виконання балансування на сервері-шлюзі (Рис. 3.11 та Рис. 3.12) часова складність складає $O(n)$ у гіршому випадку і $O(1)$ у кращому випадку, з таких причин:

- часова складність отримання списку узгоджених вузлів з вузла, що має роль сховища узгодженого відображення, дорівнює константі, яка буде

дорівнювати середньому часу доставки до вузла і назад (Round Trip Time metric [100]) та часу отримання значення – списку узгоджених вузлів за запитуваною одиницею даних (у будь-якій програмній реалізації узгодженого відображення складність такої операції дорівнює $O(1)$ у кращому випадку і $O(n)$ в гіршому випадку);

- часова складність порівняння двох числових значень дорівнює константі, оцінка розміру списку також дорівнює константі через оптимізовані структури даних у програмній мові (наприклад, лічильник в списку, що зберігається і змінюється при кожній модифікації структури);

- вибір вузла зі списку вузлів у кращому випадку – $O(1)$, у гіршому випадку – $O(n)$. Це залежить від вибору базового алгоритму балансування. Наприклад, Round-Robin має складність $O(1)$, якщо список вузлів реалізований як черга, з якої вузли прибираються один за одним. Але є інші алгоритми («Central Queue», «Least Connection» тощо), порівняльну характеристику яких подано у підрозділі 1.4.

Просторова складність метода балансування на сервері-шлюзі дорівнює $O(n)$ через зберігання на сервері списку вузлів, отриманих зі сховища узгодженого відображення

1.2) Для варіації методу у разі виконання балансування на вузлі (Рис. 3.16 та Рис. 3.17) продуктивність алгоритму буде дещо нижча через виконання більшої кількості операцій на вузлі (перевірки наявності поточного вузла в списку узгоджених вузлів чи перевірка наявності запитуваної одиниці даних на вузлі):

- часова складність отримання узгоджених вузлів, як і в попередній реалізації, дорівнює константі;
- складність перевірки на наявність поточного вузла у списку узгоджених вузлів – лінійна, то ж маємо $O(n)$;
- складність операцій порівняння та читання також дорівнює константам;

- перевірка на наявність запитуваної одиниці даних на локальному сервері варіюється від $O(1)$ до $O(n)$, оскільки локальне сховище – реалізація словнику, локальна база даних тощо;

- час доставки до вузла і назад, як і в попередній варіації алгоритму, дорівнює константі.

Отже, базуючись на таких підрахунках, часова складність такої варіації алгоритму дорівнює $O(n)$.

Просторова складність такого алгоритму не змінюється і дорівнює $O(n)$ через зберігання списку узгоджених між собою вузлів.

Оцінка фази обробки запиту на запис. Оскільки реалізація фази обробки запиту на запис не змінюється від зміни реалізації балансування на вузлі в мережі чи на вузлі-шлюзі, тому що у будь-якому разі запис реплік відбувається на всій мережі, і нема відповідної ролі для серверу-шлюзу, наразі є необхідним оцінити тільки один варіант алгоритму обробки запиту на запис.

Часова складність такого алгоритму, поданого на Рис. 3.7 та Рис. 3.8, дорівнює $O(n)$ з таких причин:

- складність перевірки наявності одиниці даних у локальному сховищі варіюється від $O(1)$ (у середня оцінка) до $O(n)$ (у гіршому випадку);

- складність виконання операції на запис складає константу – $O(1)$;

- отримання вузлів сусідів для поточного вузла складає $O(n)$ [101];

- перевірка на наявність у списку має складність $O(n)$, а порівняння числових значень - $O(1)$;

- запис до сховища узгодженого відображення складає $O(1)$ (середня оцінка) або $O(n)$ у найгіршому випадку. Але оскільки така операція є асинхронною, цією оцінкою можна знехтувати, або у разі, якщо операція синхронна, запис буде складати константу, тобто також $O(1)$.

Просторова складність алгоритму обробки запиту на запис складає $O(n)$ у один момент часу через зберігання списку сусідів для поширення реплік.

Оцінку продуктивності описаних методів проведено також практичним шляхом. Діаграму активності, що демонструє механізм оцінки методу контрольованого балансування узгодженості, подано на Рис. 4.9.

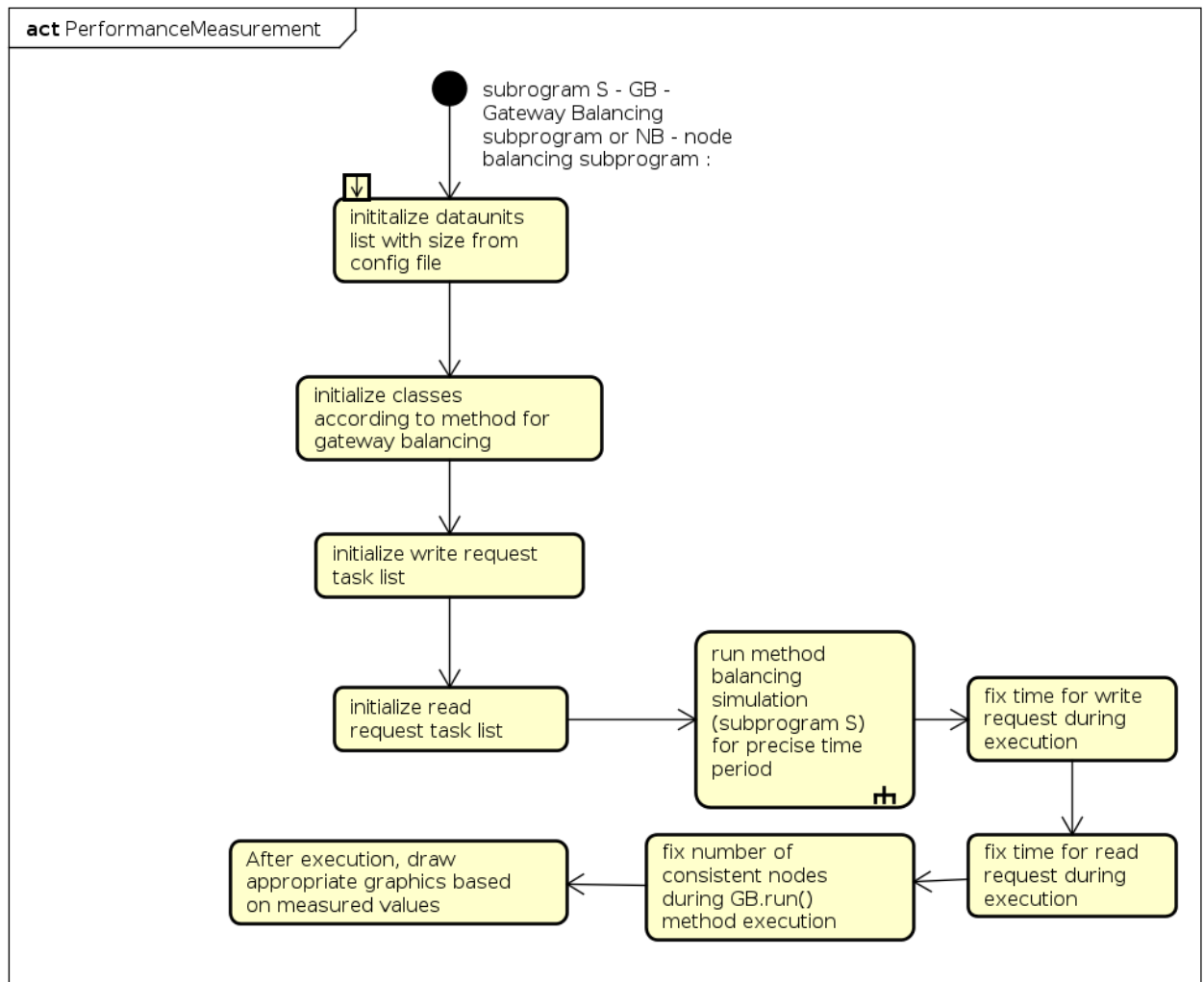


Рис. 4.9. Діаграма активності програми симуляції методів контрольованого балансування узгодженості з оцінкою часу і кількості узгоджених вузлів

Результати оцінки цього методу подано як графіки на Рис. 4.10 та Рис. 4.11 як загальних методів, описаних у формі псевдокоду та діаграм активності на Рис. 3.7–Рис. 3.10, а також у додатках Е, Ж дисертації для всіх варіантів реалізації (оцінка кількості узгоджених вузлів та оцінка часу обробки запитів, відповідно).

Такі оцінки виконані шляхом практичних вимірів часу обробки запитів на читання та запис під час симуляції кожного з наданих реалізацій методу контрольованого балансування узгодженості.

Код експериментальних досліджень подано у репозиторії в системі контролю версій і зберігання програмного коду [102].

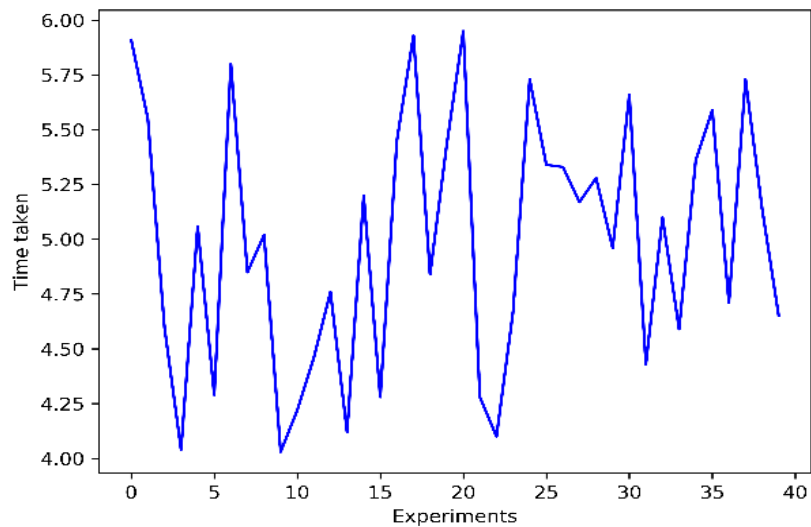


Рис. 4.10. Оцінка часової та просторової складності методу контрольованої узгодженості

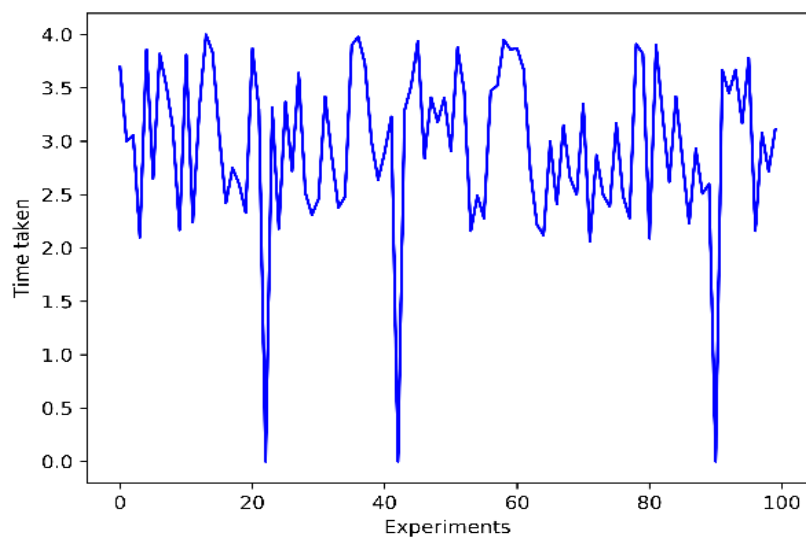


Рис. 4.11. Оцінка часової та просторової складності методу контрольованої узгодженості

4.5 Висновки за розділом

Четвертий розділ містить опис програмних моделей, які реалізують комплекс експериментальних досліджень.

Подані такі практичні результати:

1. Обґрунтовано інструментарій і описано предметну область дослідження.
2. Описано експериментальне дослідження рівня узгодженості у розподілених сховищах даних шляхом розробки програмних моделей, подано відповідні діаграми, що демонструють логіку дослідження.
3. Описано експеримент над розподіленим сховищем у вигляді симуляції процесу реплікації (однієї репліки) на РСД, подано відповідні програмні моделі у вигляді діаграм класів та послідовності.
4. Оцінено ефективність методу контрольованого балансування узгодженості, всі його етапи, у вигляді теоретичної оцінки часової та просторової складності кожного алгоритму, який реалізує такий метод. Подано програмну модель практичної оцінки часової та просторової складності, а також результати цього експерименту. У результаті експеримент відображає ефективність методу контролю узгодженості та доступності шляхом проведення багатьох симуляцій, які реалізовані за допомогою програмних моделей РСД , тобто дозволяє протягом симуляції виконати заміри витраченого часу і використаної пам'яті, а також кількості узгоджених вузлів, які демонструють рівень доступності і узгодженості у РСД у момент часу t .
5. Код всіх експериментальних досліджень зберігається у системі зберігання програмного коду [102].
6. Отримані результати викладені у роботах [2-4].

ЗАГАЛЬНІ ВИСНОВКИ ЗА РОБОТОЮ

У дисертаційній роботі поставлена та вирішена актуальна науково-практична задача забезпечення контрольованого рівня узгодженості реплік у РСД на етапі його проектування, яка складається з моделей, методів та інструментальних засобів для запобігання ризиків, що можуть з'явитися як наслідок неоптимального проектування архітектури системи РСД. Внаслідок виконаних досліджень отримано наукові та практичні результати.

1. Досліджено існуючі технології та протоколи реплікації, які забезпечують виконання CAP-вимог (дві з трьох вимог), що дозволило запропонувати статистичний підхід до задачі, що розглядається, і сформулювати основні невирішені проблеми.

2. Побудовано математичні і імітаційні моделі процесів реплікації даних у РСД, концептуальну модель РСД, що дозволило формалізувати їх основні операції з даними і властивості.

3. Формалізовано статистичні метрики для оцінки рівня виконання кожної з CAP-вимог, що дозволило побудувати модель для оцінки рівня узгодженості реплік у РСД, оцінити процес реплікації і середній час його виконання в умовах з мережевими розділеннями, втратами даних при передачі і можливими збоями вузлів.

4. Розроблено методи управління рівнем узгодженості за умови забезпечення заданих рівнів інших CAP-вимог. Запропоновано і обґрунтовано метод контрольованого балансування узгодженості за умови забезпечення заданих рівней доступності і стійкості до розділення на узгоджених вузлах РСД.

5. Метод балансування доведено до алгоритмічної реалізації і отримано теоретичні оцінки часової і просторової складності для методу балансування, які підтверджені результатами імітаційних експериментів.

6. На основі отриманих практичних результатів сформульовано загальні рекомендації щодо проектування РСД зі збалансованими рівнями виконання CAP-вимог в залежності від технічного завдання.

7. Достовірність отриманих результатів забезпечується математично коректними доведеннями основних тверджень, а також шляхом аналізу результатів комп'ютерних експериментів.

8. Результати роботи використані в освітньому процесі при підготовці фахівців рівня «бакалавр» і «магістр» за спеціальностями 122 – комп'ютерні науки і 126 – інформаційні системи і технології в Запорізькому національному університеті та Херсонському державному університеті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Zholtkevych G., Mallet F., Zaretska I., Zholtkevych G. Two Semantic Models for Clock Relations in the Clock Constraint Specification Language. Communications in Computer and Information Science. 2013. Vol. 1000, P. 190–209.
- [2] Zholtkevych, G. Метрики для обчислення узгодженості у розподілених сховищах даних. Сучасний стан наукових досліджень та технологій в промисловості. 2020. № 2(12). С. 40–48.
- [3] Rukkas, K., Zholtkevych, G. Стохастична модель для оцінки CAP-гарантій для розподілених баз даних. Сучасні інформаційні системи. 2020. №. 4 (2). С. 47–50.
- [4] Rukkas, K., Zholtkevych, G. Балансування узгодженості у розподіленому сховищі даних. Системи управління, навігації та зв'язку. Збірник наукових праць, 2020. №. 2 (60). С. 95–100.
- [5] Zaretska, I., Zholtkevych, G., Zholtkevych, G., Mallet, F. Clocks Model for Specification and Analysis of Timing in Real-Time Embedded Systems . 9th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 9–22 June 2013. Vol. 1356, P. 475–489.
- [6] Rukkas, K., Zholtkevych, G. Distributed Datastores: Towards Probabilistic Approach for Estimation of Dependability. 11th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 14–16 May 2015. Vol. 1356, P. 523–534.
- [7] Rukkas, K., Zholtkevych, G. Distributed Datastores: Consistency Metric for Distributed Datastores. XV Міжнародна науково-практична конференція «Теоретичні та прикладні аспекти побудови програмних систем» ТАAPSD'2017, 4–8 грудня 2017. С. 89–100.
- [8] Zholtkevych, G. PROBABILISTIC MODEL FOR COMPUTING TIME DELIVERY IN DISTRIBUTED DATASTORE NETWORK. VI міжнародна

науково-технічна конференція «Комп'ютерне моделювання та оптимізація складних систем», 4–6 листопада 2020 р. С. 33–35.

[9] Date C. J. An Introduction to Database Systems. 8th ed. Pearson, 2003. 1024 p.

[10] Samaras G., Al-Houmaili Y. J. Two-phase Commit. *Encyclopedia of Database Systems*. New York, NY, 2016. P. 64—71. DOI: https://doi.org/10.1007/978-1-4899-7993-3_713-2.

[11] Tanenbaum A., Van Steen M. Distributed systems - principles and paradigms. 2nd ed. Upper Saddle River, NJ : Prentice-Hall, Inc., 2006. 704 p.

[12] Binani S., Gutti A., Upadhyay S. SQL vs. NoSQL vs. NewSQL- A Comparative Study. *Communications on Applied Electronics*. 2016. Vol. 6, no. 1. P. 43—46.

[13] Piedad F., Hawkins M. High Availability: Design, Techniques, and Processes. US : Pearson, 2000. 288 p.

[14] Milner R. A Calculus of Communicating Systems. *LNCS*. 92nd ed. Berlin, Heidelberg, 1980. P. 179. DOI: https://doi.org/10.1007/3-540-10235-3_1

[15] Lamport L., Lynch N. Distributed Computing: Models and Methods. *Formal Models and Semantics*. Cambridge, MA, United States, 1990. P. 1157—1199. DOI: <https://doi.org/10.1016/B978-0-444-88074-1.50023-8>.

[16] Hoare C. A. R. Communicating sequential processes. *Communications of the ACM*. 1978. Vol. 21, no. 8. P. 666—677. DOI: <https://doi.org/10.1145/359576.359585>

[17] Brewer E. Towards robust distributed systems. *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, Portland, Oregon, 16 July 2000. DOI: <https://doi.org/10.1145/343477.343502>.

[18] Lynch N., Gilbert S. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*. 2002. Vol. 33, no. 2. P. 51—59. DOI: <https://doi.org/10.1145/564585.564601>.

[19] Brewer E. CAP Twelve years later: How the "Rules" have Changed. *Computer*. 2012. Vol. 45, no. 2. P. 23—29. DOI: [10.1109/MC.2012.37](https://doi.org/10.1109/MC.2012.37)

[20] Brewer E. Spanner, TrueTime & The CAP Theorem. Google, 2017. 7 p.
URL: <https://research.google/pubs/pub45855/> (date of access: 17.02.2020).
Назва з екрану.

[21] Gilbert S., Lynch N. Perspectives on the CAP Theorem. *Computer*. 2012. Vol. 45, no. 2. P. 30—36. DOI: [10.1109/MC.2011.389](https://doi.org/10.1109/MC.2011.389)

[22] Cloud Spanner. [Електронний ресурс]: Google Cloud. Режим доступу: <https://cloud.google.com/spanner#section-2> (Дата звернення: 10.05.2019). Назва з екрану.

[23] Cloud Spanner TrueTime and external consistency. [Електронний ресурс]: Google Cloud. Режим доступу: <https://cloud.google.com/spanner/docs/true-time-external-consistency> (Дата звернення: 21. 07. 2020). Назва з екрану.

[24] Further normalization of the database relational model. Englewood Cliffs, NJ : Prentice Hall, 1972. 184 p.

[25] Streaming Replication Protocol. [Електронний ресурс]: PostgreSQL Documentation. Режим доступу: <https://www.postgresql.org/docs/9.5/protocol-replication.html> (Дата звернення: 05.03.2018). Назва з екрану.

[26] Indexing impacts on DML statements [Електронний ресурс]: Use the index, Luke! A guide to Database Performance for Developers. Режим доступу: <https://use-the-index-luke.com/sql/dml/insert>. (Дата звернення: 27.02.2020). Назва з екрану.

[27] Log-Shipping Standby Servers. Section 25.2.5, 25.2.7 [Електронний ресурс]: PostgreSQL Documentation <https://www.postgresql.org/docs/9.2/warm-standby.html>. . (Дата звернення: 28.05.2019). Назва з екрану.

[28] Shlomi N. MySQL and the CAP theorem: relevance & misconceptions. [Електронний ресурс]: презентація Speaker Deck. Електр. дані (47 слайдів). Х., 2019.

[29] What is a Document Database? [Електронний ресурс]. Mongo documentation. Режим доступу: <https://www.mongodb.com/document-databases> (Дата звернення: 15.04.2019). Назва з екрану.

[30] Comparing NoSQL Databases with a Relational Database: Performance and Space / R. L. João et al. *International Journal of Big Data*. 2015. Vol. 2, no. 1. P. 1—14. DOI: [10.29268/stbd.2015.2.1.1](https://doi.org/10.29268/stbd.2015.2.1.1)

[31] Replication. [Електронний ресурс]. Mongo documentation. Режим доступу: <https://docs.mongodb.com/manual/replication/>. (Дата звернення: 17.04.2019). Назва з екрану.

[32] Demers A. et al Epidemic algorithms for replicated database maintenance. *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*. ACM, 1987. P. 1—12. DOI: <https://doi.org/10.1145/41840.41841>.

[33] Overview. [Електронний ресурс]: Apache Cassandra Documentation. Режим доступу: <https://cassandra.apache.org/doc/latest/architecture/overview.html> (Дата звернення: 13.06.2019). Назва з екрану.

[34] Introduction to Apache Cassandra's Architecture [Електронний ресурс]: DZone. Режим доступу: <https://dzone.com/articles/introduction-apache-cassandras>. (Дата звернення: 13.06.2019). Назва з екрану.

[35] Реплікація даних. [Електронний ресурс]: DataStax Documentation. Режим доступу: <https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/architecture/archDataDistributeReplication.html> . (Дата звернення: 17.06.2019). Назва з екрану.

[36] How is the consistency level configured? Read Consistency Levels. [Електронний ресурс]: DataStax Documentation. Режим доступу: <https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlConfigConsistency.html>. (Дата звернення: 02.08.2019). Назва з екрану.

[37] Alakeel A. M. A Guide to Dynamic Load Balancing in Distributed Computer Systems. *IJCSNS International Journal of Computer Science and Network Security*. 2010. Vol. 10, no. 6. P. 153—160.

[38] Healthchecks and Heartbeats. [Електронний ресурс]. Coded Geekery. Режим доступу: <https://codedgeekery.com/blog/healthchecks-and-heartbeats>. (Дата звернення: 12.02.2018). Назва з екрану

[39] Garg A. A comparative study of static and dynamic Load Balancing Algorithms. *IJARCSMS*. 2014. Vol. 2, no. 12. P. 386—392.

[40] Wescott B. Every Computer Performance Book: How to Avoid and Solve Performance Problems ^{[[1]]}on The Computers You Work With. North Charleston, SC, USA : CreateSpace Independent Publishing Platform, 2013. 222 p.

[41] Samadi Y., Zbakh M. Threshold-based load balancing algorithm for Big Data on a Cloud environment. *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*. 2017. P. 1—6. DOI: <https://doi.org/10.1145/3090354.3090372>.

[42] The curse of the first-in–first-out queue discipline. *Games and Economic Behavior*. 2017. Vol. 104. P. 165—176. DOI: <https://doi.org/10.1016/j.geb.2017.03.004>.

[43] Lamport L. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*. 1978. Vol. 21, no. 7. P. 558—565. DOI: <https://doi.org/10.1145/359545.359563>.

[44] André C. The Time Model of logical clocks available in the OMG MARTE profile. *Synthesis of embedded software: Frameworks and methodologies for correctness by construction*. 2010. P. 201—227. DOI: [10.1007/978-1-4419-6400-7_7](https://doi.org/10.1007/978-1-4419-6400-7_7)

[45] Mallet, F. UML Profile for MARTE: Time Model and CCSL. 9th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, June 2013. – vol. 1000, pp. 289-294

[46] Harzheim E. Ordered Sets. New York, NY: Springer-Verlag, 2005. 386 p. DOI: [10.1007/b104891](https://doi.org/10.1007/b104891).

[47] Thiruvathukal G. K., Kaylor S. Lamport's Logical Clocks. *Distributed Systems (COMP 339-439) at Loyola University Chicago*. URL: <https://ds.cs.luc.edu/clocks/clocks.html#lamports-logical-clocks> (date of access: 20.10.2020).

[48] Pessach Y. Distributed Storage: Concepts, Algorithms, and Implementations. Amazon, 2013. 108 p. URL: <https://www.amazon.com/Distributed-Storage-Concepts-Algorithms-Implementations-ebook/dp/B00BGBIA5O>.

[49] Network definition [Электронный ресурс]: Math Insight. Режим доступа: <https://mathinsight.org/definition/network>. (Дата звернення: 12.12.2015). Назва з екрану.

[50] Dharma Rao S., Siva Ram Murthy C. Distributed dynamic QoS-aware routing in WDM optical networks. *Computer Networks*. 2005. Vol. 48, no. 4. P. 585—604. DOI: <https://doi.org/10.1016/j.comnet.2004.11.003>.

[51] Enokido T., Barolli L., Takizawa M. Network-Based Information Systems. *NBiS: International Conference on Network-Based Information Systems. Proceedings*, Regensburg, 3 September 2007. 2007. P. 375—384.

[52] Holland P., Leinhardt S. Transitivity in Structural Models of Small Groups. *Small Group Research*. 1971. Vol. 2, no. 2. P. 107—124. <https://doi.org/10.1177/104649647100200201>.

[53] Watts D. J., Strogatz S. H. Collective dynamics of ‘small-world’ networks. *Nature*. 1998. Vol. 393. P. 440—442. DOI: <https://doi.org/10.1038/30918>.

[54] Rényi E. On random graphs. *Publ. Math. Debrecen*. 1959. Vol. 6. P. 290—297.

[55] Albert R., Barabasi A.-L. Statistical Mechanics Of Complex Networks. *Review of Modern Physics*. 2001. Vol. 74, no. 1. DOI: <https://doi.org/10.1103/RevModPhys.74.47>

[56] Barabási A.-L., Albert R. Emergence of Scaling in Random Networks. *Science*. 1999. Vol. 286, no. 5439. P. 509—512. DOI: <https://doi.org/10.1126/science.286.5439.509>.

[57] Mehlhorn K., Sanders P. Hash Tables and Associative Arrays. *Algorithms and Data Structures*. Berlin, Heidelberg, 2008. P. 81—98. DOI: https://doi.org/10.1007/978-3-540-77978-0_4.

[58] What is a Transaction? [Электронный ресурс]: Microsoft Docs. Режим доступа: <https://docs.microsoft.com/uk-ua/windows/win32/ktm/what-is-a-transaction?redirectedfrom=MSDN> (дата звернення: 20.10.2018). Назва з екрану.

[59] Jelasity M. Gossip-based Protocols for Large-scale Distributed Systems: Doctoral Dissertation. Szeged, 2013. 162 p.

[60] Madria S. Handling of Mutual Conflicts in Distributed Databases Using Timestamps. *Computer Journal*. 1998. Vol. 41, no. 6. P. 376—385. DOI: doi.org/10.1093/comjnl/41.6.376

[61] Bondi A. B. Characteristics of scalability and their impact on performance. *Proceedings of the 2nd international workshop on Software and performance*, New York, NY. 2000. P. 195—203. DOI: <https://doi.org/10.1145/350391.350432>.

[62] Fowler M., Sadalage P. J. Distribution Models. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. 2012.

[63] Dai T. Understanding Real-World Timeout Problems in Cloud Server Systems. *IEEE International Conference on Cloud Engineering (IC2E)*, Orlando, FL, 17 April 2018. 2018. P. 1—11. DOI: [10.1109/IC2E.2018.00022](https://doi.org/10.1109/IC2E.2018.00022).

[64] Andrews G. E. The Theory of Partitions. Cambridge, United Kingdom : Cambridge University Press, 1984. 272 p. DOI: <https://doi.org/10.1017/CBO9780511608650>.

[65] De-hui L. Overview of Gossip Algorithm in Distribute System. *Computer Science*. 2010.

[66] Khelifa M et al. Evaluation of the minimal replication time of Cauliflower mosaic virus in different hosts. *Virology*. 2010. Vol. 396, no. 2. P. 238—245. DOI: <https://doi.org/10.1016/j.virol.2009.09.032>.

[67] Bondy J. A., Murty U.S.R. Bondy. New York, NY : North-Holland, 1976.

[68] Cherkassky B.V, Goldberg A.V, Radzik T. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*. 1996. Vol. 73, no. 2. P. 129—174. DOI: <https://doi.org/10.1007/BF02592101>.

[69] Gut A. Probability: A Graduate Course. New York : Springer-Verlag, 2013. 602 p. DOI: [10.1007/978-1-4614-4708-5](https://doi.org/10.1007/978-1-4614-4708-5).

[70] M. Ross S. Introduction to Probability Models. 11th ed. Oxford : Elsevier, 2014. 767 p.

[71] The Architecture of Transaction Processing Systems: slide-lecture for students [Електроний ресурс]: [presentation], 116 slides. March 2014, <http://www.cs.sunysb.edu/~liu/cse315/23.pdf>.

[72] Load Balancer. [Електронний ресурс]. NGINX Documentation. Режим доступу: <https://docs.nginx.com/nginx/admin-guide/load-balancer/>. Дата звернення (09.01.2020). Назва з екрану.

[73] Apache Module mod_proxy_balancer. [Електронний ресурс]. Apache Http Server Project. Режим доступу: http://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html. Дата звернення (13.01.2020). Назва з екрану.

[74] Kim W., Batory D. S., Reiner D. S. Query Processing in Database System. Springer, 2011. 388 p.

[75] Malhotra S. et al. Generalized Query Processing Mechanism in Cloud Database Management System. *Big Data Analytics. Advances in Intelligent Systems and Computing*. Singapore, 2018. P. 641—648.. DOI: https://doi.org/10.1007/978-981-10-6620-7_61.

[76] Lamport L., Pease M., Shostak R. Reaching Agreement in the Presence of Faults. *Journal of the ACM*. 1980. Vol. 27, no. 2. P. 228—234. DOI: <https://doi.org/10.1145/322186.322188>.

[77] Aguilera M. Stumbling over Consensus Research: Misunderstandings and Issues. *Replication. Lecture Notes in Computer Science*. Berlin, Heidelberg, 2010. P. 59—72. DOI: https://doi.org/10.1007/978-3-642-11294-2_4.

[78] Ongaro D., Ousterhout J. Philadelphia, PA978-1-931971-10-2Open access to the Proceedings of USENIX ATC '14: 2014 USENIX Annual Technical Conference is sponsored by USENIX. In Search of an Understandable Consensus Algorithm. *Proceedings of 2014 USENIX Annual Technical Conference*, Philadelphia, PA, 19–20 June 2014. 2014. P. 305—319.

[79] Accelerate your transformation with Google Cloud [Електронний ресурс]. Google Cloud. Режим доступу: <https://cloud.google.com/>. Дата звернення (27.02.2020). Назва з екрану.

[80] Start building on AWS today. [Електронний ресурс]. Amazon Web Services Режим доступу: <https://aws.amazon.com/> (Дата звернення: 01.03.2020). Назва з екрану.

[81] On-premises, hybrid, multicloud, or at the edge—create secure, future-ready cloud solutions on Azure [Електронний ресурс]. Microsoft Azure. Режим доступу: <https://azure.microsoft.com/en-us/> (дата звернення 15.03.2020). Назва з екрану.

[82] Unuvar, M. et al. Selecting Optimum Cloud Availability Zones by Learning User Satisfaction Levels. *IEEE Transactions on Services Computing*. 2015. Vol. 8, no. 2. P. 199—211. DOI: [10.1109/TSC.2014.2381225](https://doi.org/10.1109/TSC.2014.2381225).

[83] Xie, X. et al. AZ-Code: An Efficient Availability Zone Level Erasure Code to Provide High Fault Tolerance in Cloud Storage Systems. *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*, Santa Clara, CA,, 20–24 May 2019. 2019. P. 230—243. DOI: [10.1109/MSST.2019.000004](https://doi.org/10.1109/MSST.2019.000004).

[84] Fowler M. Patterns of Enterprise Application Architecture. Boston, MA, United States : Addison-Wesley Longman Publishing Co., Inc., 2002. 576 p.

[85] DB Replication (I): Introduction to database replication [Електронний ресурс]. Brains To Bytes. Режим доступу: <https://www.brainstobytes.com/db-replication-i-introduction-to-database-replication/> (Дата звернення 23.09.2017). Назва з екрану.

[86] Ying L., Shakkottai S., Reddy A. On Combining Shortest-Path and Back-Pressure Routing Over Multihop Wireless Networks. *INFOCOM, IEEE Computer and Communications Societies*, Rio de Janeiro, 19–25 April 2009. 2009. P. 1674—1682. DOI: [10.1109/INFCOM.2009.5062086](https://doi.org/10.1109/INFCOM.2009.5062086).

[87] Kwon S., Shroff N. B. Analysis of Shortest Path Routing for Large Multi-Hop Wireless Networks. *IEEE/ACM Transactions on Networking*. 2009. Vol. 17, no. 3. P. 857—869. DOI: [10.1109/TNET.2008.2002222](https://doi.org/10.1109/TNET.2008.2002222).

[88] Pendharkara P. C., Rodger J., Subramanian G. H. An empirical study of the Cobb-Douglas production function properties of software development effort. *Information and Software Technology*. 2008. Vol. 50, no. 12. P. 1181—1188. DOI: [10.1016/j.infsof.2007.10.019](https://doi.org/10.1016/j.infsof.2007.10.019)

[89] Yogesh K. A REVIEW PAPER ON SOFTWARE DEVELOPMENT LIFECYCLE MODELS. *Journal of Emerging Technologies and Innovative Research*. 2018. Vol. 5, no. 2. P. 107—113.

[90] Opt In vs Opt Out [Електронний ресурс]. Termly. Режим доступу: <https://termly.io/resources/articles/opt-in-vs-opt-out/> (дата звернення 30.06.2020). Назва з екрану.

[91] ISO/IEC 19501:2005. Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2. Effective from 2005-04-01. Official edition. 2005. 432 p.

[92] Bernhard R. Modeling with UML: Language, Concepts, Methods. Springer International Publishing, 2016. 281 p. DOI: [10.1007/978-3-319-33933-7](https://doi.org/10.1007/978-3-319-33933-7).

[93] Osman M. H., Chaudron M. R. V. UML Usage in Open Source Software Development: A Field Study. *3rd International Workshop on Experiences and Empirical Studies in Software Modeling*, Miami, 1 October 2013. 2013. P. 23—32.

[94] Python Documentation [Електронний ресурс]. Python Режим доступу: <https://www.python.org/> (Дата звернення: 03.10.2020). Назва з екрану.

[95] Network ANALYSIS FOR python [Електронний ресурс]. Networkx Режим доступу: <https://networkx.org> (дата звернення: 09.10.2020). Назва з екрану.

[96] Matplotlib: Visualization with Python¶ [Електронний ресурс]. Matplotlib Documentation. Режим доступу: <https://matplotlib.org/stable/index.html>. Дата звернення (07.10.2020). Назва з екрану.

[97] Asynchronous I/O [Електронний ресурс]. Python Documentation. Режим доступу: <https://docs.python.org/3/library/asyncio.html> (Дата звернення 11.10.2020). Назва з екрану.

[98] Sipser M. Introduction to the theory of Computation. 2nd ed. Boston, MA : Thomson Course Technology, 2006. 453 p.

[99] Arora S., Barak B. Computational Complexity: A Modern Approach. Cambridge, United Kingdom : Cambridge University Press, 2009. 594 p.

[100] Comer D. Internetworking with TCP/IP - Principles, Protocols and Architecture. Upper Saddle River, NJ, USA : Prentice Hall, 2000. 755 p.

[101] Networkx source code. [Електронний ресурс]: Github. Режим доступу: <https://github.com/networkx/networkx/blob/master/networkx/classes/graph.py#L1202> (Дата звернення 11.10.2020). Назва з екрану.

[102] Simulation of Processes in Distributed Data Stores Project. [Електронний ресурс]: Github. Режим доступу: <https://github.com/galynazholtkevych/DDS-Simulation> (Дата звернення 15.10.2020). Назва з екрану.

ДОДАТОК А

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

[1] Zholtkevych G., Mallet F., Zaretska I., Zholtkevych G. Two Semantic Models for Clock Relations in the Clock Constraint Specification Language. Communications in Computer and Information Science. 2013. Vol. 1000, P. 190–209.

[2] Zholtkevych, G. Метрики для обчислення узгодженості у розподілених сховищах даних. Сучасний стан наукових досліджень та технологій в промисловості. 2020. № 2(12). С. 40–48.

[3] Rukkas, K., Zholtkevych, G. Стохастична модель для оцінки CAP-гарантій для розподілених баз даних. Сучасні інформаційні системи. 2020. №. 4 (2). С. 47–50.

[4] Rukkas, K., Zholtkevych, G. Балансування узгодженості у розподіленому сховищі даних. Системи управління, навігації та зв'язку. Збірник наукових праць, 2020. №. 2 (60). С. 95–100.

[5] Zaretska, I., Zholtkevych, G., Zholtkevych, G., Mallet, F. Clocks Model for Specification and Analysis of Timing in Real-Time Embedded Systems . 9th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 9–22 June 2013. Vol. 1356, P. 475–489.

[6] Rukkas, K., Zholtkevych, G. Distributed Datastores: Towards Probabilistic Approach for Estimation of Dependability. 11th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, 14–16 May 2015. Vol. 1356, P. 523–534.

[7] Rukkas, K., Zholtkevych, G. Distributed Datastores: Consistency Metric for Distributed Datastores. XV Міжнародна науково-практична конференція «Теоретичні та прикладні аспекти побудови програмних систем» ТАAPSD'2017, 4–8 грудня 2017. С. 89–100.

[8] Zholtkevych, G. PROBABILISTIC MODEL FOR COMPUTING TIME DELIVERY IN DISTRIBUTED DATASTORE NETWORK. VI міжнародна

науково-технічна конференція «Комп'ютерне моделювання та оптимізація складних систем», 4–6 листопада 2020 р. С. 33–35.

ДОДАТОК Б

АКТИ ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

вул. Університетська, 27, м. Херсон, 73003. Тел.: +38(0552) 32-67-05, 32-67-31; факс 49-21-14; e-mail: office@ksu.kh.ua; http://www.kspu.edu.ua
код за ЄДРПОУ 02125609 р/р UA22820172034311002200000120; UA068201720343120002000000120
банк Держказначейська служба України, м. Київ

20.01.2021 р. № 12-33/62

На № _____ від _____ 2021 р.

ДОВІДКА

про використання в освітньому процесі
результатів дисертаційного дослідження **Жолткевич Галини Григоріївни**
на тему **“Математичне моделювання процесів реплікації даних у
розподілених сховищах”** за спеціальністю 01.05.02 - математичне
моделювання та обчислювальні методи

Розроблена в дисертаційному дослідженні Г.Г. Жолткевич “Математичне моделювання процесів реплікації даних у розподілених сховищах” за спеціальністю 01.05.02 – математичне моделювання та обчислювальні методи бібліотека утиліт моделювання і аналізу процесів реплікації у розподілених сховищах даних (додаткова інформація за лінкою <https://github.com/galynazholtkevych/DDS-Simulation>), використана в освітньому процесі факультету комп’ютерних наук, фізики та математики Херсонського державного університету за освітньо-професійною програмою «Інформаційні системи і технології» спеціальності 126 – Інформаційні системи і технології другого (магістерського) рівня вищої освіти при викладанні дисципліни «Управління інформаційними системами та сховищами даних», що сприяло удосконаленню професійної підготовки майбутніх інженерів-програмістів.

Упровадження засвідчило високий рівень розробки й практичну значущість розроблених матеріалів. Уважаємо, що результати дисертаційного дослідження Г.Г. Жолткевич можна рекомендувати до використання в освітньому процесі закладів вищої освіти, які здійснюють підготовку майбутніх учителів.

Результати дисертаційного дослідження затверджено на засіданні кафедри інформатики, програмної інженерії та економічної кібернетики (протокол №4 від 05.11.2020 р.) та вченій раді факультету комп'ютерних наук, фізики та математики Херсонського державного університету (протокол № 3 від 16.11.2020 р.).

В.о. ректора

Володимир ПЕСЧАНЕНКО
0552 326768



Сергій ОМЕЛЬЧУК

ЗАТВЕРДЖУЮ
 Ректор Запорізького національного університету
 Міністерства освіти і науки України
 М. О. Фролов
 2020 р.



ДОВІДКА

про використання в освітньому процесі
 результатів дисертаційного дослідження Жолткевич Галини Григорівни
 на тему «Математичне моделювання процесів реплікації даних у розподілених
 сховищах» за спеціальністю 01.05.02 – математичне моделювання та
 обчислювальні методи

Розроблена в дисертаційному дослідженні Жолткевич Г. Г. «Математичне моделювання процесів реплікації даних у розподілених сховищах» за спеціальністю 01.05.02 – математичне моделювання та обчислювальні методи бібліотека утиліт моделювання і аналізу процесів реплікації у розподілених сховищах даних (див. <https://github.com/galynazholtkevych/DDS-Simulation>), використана в освітньому процесі математичного факультету Запорізького національного університету за освітньо-професійною програмою «Комп'ютерні науки» спеціальності 122 – Комп'ютерні науки першого (бакалаврського) рівня вищої освіти при викладанні дисципліни «Паралельні та розподілені обчислення».

Члени комісії:

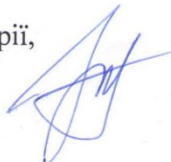
Декан математичного факультету
 (голова комісії),
 д.т.н., професор

 С. І. Гоменюк

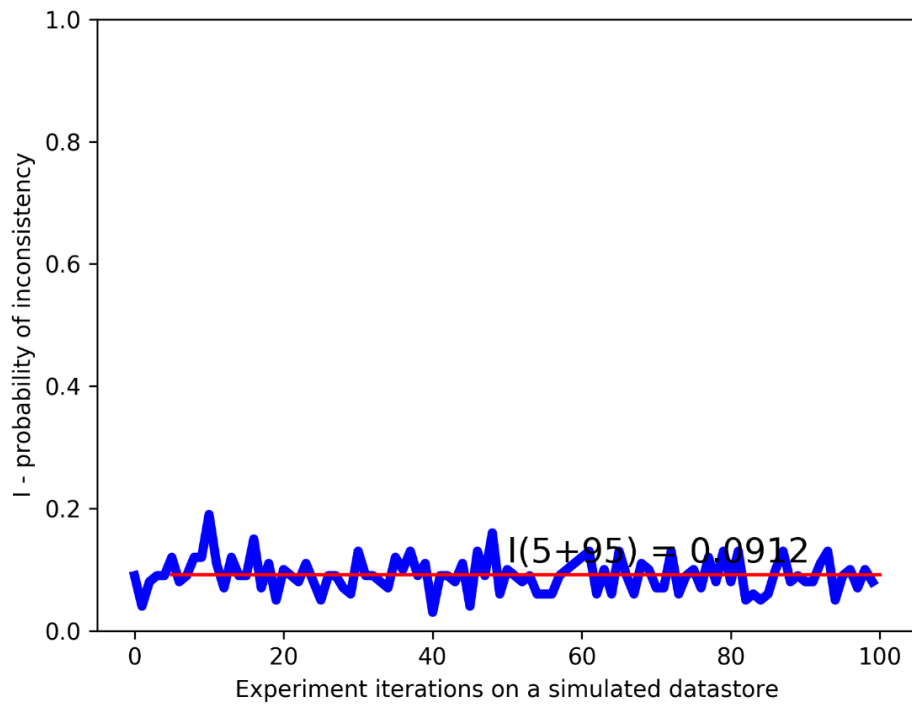
Завідувач кафедри комп'ютерних наук,
 к.т.н., доцент

 С. Ю. Борю

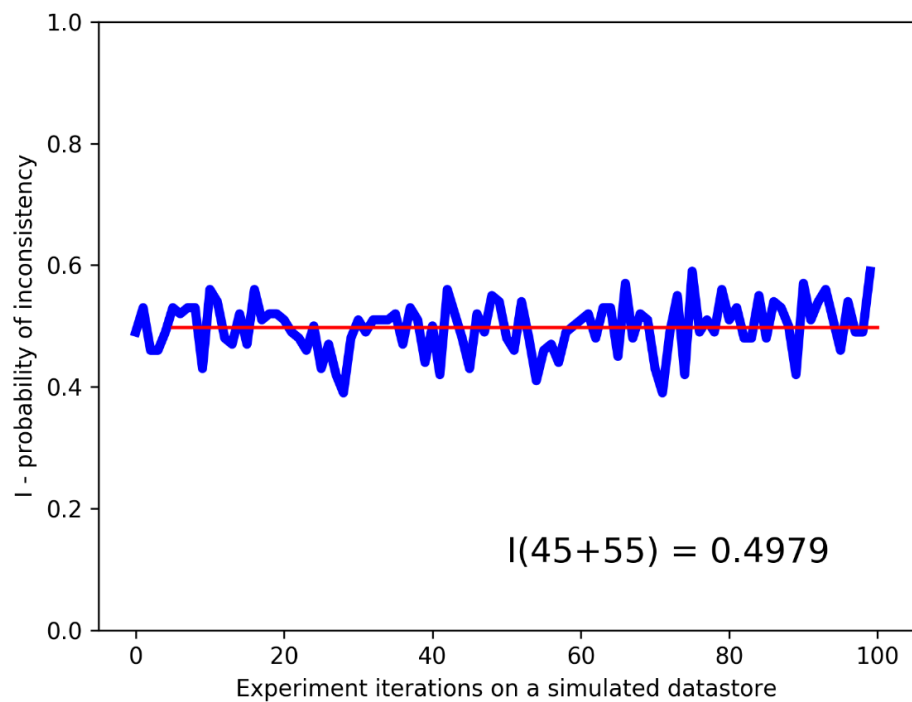
Завідувач кафедри програмної інженерії,
 к.ф.-м.н., доцент

 А. О. Лісняк

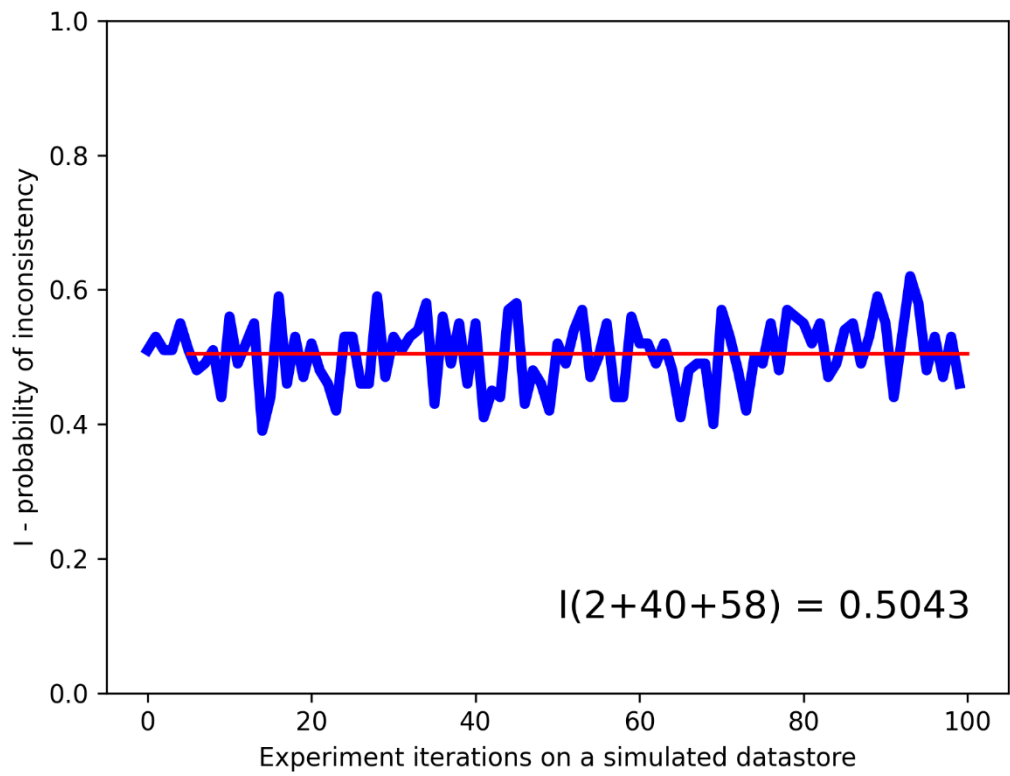
ДОДАТОК В
РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ ОБЧИСЛЕННЯ РІВНЯ
УЗГОДЖЕНОСТІ



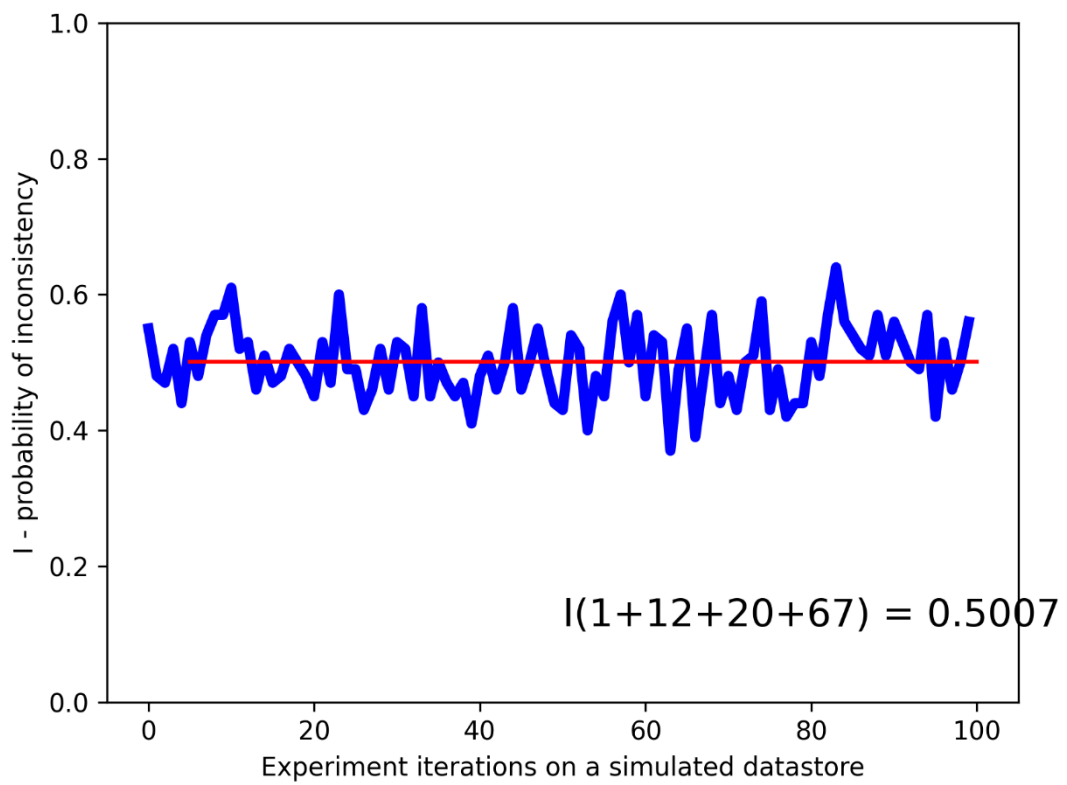
Рівень узгодженості у РСД (100 вузлів). Розділення (5, 95)



Рівень узгодженості у РСД (100 вузлів). Розділення (45,55)

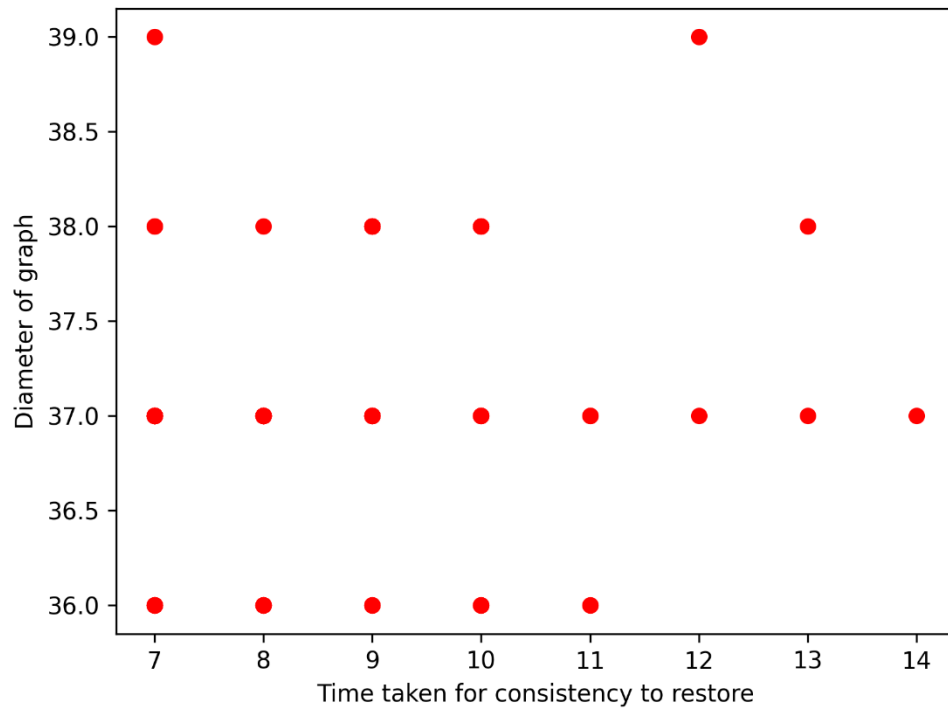


Рівень узгодженості у РСД (100 вузлів). Розділення (2, 40, 58)

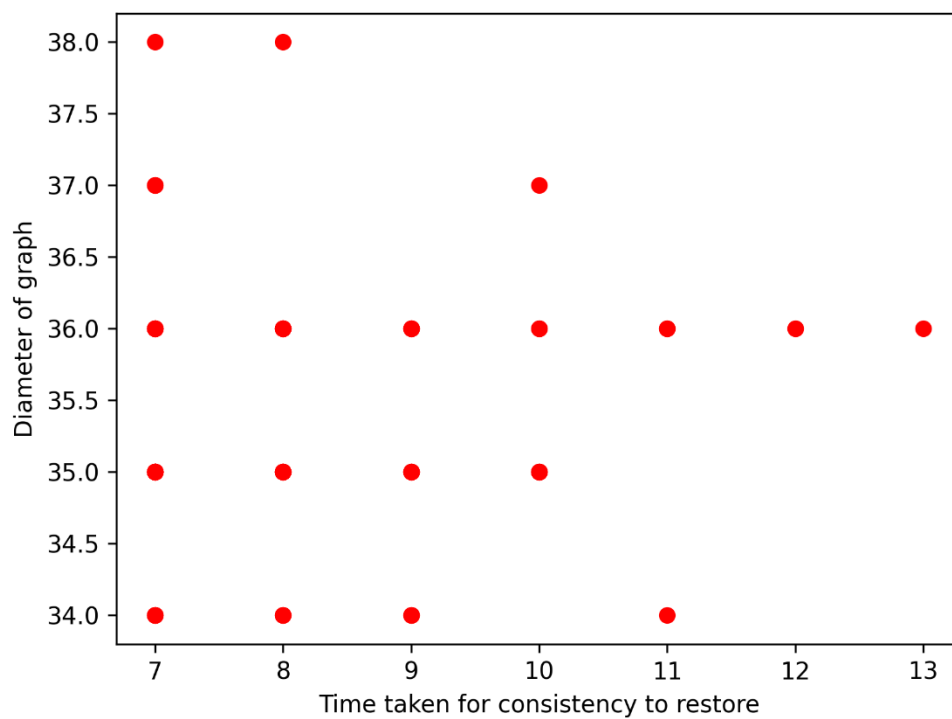


Рівень узгодженості у РСД (100 вузлів). Розділення (1, 12, 20, 67)

ДОДАТОК Г
РЕЗУЛЬТАТИ СИМУЛЯЦІЇ РЕПЛІКАЦІЇ У РСД



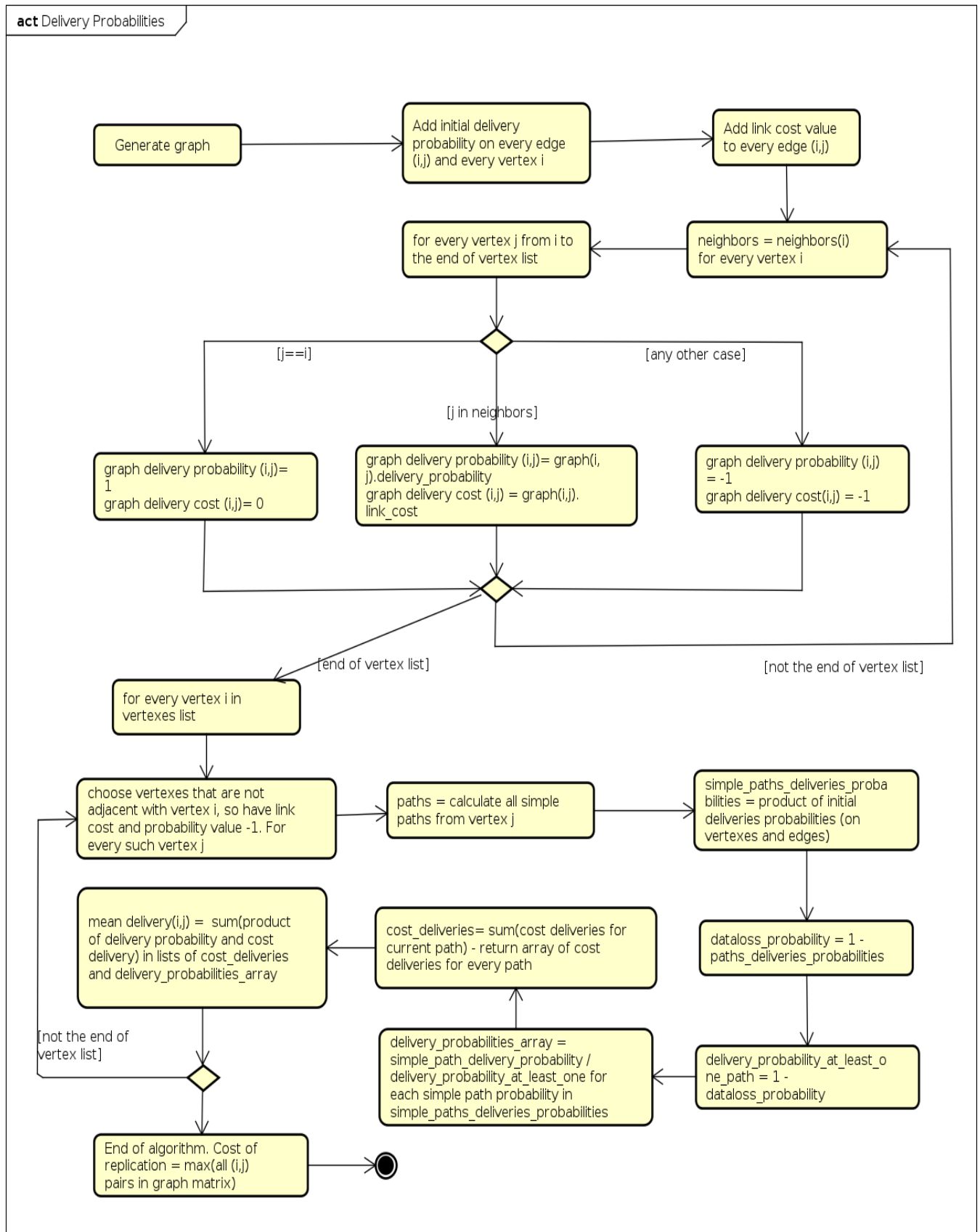
Симуляція реплікації у РСД на 3000 вузлів. 100 ітерацій



Симуляція реплікації у РСД на 2000 вузлів. 100 ітерацій

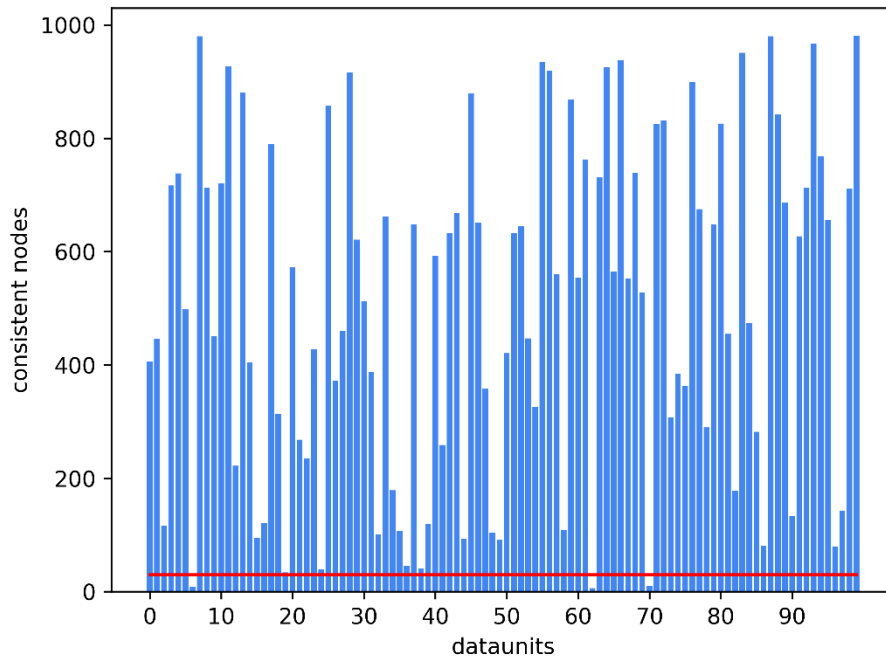
ДОДАТОК Д

ПРОГРАМНА МОДЕЛЬ ОБЧИСЛЕННЯ МАТРИЦІ СЕРЕДНЬОЇ ДОСТАВКИ У РСД

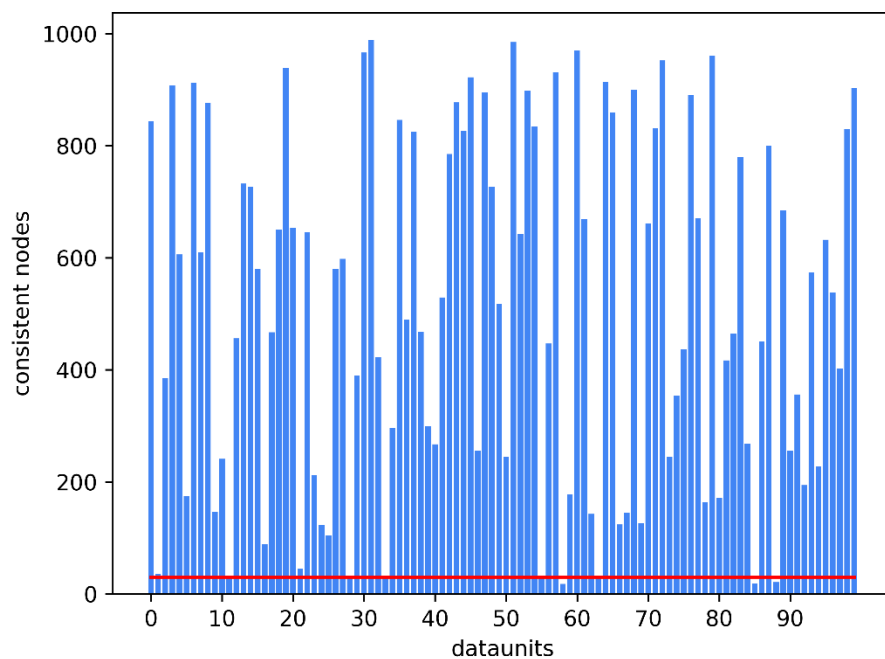


ДОДАТОК Е
РЕЗУЛЬТАТИ СИМУЛЯЦІЇ МЕТОДУ КОНТРОЛЬОВАНОГО
БАЛАНСУВАННЯ УЗГОДЖЕНОСТІ

Шлюз у ролі балансування

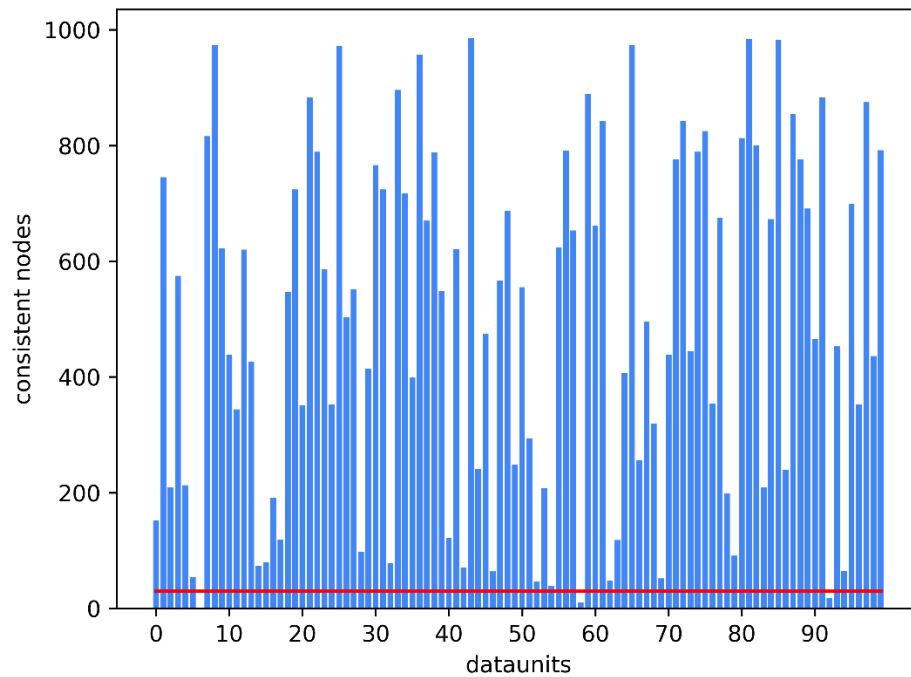


Кількість узгоджених вузлів. 1000 запитів на запис і 2000 запитів на читання.

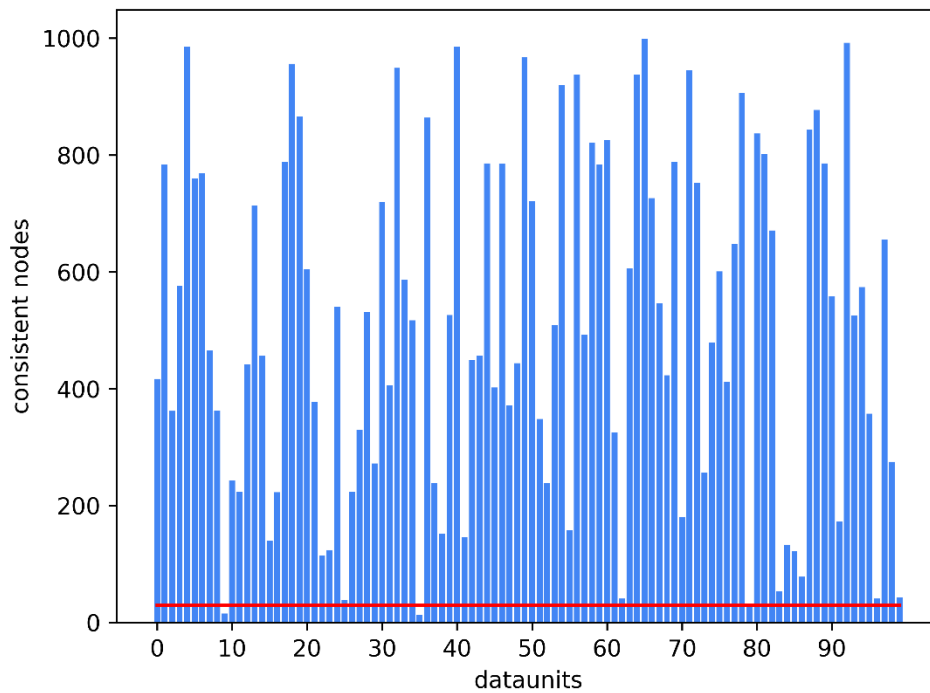


Кількість узгоджених вузлів. 2000 запитів на запис, 1000 запитів на читання

Вузли у ролі балансування



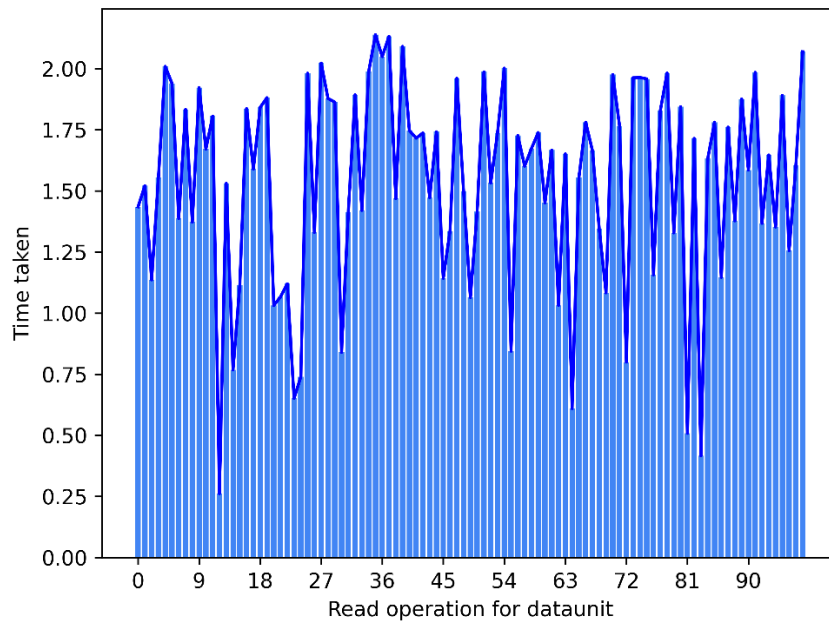
Кількість узгоджених вузлів. 2000 запитів на запис, 1000 запитів на читання



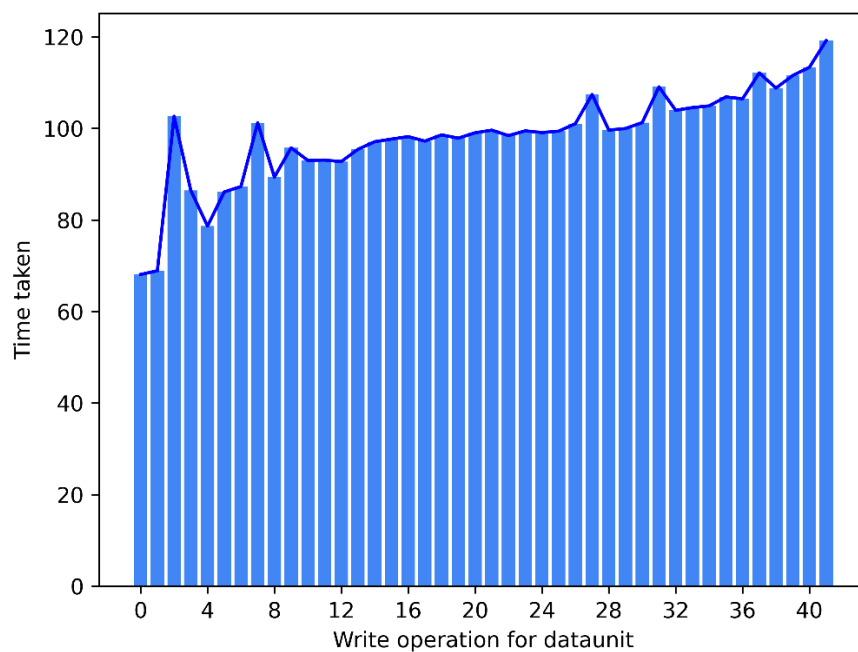
Кількість узгоджених вузлів. 1000 запитів на запис, 2000 запитів на читання

ДОДАТОК Ж**РЕЗУЛЬТАТИ ВИМІРЮВАНЬ ПРОСТОРОВОЇ ТА ЧАСОВОЇ
СКЛАДНОСТІ У СИМУЛЯЦІЇ МЕТОДУ КОНТРОЛЬОВАНОГО
БАЛАНСУВАННЯ УЗГОДЖЕНОСТІ**

Перша реалізації алгоритму: балансування на шлюзі

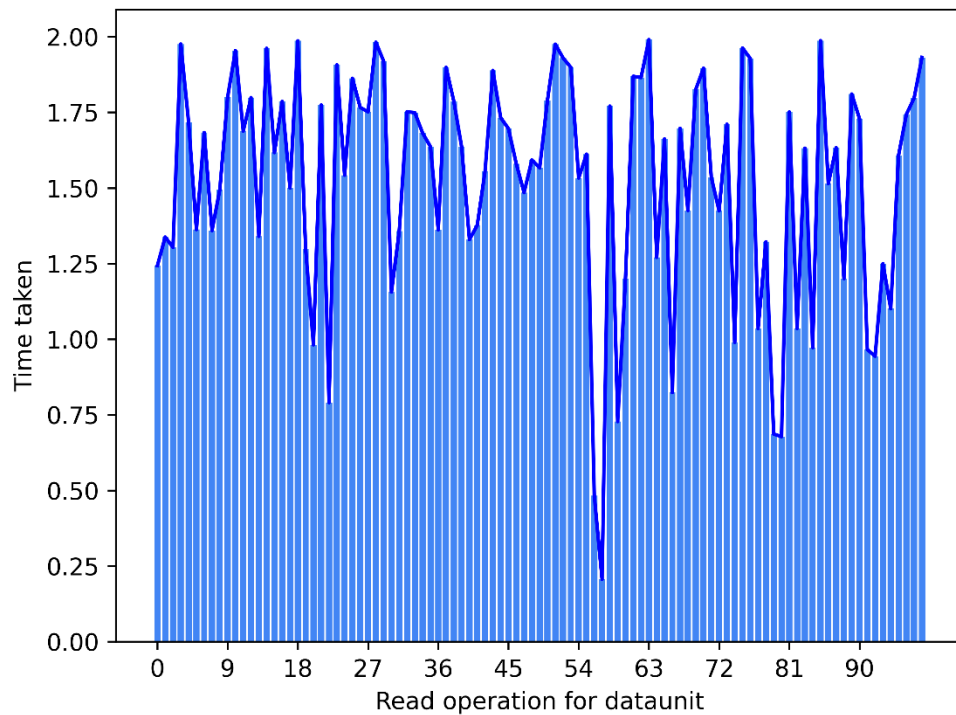


Оцінка часу обробки запита на читання. 100 операцій, 1000 вузлів

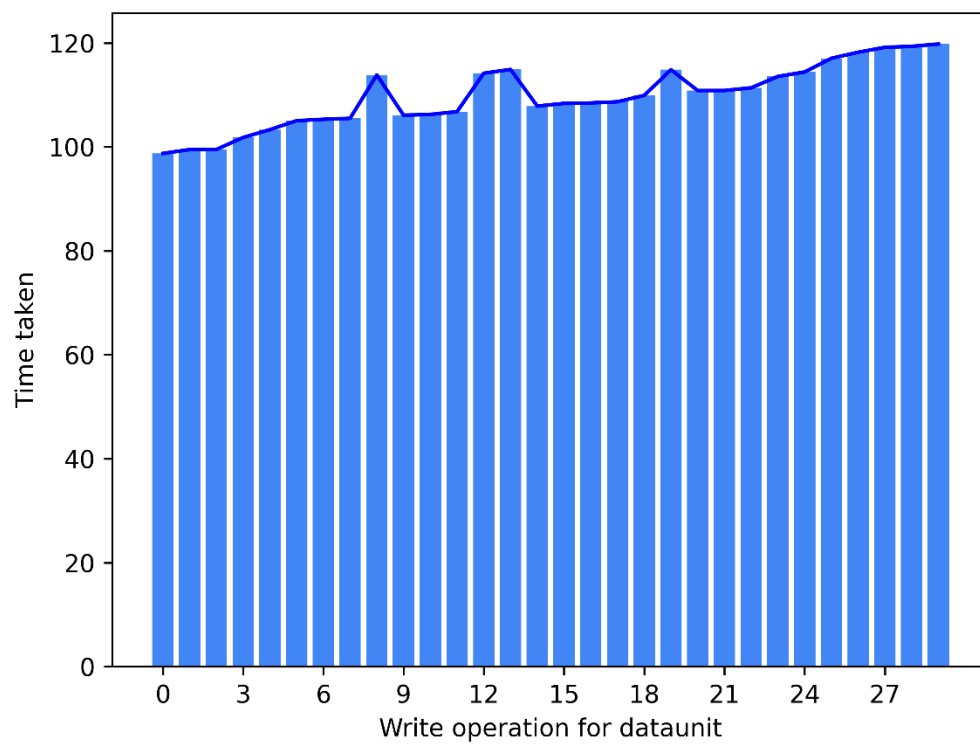


Оцінка часу обробки запита на запис. 40 операцій, 100 вузлів

Друга реалізація алгоритму: балансування на вузлі



Оцінка часу обробки запита на читання. 100 операцій, 1000 вузлів



Оцінка часу обробки запита на запис. 30 операцій, 100 вузлів