УДК 004.7

Млинський Б.М., здобувач,
Дячук О.Ю., ст. викладач,
Колощук М.С., ст. викладач
Державний університет «Житомирська політехніка»

ОСОБЛИВОСТІ ЗАБЕЗПЕЧЕННЯ ВІДМОВОСТІЙКОСТІ В РОЗПОДІЛЕНИХ МЕРЕЖЕВИХ СИСТЕМАХ: АНАЛІЗ ПРОБЛЕМИ SPLIT-BRAIN ТА ПОРІВНЯННЯ З АВТОНОМНИМИ АРХІТЕКТУРАМИ

Сучасні розподілені мережеві системи формуються як розподілені обчислювальні середовища, що складаються з великої кількості взаємопов'язаних вузлів і застосовуються у хмарних інфраструктурах, дата-центрах та сервісно-орієнтованих архітектурах. Однією з ключових вимог до таких систем є забезпечення відмовостійкості (fault tolerance), яка визначає здатність системи зберігати працездатність за умов часткових відмов окремих компонентів.

Метою роботи є аналіз причин виникнення проблеми split-brain у розподілених системах та порівняння підходів до забезпечення відмовостійкості з автономними архітектурами.

На відміну від автономних або централізованих архітектур, де існує єдина точка відмови (single point of failure), розподілені системи орієнтовані на те, що відмови є невід'ємною частиною їх функціонування. Досягнення відмовостійкості базується на використанні надмірності (redundancy), реплікації даних (replication) та механізмів автоматичного відновлення.

Реплікація дозволяє зберігати кілька копій даних на різних вузлах системи, що забезпечує їх доступність навіть у випадку відмови окремих компонентів. При цьому застосовуються різні моделі узгодженості, зокрема strong consistency, коли всі копії даних підтримуються в узгодженому стані в режимі реального часу, та eventual consistency, яка допускає тимчасові розбіжності стану системи з метою підвищення продуктивності та зменшення затримок [3].

Фундаментальним обмеженням розподілених систем є теорема CAP, відповідно до якої система не може одночасно гарантувати consistency, availability та partition tolerance. Оскільки мережеві розділення є неминучими у великих мережах, практичні реалізації змушені обирати компроміс між узгодженістю та доступністю. У цьому контексті виникає проблема split-brain, яка є одним із найбільш небезпечних сценаріїв відмов [3].

Split-brain виникає у випадку мережевого розділення (network partition), коли кластер поділяється на кілька ізольованих частин, кожна

з яких продовжує функціонувати незалежно. У таких умовах може виникнути ситуація з множинними керівними вузлами (multiple leaders), коли кілька вузлів одночасно виконують роль координатора. Це пов’язано з тим, що кожен вузол приймає рішення на основі локального уявлення про стан системи (local view), яке може бути неповним або застарілим через затримки передачі даних або втрату пакетів [5].

Наслідки split-brain є критичними для систем, що працюють з транзакційними або консистентними даними. Незалежна обробка запитів у різних частинах системи призводить до конфліктів запису (write conflicts) та розбіжності даних (data divergence). Після відновлення мережевого з’єднання виникає необхідність узгодження станів (reconciliation), що є складним процесом і може супроводжуватися частковою втратою інформації або порушенням логічної цілісності даних. Для вирішення таких конфліктів використовуються спеціалізовані підходи, включаючи політики на кшталт last-write-wins або більш складні механізми, такі як CRDT, однак вони не завжди гарантують коректність з точки зору прикладної логіки [1].

Для мінімізації ризику виникнення split-brain застосовуються алгоритми досягнення згоди (consensus algorithms), серед яких найбільш відомими є Paxos та Raft. Вони забезпечують узгоджене прийняття рішень між вузлами навіть у разі часткових відмов. Основою їх роботи є використання кворуму (quorum), тобто рішення вважається прийнятим лише за умови підтримки більшістю вузлів. Це дозволяє гарантувати, що лише одна частина системи може виконувати операції запису, тоді як інші вузли переходять у пасивний стан.

Додатково використовуються механізми ізоляції вузлів, зокрема fencing, які запобігають одночасному доступу кількох вузлів до спільних ресурсів. У практичних реалізаціях застосовується підхід STONITH, що передбачає примусове відключення вузла для уникнення конфліктів. Важливу роль також відіграють механізми контролю стану вузлів, зокрема heartbeat, які забезпечують регулярну перевірку доступності компонентів системи. Однак такі механізми не є абсолютно надійними, оскільки не дозволяють точно відрізнити реальну відмову від тимчасових мережевих затримок, що може призводити до помилкових рішень [2].

У порівнянні з розподіленими системами, автономні архітектури мають значно простішу структуру. Всі операції виконуються в межах одного вузла або контрольованого середовища, що забезпечує високу узгодженість даних без необхідності складних механізмів синхронізації. Відсутність розподіленої координації спрощує проєктування системи та знижує ризик виникнення логічних конфліктів [4].

Водночас такі системи мають суттєвий недолік у вигляді єдиної точки відмови. У разі виходу з ладу центрального вузла система повністю втрачає працездатність. Навіть при використанні резервних механізмів перемикання можливі затримки у відновленні роботи та ризик втрати даних. Крім того, централізовані системи мають обмежені можливості масштабування, що обмежує їх застосування у великих обчислювальних середовищах [1].

Проведений теоретичний аналіз сучасних підходів до забезпечення відмовостійкості дозволив узагальнити ключові особливості функціонування розподілених систем в умовах мережевих розділень.

У результаті проведеного аналізу встановлено, що проблема split-brain є наслідком фундаментальних обмежень розподілених систем, пов’язаних із необхідністю балансування між узгодженістю та доступністю. Порівняння з автономними архітектурами показало, що підвищення відмовостійкості досягається ціною ускладнення механізмів координації та управління даними. Найбільш ефективним підходом до запобігання split-brain є поєднання алгоритмів узгодження, кворумних механізмів та засобів ізоляції вузлів.

Подальший розвиток цієї галузі пов’язаний із дослідженням адаптивних систем, які здатні динамічно змінювати параметри узгодженості залежно від стану мережі, а також із розробкою нових моделей обробки даних, що дозволяють природно обробляти конфлікти без значного ускладнення системи. Це відкриває перспективи створення більш надійних і масштабованих мережевих систем, здатних ефективно функціонувати в умовах постійних змін і часткових відмов [1].

Список використаних джерел

1. Assiri B., Sheneamer A. Fault tolerance in distributed systems using deep learning approaches // PLOS ONE. 2024. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0310657

1. Sari A., Akkaya M. Fault tolerance mechanisms in distributed systems // International Journal of Computer Applications. 2015. Vol. 8. URL: https://www.researchgate.net/publication/287198069_Fault_Tolerance_Mechanisms_in_Distributed_Systems

2. Fault tolerance in distributed systems: a survey. 2019. URL: https://www.researchgate.net/publication/330123117_Fault_Tolerance_in_Distributed_Systems_A_Survey

3. Liu S. A survey on fault-tolerance in distributed optimization and machine learning. 2021. URL: https://www.semanticscholar.org/paper/A-Survey-on-Fault-tolerance-in-Distributed-and-Liu/972be47cdd1b43f2f8245a8e34efcd6d7f251c4f

4. Brewer E. Towards robust distributed systems. 2000. URL: https://www.researchgate.net/publication/221343719_Towards_robust_distributed_systems