https://doi.org/10.31891/2219-9365-2026-85-37
УДК 004.75

КРЕЩУК Владислав
Хмельницький національний університет
https://orcid.org/0009-0000-2691-5450
e-mail: vladyslavkreshchuk@gmail.com
ЛИГУН Олексій
Хмельницький національний університет
https://orcid.org/0009-0004-5727-5096
e-mail: oleksii.lyhun@gmail.com
СОРОЧИНСЬКИЙ Олександр
Хмельницький національний університет
https://orcid.org/0009-0003-7966-4861
e-mail: sorochinskiy159@gmail.com

МЕТОД УНИКНЕННЯ ВЗАЄМОБЛОКУВАНЬ ЗАВДАНЬ В РОЗПОДІЛЕНИХ СИСТЕМАХ НА ОСНОВІ ПРОТОКОЛУ МІЖПРОЦЕСНОЇ ВЗАЄМОДІЇ

У статті представлено результати розроблення та дослідження методу уникнення взаємоблокувань завдань у розподілених обчислювальних системах, що ґрунтується на інтеграції подієво-орієнтованої моделі виконання з формалізованим протоколом міжпроцесної взаємодії. Запропонований підхід формує цілісну методологічну основу керування розподіленими обчисленнями, у якій реактор розглядається як центральний координаційний механізм узгодження структурних залежностей між задачами та процедур доступу до ресурсів. Ключовим результатом є забезпечення структурної відсутності циклів очікування за рахунок поєднання алгоритмів виявлення потенційних конфліктів із адаптивним протоколом координації.

У роботі сформовано комплексну систему метрик, що дозволяє кількісно оцінювати стан вузлів, параметри потоків і рівень ресурсного навантаження, що, у свою чергу, створює основу для обґрунтованого вибору стратегій планування, балансування та масштабування. Експериментальні результати, представлені у графічній та табличній формах, підтверджують збереження лінійної динаміки продуктивності при збільшенні кількості вузлів і прогнозоване зростання накладних витрат без втрати стійкості системи. Доведено, що інтеграція формальної моделі залежностей із протоколом обміну повідомленнями гарантує просування критичних шляхів виконання навіть у гетерогенних середовищах.

Отримані результати свідчать про практичну придатність запропонованого методу для побудови масштабованих і надійних розподілених систем, у яких поєднуються вимоги високої ефективності використання ресурсів і стійкості до взаємоблокувань. Перспективи подальших досліджень пов'язані з адаптацією підходу до різнорідних архітектур вузлів і розширених конфігурацій із великою кількістю компонентів.

Ключові слова: розподілена система, комп'ютерна система, взаємоблокування, процеси, протокол, завдання, реактор.

KRESHCHUK Vladyslav, LYHUN Oleksii, SOROCHYNSKYI Oleksandr
Khmelnytskyi National University

METHOD OF AVOIDING INTERLOCKS OF TASKS IN DISTRIBUTED SYSTEMS BASED ON INTERPROCESS INTERACTION PROTOCOL

The article presents the results of the development and research of the method of avoiding mutual blocking of tasks in distributed computing systems, which is based on the integration of an event-oriented model of execution with a formalized protocol of interprocess interaction. The proposed approach forms a holistic methodological basis for managing distributed computing, in which the reactor is considered as a central coordination mechanism for coordinating structural dependencies between tasks and resource access procedures. The key result is ensuring the structural absence of waiting cycles due to the combination of algorithms for detecting potential conflicts with an adaptive coordination protocol.

In the work, a complex system of metrics was formed, which allows to quantitatively evaluate the state of nodes, flow parameters and the level of resource load, which, in turn, creates a basis for a reasonable choice of planning, balancing and scaling strategies. The experimental results, presented in graphic and tabular forms, confirm the preservation of the linear dynamics of performance with an increase in the number of nodes and the predicted increase in overhead costs without loss of system stability. Integrating a formal dependency model with a messaging protocol has been proven to guarantee the advancement of critical execution paths even in heterogeneous environments.

The obtained results indicate the practical suitability of the proposed method for building scalable and reliable distributed systems that combine the requirements of high efficiency of resource use and resistance to mutual blocking. Prospects for further research are related to the adaptation of the approach to heterogeneous node architectures and advanced configurations with a large number of components.

Keywords: distributed system, computer system, interlocking, processes, protocol, task, reactor.

Стаття надійшла до редакції / Received 20.12.2025
Прийнята до друку / Accepted 16.01.2026
Опубліковано / Published 05.03.2026

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

З кожним роком стає дедалі складніше підвищувати обчислювальну потужність окремої машини виключно за рахунок збільшення тактової частоти процесора чи кількості ядер, то сучасні інформаційні технології активно переходять до використання паралельних і розподілених систем [1, 2]. Такий підхід дозволяє досягти суттєвого приросту продуктивності шляхом декомпозиції складної задачі на менші підзадачі, які можуть виконуватися одночасно на кількох обчислювальних вузлах або процесах. Розподілені обчислення відкривають широкі можливості для масштабування сервісів, підвищення відмовостійкості та ефективного використання ресурсів, що є особливо актуальним в умовах стрімкого розвитку хмарних технологій, великих даних та високонавантажених систем [3, 4].

Водночас перехід від послідовної моделі виконання до паралельної та розподіленої суттєво ускладнює процес проєктування, реалізації та супроводу програмного забезпечення. Якщо в послідовних системах основна увага приділяється алгоритмічній оптимізації та ефективному використанню пам'яті, то в паралельних середовищах додаються проблеми синхронізації, координації доступу до спільних ресурсів, забезпечення узгодженості даних і коректної міжпроцесної взаємодії. Неправильна організація обміну повідомленнями або керування ресурсами може призвести до критичних збоїв у роботі системи, які складно виявити та усунути.

Однією з найбільш небезпечних і водночас поширених проблем у паралельних і розподілених системах є взаємоблокування. Воно визначається як стан, у якому виконання програми повністю або частково зупиняється через те, що система не може надати необхідні ресурси для продовження роботи процесів, або ж виникає циклічна залежність між ними. У такій ситуації кожен із процесів очікує на ресурс, який утримується іншим процесом, що, своєю чергою, також перебуває в стані очікування. У результаті жоден із процесів не може продовжити виконання, а система переходить у стан блокування.

Особливу складність становить те, що в розподілених системах взаємоблокування можуть виникати не лише через спільне використання апаратних ресурсів, а й через некоректну логіку протоколів обміну повідомленнями, затримки в мережі або порушення порядку виконання операцій. Виявлення таких ситуацій ускладнюється відсутністю централізованого контролю та необхідністю аналізу глобального стану системи. Наслідками взаємоблокувань можуть бути зниження продуктивності, втрата доступності сервісів, порушення цілісності даних та фінансові збитки.

У зв'язку з цим особливої актуальності набуває розробка ефективних методів запобігання виникненню взаємоблокувань ще на етапі проєктування системи. На відміну від підходів, що передбачають лише виявлення та усунення взаємоблокувань після їх виникнення, методи уникнення дозволяють організувати взаємодію процесів таким чином, щоб виключити саму можливість формування циклічних залежностей. Одним із перспективних напрямів у цьому контексті є використання формалізованих протоколів міжпроцесної взаємодії, які регламентують порядок обміну повідомленнями, виділення та звільнення ресурсів, а також узгодження станів процесів.

Таким чином, актуальність роботи зумовлена необхідністю підвищення надійності та ефективності розподілених систем шляхом запобігання взаємоблокуванням.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Розподілені системи стають дедалі поширенішими в сучасному інформаційному просторі, оскільки обсяги даних та інтенсивність обчислювальних навантажень постійно зростають, а підвищення продуктивності однієї окремої машини стає економічно та технічно менш доцільним. Масштабування «вгору» поступово поступається масштабуванню «в ширину», коли обчислювальні ресурси нарощуються шляхом об'єднання багатьох вузлів у єдину систему. Такий підхід дозволяє досягати високої продуктивності, гнучкості та відмовостійкості, однак водночас породжує низку нових викликів, які відсутні або менш виражені у традиційних послідовних виконаннях [5, 6].

У розподіленому середовищі виконання завдань відбувається паралельно на кількох вузлах, які взаємодіють між собою через мережу. Це створює додаткові складнощі, пов'язані з координацією доступу до ресурсів, синхронізацією станів, затримками передавання повідомлень і частковою відмовою окремих компонентів. Однією з критичних проблем є ситуація, коли система більше не може просуватися вперед через те, що завдання були розподілені таким чином, що вони взаємно блокують одне одного [7, 8]. У такому випадку процеси перебувають у стані очікування ресурсів або повідомлень, які ніколи не будуть отримані, що призводить до повної або часткової зупинки системи.

Зі збільшенням масштабу та складності розподілених систем зростає й складність виявлення та аналізу таких станів. Відсутність централізованого контролю, асинхронність обміну даними та динамічний характер розподілу ресурсів ускладнюють своєчасне врахування можливості виникнення глухого кута ще на етапі проєктування. У результаті навіть незначні помилки в логіці планування або розподілу ресурсів можуть мати суттєві наслідки для всієї системи.

У статті [9] було запропоновано схему диспетчеризації, що ґрунтується на принципах уникнення

взаємоблокувань та забезпечує гарантії живучості системи. Запропонований підхід спрямований на те, щоб система ніколи не переходила в стан взаємоблокування, навіть за умов інтенсивного розподілу задач і ресурсів. Вона поєднує механізми планування та контролю залежності між процесами, забезпечуючи формальні гарантії коректності виконання. Ця робота розпочинається з детального аналізу підходу та його теоретичних засад. Зокрема, розглядається певний вимір проблеми від розподілу сайтів між вузлами до формально визначеного з прямими наслідками для здійсненності та ефективності системи. Уточнення цього аспекту дозволяє глибше зрозуміти обмеження моделі та визначити умови, за яких гарантії уникнення взаємоблокувань зберігаються. В роботі пропонується конкретна стратегія розподілу вузлів на сайти в однорідному середовищі, де всі обчислювальні ресурси мають подібні характеристики. На основі цієї моделі здійснюється узагальнення для гетерогенного середовища, в якому вузли можуть відрізнятися за продуктивністю, обсягом пам'яті чи мережними параметрами. Окрему увагу приділено можливостям оптимізації ефективності в межах запропонованих розподілів без порушення гарантій живості системи. У роботі також представлено відповідні евристики планування, які дозволяють зменшити накладні витрати на координацію процесів та покращити балансування навантаження. Завершальним етапом є розробка моделі динамічної підграфової доцільності, що базується на реалізації цієї моделі і враховує зміну стану системи в реальному часі. Поєднання здійсненності, яку забезпечує стратегія розподілу реакторів, із гарантіями уникнення взаємоблокування та максимальної живучості системи формує цілісне й оптимальне середовище для ефективного розподіленого виконання складних обчислювальних задач.

У складних системах розподілу ресурсів [10, 11], зокрема в автоматизованих виробничих системах, гнучких виробничих модулях, логістичних комплексах, а також у високонавантажених програмно-апаратних середовищах, ефективне управління потоками завдань є критично важливим для забезпечення безперервності технологічних процесів. Використання багатопотокових і паралельних програм дозволяє значно підвищити продуктивність системи, скоротити час виконання операцій, оптимізувати використання обладнання та зменшити енергоспоживання. Завдяки одночасному виконанню декількох процесів досягається краща утилізація ресурсів, знижується час простою машин і підвищується загальна пропускна здатність системи [12].

Проте впровадження паралелізму неминуче супроводжується появою додаткових ризиків і технічних ускладнень. До найбільш поширених проблем належать умови конкуренції, некоректна синхронізація доступу до спільних ресурсів, взаємоблокування, а також ситуації голодування [13, 14]. Такі явища можуть суттєво знижувати продуктивність системи, призводити до втрати узгодженості даних, збільшення часу простою обладнання та навіть створювати загрозу безпеці виробничих процесів. Особливо критичними ці наслідки є для автоматизованих виробничих систем, де порушення логіки керування може спричинити не лише економічні збитки, а й аварійні ситуації. Саме тому питання діагностики несправностей, виявлення причин блокувань, розробки методів їх усунення та запобігання взаємоблокуванням у паралельних програмах викликають є актуальними [15, 16].

З метою уникнення тупікових ситуацій у складних динамічних системах активно застосовується теорія дискретного керування [17, 18], яка забезпечує формальні засоби моделювання, аналізу та синтезу керуючих стратегій. Одним із найпотужніших інструментів у цій галузі є мережі Петрі, що завдяки своїй наочності та високій виразній здатності широко використовуються для моделювання автоматизованих виробничих систем [19], багатопоточних програмних систем [20], систем обслуговування, транспортних мереж та інших прикладних задач [21]. Мережі Петрі дозволяють формалізувати паралельність, конкуренцію, синхронізацію та розподіл ресурсів у межах єдиної математичної моделі, що значно полегшує аналіз поведінки системи.

Таким чином, проблема синтезу ефективного, структурно простого й обчислювально прийнятного супервізора для великомасштабних розподілених систем залишається не вирішеною. Це зумовлює необхідність подальших досліджень, спрямованих на розробку нових методів керування, які поєднуватимуть гарантії живучості, відсутності взаємоблокувань та мінімальної структурної складності. Також, необхідно провести аналіз щодо дисперсії розмірів пулів потоків та її вплив на максимальну доцільність і здійсненність системи.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є забезпечення узгодженої координації процесів, мінімізація ризику виникнення циклічних залежностей та підвищення загальної стабільності функціонування системи шляхом розроблення методу уникнення взаємоблокувань завдань у розподілених системах на основі протоколу міжпроцесної взаємодії.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Моделювання взаємоблокувань завдань в розподілених системах на основі протоколу міжпроцесної взаємодії

Моделювання взаємоблокувань завдань в розподілених системах на основі протоколу міжпроцесної взаємодії здійснимо з використанням теорії графів та реакторів, що дають змогу відокремити події у вузлах.

Для загальних орієнтованих ациклічних графів (ОАГ), що моделюють графи викликів у розподілених системах, питання конфлікту при призначенні вузлів реакторам набуває принципового значення. На відміну від структур типу дерева, ОАГ допускають існування вузлів із кількома вхідними шляхами, що породжує потенційну неоднозначність у визначенні реактора для такого вузла. Якщо в дереві кожна вершина має єдиний шлях від кореня, то в ОАГ одна й та сама вершина може належати кільком шляхам різної довжини. Відтак виникає потреба у формальному механізмі визначення пріоритету між шляхами з метою забезпечення узгодженості розподілу.

Пропонована стратегія ґрунтується на принципі пріоритету довшого шляху. Інтуїтивно це відповідає орієнтації на критичний шлях, тобто на ту послідовність вузлів, яка визначає мінімальний можливий час завершення всього графа. Якщо вузол належить двом або більше шляхам, то реактор для нього визначається тим шляхом, який має більшу довжину. У випадку рівності довжин застосовується довільний, але детермінований порядок, що не впливає на фінальне забарвлення. Такий підхід забезпечує, що конфлікти вирішуються не локально, а з урахуванням глобальної структури графа.

Ключовою метою є збереження інваріанта нерозподілу, який формулюється як умова $\lceil L / n \rceil \geq \max(R(p))$ для будь-якого шляху p, де L - довжина критичного шляху, n - кількість реакторів, а max(R(p)) - максимальна кількість вузлів шляху p, призначених одному реактору. Цей інваріант гарантує, що якщо ресурси достатні для обслуговування критичного шляху, то стратегія розподілу не створить штучного вузького місця через невдале призначення реакторів.

Реалізаційно стратегію базуватимемо на двофазному алгоритмі. На першій фазі для кожного джерельного вузла ОАГ виконаємо DFS-обхід. Уздовж кожного шляху використовується циклічний лічильник із періодом n, що забезпечує рівномірне чергування реакторів. Кожному шляху присвоюється унікальний ідентифікатор, а його довжина накопичується як рекурсивний параметр. Результатом цієї фази є хеш-таблиця, що відображає ідентифікатори шляхів у їхню довжину.

На другій фазі здійснюється повторний обхід графа, під час якого кожна вершина отримує реактор відповідно до того шляху, що має максимальну довжину серед усіх шляхів, які проходять через цю вершину. Завдяки доведеному факту, що шляхи однакової довжини можна впорядковувати довільно без зміни кінцевого забарвлення, алгоритм є детермінованим і не залежить від конкретного порядку обходу.

Асимптотична складність процедури у найгіршому випадку становить $O(|V|(|V|+|E|))$, оскільки алгоритм запускається з кожного джерела. Додатковий обхід та операції з хеш-таблицею не змінюють порядку складності. Важливо, що алгоритм може виконуватися офлайн, під час етапу завантаження або компіляції графа, що усуває його вплив на час виконання системи.

Доведення коректності базується на аналізі можливих спроб порушення інваріанта нерозподілу. Припустимо, що існує шлях p, для якого можна сконструювати набір шляхів p_i, що перезаписують призначення його вершин так, щоб max(R(p)) перевищив допустиму межу. Для цього кожен шлях p_i повинен бути довшим за p, оскільки коротший або рівний шлях не може змінити забарвлення через правило пріоритету. Більше того, щоб змінити колір вершини, довжина нового шляху має перевищувати попередню щонайменше на n (через циклічність мод n). Це означає, що кожна така ітерація збільшує L щонайменше на n. Відтак будь-яка спроба примусового перезапису неминуче призводить до зростання L, що зберігає виконання інваріанта. Таким чином, не існує конструкції, яка могла б порушити інваріант без одночасного збільшення критичного шляху.

Для систем, близьких до межі здійсненності, випадковий розподіл дедалі частіше порушує інваріант нерозподілу, особливо зі збільшенням кількості реакторів. Натомість запропонована стратегія не демонструє жодного порушення в експериментальних серіях. Це свідчить про те, що стратегічний, структурно обґрунтований розподіл є не лише оптимізаційною перевагою, а необхідною умовою забезпечення коректності у масштабованих розподілених системах.

Отже, запропонований підхід забезпечує формально доведену узгодженість призначення реакторів у довільних ОАГ, зберігає інваріант нерозподілу та гарантує пріоритет критичного шляху. Це створює теоретичну основу для побудови відмовостійких і продуктивних механізмів планування в розподілених середовищах.

Реактор у контексті графа викликів або обчислювального графа походить із архітектурного шаблону Reactor pattern та є одним із базових підходів до побудови високопродуктивних подієво-орієнтованих систем. У класичному розумінні реактор це механізм організації обробки подій, що поєднує мультиплексування джерел подій, їх диспетчеризацію та виконання відповідних обробників у межах контрольованого середовища виконання. Однак у дослідницькому контексті розподілених і паралельних систем це поняття набуває розширеного змісту та трансформується в абстрактну модель керування виконанням залежних задач.

У моделі графа викликів реактор розглядається як логічна одиниця виконання, якій призначається підмножина вузлів графа. Вузли інтерпретуються як окремі обчислювальні операції або задачі, а ребра як відношення залежності чи причинно-наслідкового виклику. Таким чином, реактор виступає абстрактним ресурсом планування, що визначає середовище, у якому виконується відповідна група вузлів. У фізичній реалізації реактор може відповідати окремому потоку, пулу потоків або ізольованому циклу обробки подій,

проте в теоретичній моделі він трактується як ресурс із власною чергою задач і обмеженою пропускною здатністю. Запровадження цієї абстракції є необхідним для формалізації розподілу паралелізму та контролю конкуренції в розподілених системах. Відсутність глобального часу, асинхронність обміну повідомленнями, часткові відмови та варіативність затримок мережі створюють передумови для виникнення взаємоблокувань. Якщо кілька логічно залежних вузлів виконуються в одному середовищі без чітко визначеної дисципліни планування, можливе утворення циклічних очікувань. Призначення вузлів різним реакторам дозволяє структуровано розмежувати області виконання, мінімізувати локальну конкуренцію за ресурси та формалізувати умови, за яких неможливе виникнення циклічного очікування. Саме тому введення поняття реактора є методологічною передумовою розроблення методу уникнення взаємоблокувань у розподілених системах.

З формальної точки зору реактор задамо функцією R: V → {r_1, r_2, …, r_n}, де V - множина вузлів графа, а {r_1, …, r_n} - множина реакторів. Це відображення задає розбиття графа на підмножини, кожна з яких виконується в межах окремого середовища планування. На відміну від класичного розфарбовування графа, де кольори мають лише топологічне значення, реактори мають операційний сенс. Вони відповідають конкретним обчислювальним ресурсам, що впливають на часові характеристики виконання, довжину критичного шляху та можливість блокувань. Коректно вибрана стратегія відображення дозволяє обмежити максимальну кількість послідовних вузлів одного шляху, що виконуються в одному реакторі, тим самим знижуючи ризик локального накопичення залежності. На рис. 1 зображено приклад з використанням реактора.

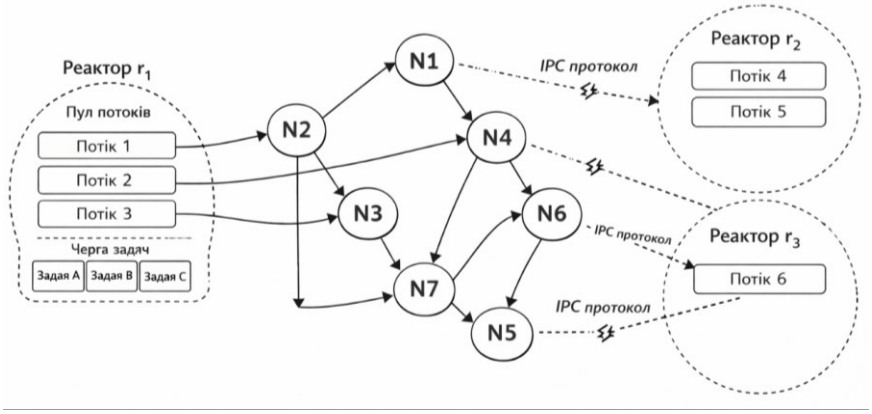

Рис. 1. Приклад графа з реактором

Поданий граф на рис. 1 є прикладом формалізованої моделі виконання задач у розподіленій системі з використанням реакторної архітектури. Він поєднує дві сутності, а саме: граф залежності (граф викликів); множину реакторів як ізольованих середовищ виконання, пов'язаних протоколом міжпроцесної взаємодії (IPC). Така структура дозволяє одночасно описувати логіку обчислень і механізм їх фізичного виконання.

У центрі схеми на рис. 1 розташований граф викликів, що складається з вузлів N1–N7 та орієнтованих ребер між ними. Вузол інтерпретується як елементарна обчислювальна операція, сервісний виклик або транзакційний крок. Орієнтоване ребро відображає залежність типу «після виконання A може бути виконано B» або «B потребує результату A». Таким чином, граф задає частковий порядок виконання задач. Якщо розглядати його формально, то це орієнтований ациклічний граф (ОАГ), що визначає множину допустимих послідовностей виконання.

Ліва та права частини схеми відображають реактори r_1, r_2 та r_3. Кожен реактор представлений як ізольоване середовище з власним пулом потоків і чергою задач. Пул потоків визначає внутрішній рівень паралелізму, тоді як черга задач задає дисципліну планування. Реактор отримує задачі (вузли графа), обробляє їх відповідно до внутрішнього алгоритму планування та може ініціювати взаємодію з іншими реакторами через IPC-протокол.

Пунктирні стрілки між вузлами графа на рис. 1 та іншими реакторами позначають міжпроцесну взаємодію. Це означає, що виконання певного вузла вимагає відправлення повідомлення до іншого реактора та очікування відповіді. Саме в цій точці логічний граф залежностей перетворюється на мережу ресурсних очікувань.

Граф може описувати блокування в розподілених системах через накладання двох структур, а саме: графа залежності; графа ресурсного очікування. Якщо вузол N4 у реакторі r_1 надсилає запит до реактора r_2 і блокується до отримання відповіді, а в той самий час у r_2 виконується вузол, який потребує результату з r_1, то виникає циклічне очікування між реакторами. Формально це означає, що в системі формується цикл у графі очікування ресурсів, навіть якщо первинний граф викликів є ациклічним.

Блокування можливе через кілька механізмів. По-перше, через обмеженість потоків у межах реактора. Якщо всі потоки зайняті задачами, що очікують відповіді від інших реакторів, то нова задача, яка могла б

розірвати цикл залежностей, не буде запущена. По-друге, через синхронну модель IPC, коли реактор утримує ресурс під час очікування відповіді. По-третє, через конкуренцію за спільні логічні ресурси, які не відображені явно у графі викликів, але проявляються у виконанні.

У такій моделі виділимо два типи графів так:

1) граф причинно-наслідкових залежностей;

2) граф очікування ресурсів, який формується динамічно під час виконання.

Блокування виникає тоді, коли у графі очікування ресурсів з'являється цикл. Представлена схема демонструє, як такий цикл може виникнути між реакторами через IPC-виклики. Якщо призначення вузлів реакторам здійснюється за визначеною стратегією, наприклад, циклічним розподілом із пріоритетом критичного шляху, тоді можна формально довести, що цикл очікування не утворюється або що завжди існує хоча б один реактор, здатний продовжити виконання. Тобто реакторна декомпозиція графа дозволяє контролювати структуру можливих ресурсних конфліктів.

Отже, даний граф є формальною моделлю взаємодії обчислювальних задач і ресурсів у розподіленому середовищі. Він демонструє, як логічні залежності трансформуються у ресурсні очікування, і показує механізм виникнення взаємоблокувань через циклічні міжреакторні залежності. Саме тому така модель є придатною для аналізу коректності протоколів міжпроцесної взаємодії та розроблення методів гарантування прогресу виконання.

Ключовим доповненням до цієї моделі є протокол міжпроцесної взаємодії, який визначає правила обміну повідомленнями між реакторами. Оскільки вузли графа, призначені різним реакторам, можуть мати залежності, то їх взаємодія відбувається через передачу подій або повідомлень. Протокол міжпроцесної взаємодії встановлює порядок ініціювання викликів, підтвердження завершення операцій, обробки помилок і тайм-аутів. У контексті уникнення взаємоблокувань важливо, щоб цей протокол виключав можливість двостороннього синхронного очікування між реакторами. Це досягається шляхом використання асинхронної моделі запит–відповідь, односторонніх повідомлень або дисципліни «неблокуючого виклику», за якої реактор не утримує ресурс під час очікування відповіді від іншого реактора.

Таким чином, реактор виступає не лише абстрактною одиницею обчислювального ресурсу, а й елементом формальної моделі, що поєднує структуру графа залежностей із протоколом міжпроцесної взаємодії. Саме інтеграція стратегії розподілу реакторів із коректно визначеним протоколом обміну повідомленнями створює основу для побудови методу уникнення взаємоблокувань. Вона дозволяє довести, що за певних умов призначення вузлів та правил взаємодії між реакторами система не утворює циклів очікування, а отже, забезпечує гарантовану можливість просування виконання критичного шляху навіть у складних розподілених середовищах.

Метрики для організації оцінювання виконання потоків в розподілених системах

Визначимо підходи до організації роботи систем із гетерогенними серверами швидкості. У дослідженні високопродуктивних обчислювальних систем особливу увагу приділяють задачі ефективного розподілу потоків між серверами з різними швидкісними характеристиками. Такі системи називають гетерогенними, оскільки їхні обчислювальні вузли мають відмінні параметри продуктивності. Проблема оптимального розподілу ресурсів у цьому контексті може бути формалізована через два базові підходи, кожен з яких відповідає різним початковим умовам конфігурації системи. Визначимо два підходи в залежності від кількості потоків в розподілених системах.

Перший підхід до розподілу потоків визначимо з урахуванням того, що система функціонує з фіксованою кількістю потоків. Тоді, у межах першого підходу вважатимемо, що задано множину серверів із різними швидкостями обробки та загальну кількість потоків (S), які необхідно розподілити між серверами. І на основі такого початкового вибору задамо моделі. Нехай множину швидкостей серверів визначимо так:

$$M_H = \left\{m_{H,1}, m_{H,2}, \dots, m_{H,N_{M_H}}\right\}, \tag{1}$$

де $m_{H,i}$ - швидкість і-го сервера; N_{M_H} – кількість серверів; $i = 1,2, \dots, N_{M_H}$.

З метою коректного порівняння продуктивності серверів проведемо нормалізацію відносно найбільшого спільного дільника для елементів множини M_H так:

$$q_H = f_H(m_{H,1}, m_{H,2}, \dots, m_{H,N_{M_H}}), \tag{2}$$

де f_H – функція знаходження найменшого за значенням елементу серед елементів множини M_H.

Тоді визначаємо коефіцієнти відносної швидкості серверів так:

$$b_i = \frac{m_{H,i}}{q_H}, \tag{3}$$

де $m_{H,i}$ - швидкість і-го сервера; N_{M_H} – кількість серверів; $i = 1,2, \dots, N_{M_H}$; q_H - найменший за значенням елемент серед елементів множини M_H.

Сумарний коефіцієнт відносної швидкості серверів дає змогу оцінити частку кожного сервера в загальній продуктивності розподілених систем. Визначимо його так:

$$b = \sum_{i=1}^{N_{M_H}} b_i, \tag{4}$$

де b_i - коефіцієнти відносної швидкості і-го сервера; N_{M_H} – кількість серверів; $i = 1,2, \dots, N_{M_H}$.

Тоді частка $\frac{b_i}{b}$ буде відображати частку загальної продуктивності системи, що припадає на (і)-й сервер.

Розподіл потоків здійснимо за такою схемою:

1) первинний розподіл потоків виконуватимемо пропорційно відносним швидкостям:

$$V_i = V \cdot \frac{b_i}{b}; \tag{5}$$

2) оскільки використовується операція округлення, то виникає залишок потоків:

$$r = V - \sum_{i=1}^{N_{M_H}} V_i, \tag{6}$$

3) потрібно здійснити перевірку щодо коректності отриманого результату так:

$$r < N_{M_H}, \tag{7}$$

де b_i - коефіцієнти відносної швидкості і-го сервера; N_{M_H} – кількість серверів; $i = 1,2, \dots, N_{M_H}$.

Згідно формули (7) похибка округлення кожного доданка не може бути більшою за одиницю. Якщо нерівність не виконується, тоді в системі наявна проблема. Залишкові потоки розподіляються ітеративно, починаючи з серверів із найбільшою швидкістю, що мінімізує очікуваний час виконання.

Така схема дає змогу редукувати гетерогенну систему до еквівалентної однорідної моделі шляхом уявного "розщеплення" кожного сервера на b_i віртуальних підсерверів із однаковими характеристиками. Це створює аналітичну можливість застосування методів, які розроблені для однорідних систем, при збереженні максимальної заповненості та балансування навантаження.

За другим підходом розглядатимемо розподіл реакторів у системі з попередньо визначеними пулами потоків. Приймемо, що сервери мають різні швидкості і те, що кожен сервер уже оснащений пулом потоків однакового розміру. Завдання полягає у розподілі реакторів або обчислювальних вузлів між серверами таким чином, щоб зберігалася інваріантність здійсненності та враховувалися швидкісні відмінності серверів. Введемо метрики для фіксації цих дій в розподілених системах.

Аналіз заповненості системи

Нехай L - довжина критичного шляху, що вимірюється в потоках, S - загальна кількість потоків.

Якщо L= S, то система повністю заповнена і будь-яка спроба нерівномірного розподілу може порушити гарантії здійсненності.

Якщо ж L< S, то виникає резерв продуктивності, що дозволяє здійснити контрольований дисбаланс на користь швидших серверів.

Здійснимо формалізацію частот і періодів як метрик для процесів чи завдань в розподілених системах. Аналогічно до першого підходу визначимо так відносну частоту планування сервера:

$$f_i = \frac{b_i}{b}, \tag{8}$$

де b_i - коефіцієнти відносної швидкості і-го сервера; N_{M_H} – кількість серверів; $i = 1,2, \dots, N_{M_H}$; b - загальна продуктивності розподілених систем.

Ці значення f_i відображають відносну частоту планування сервера.

Період обслуговування сервера:

$$T_i = \frac{1}{f_i}, \tag{9}$$

де f_i - коефіцієнти відносної швидкості і-го сервера; N_{M_H} – кількість серверів; $i = 1,2, \dots N_{M_H}$.

Таким чином, швидші сервери мають менші періоди та повинні плануватися частіше.

Для формування простору можливих рішень використаємо експоненціальний розподіл періодів, адаптований не до розмірів пулів, а до швидкісного розподілу. Визначимо його вектором так:

$$T = (T_1, T_2, \dots, T_{N_{M_H}}), \tag{10}$$

де N_{M_H} – кількість серверів; $i = 1,2, \dots N_{M_H}$; T_i - період обслуговування і-го сервера.

Проте виникає конфлікт оптимізаційних критеріїв в контексті максимізації продуктивності і

забезпечення здійсненності.

Розглянемо забезпечення здійсненності в однорідному випадку. Оскільки всі сервери мають однаковий розмір пулу потоків, то перевірку здійсненності можна спростити. Для цього введемо мінімальний допустимий період так:

$$y = \frac{L}{S} + 1, \tag{11}$$

де L - довжина критичного шляху, що вимірюється в потоках, S - загальна кількість потоків.

Якщо всі періоди $T_i \geq y$, то гарантується відсутність конфліктів планування. Найменший період відповідає найшвидшому серверу. Обмеження мінімального періоду зменшує простір пошуку та дозволяє будувати лише гарантовано здійсненні розподіли.

Критерій оптимальності дає змогу перевіряти наближення поточного стану до ідеально можливого варіанту. Якість отриманого експоненціального розподілу оцінимо за ступенем наближення до ідеального швидкісного профілю $T = (T_1, T_2, \dots, T_{N_{M_H}})$ (формула (10)).

Чим ближчий фактичний розподіл до теоретично оптимального, за умови дотримання інваріантів здійсненності, тим вища сумарна продуктивність системи.

Обидва підходи демонструють різні стратегії адаптації до гетерогенності. Перший підхід орієнтований на пропорційний розподіл потоків за умови централізованого контролю. Другий підхід досліджує можливості контрольованого дисбалансу за умови збереження інваріантів здійсненності. Обидва підходи є основою формалізованої моделі, яка дозволяє одночасно враховувати неоднорідність обчислювальних ресурсів, мінімізувати час виконання, гарантувати коректність та стабільність функціонування системи.

Оцінювання виконання потоків у розподілених системах із гетерогенними серверами швидкості базується на системі кількісних метрик, що дозволяють формально описати продуктивність, баланс навантаження, ефективність використання ресурсів і якість обслуговування. У таких системах кожен сервер характеризується власною швидкістю обробки за параметром середньої кількості завдань або інструкцій, які сервер здатен виконати за одиницю часу. Якщо система складається з m серверів, то їхня сукупна теоретична продуктивність дорівнює сумі індивідуальних швидкостей, однак фактична продуктивність залежить від політики розподілу потоків і рівня завантаження.

Базовою метрикою є інтенсивність вхідного потоку завдань, що визначає середню кількість запитів або потоків, які надходять до системи за одиницю часу. Цей показник дозволяє визначити відповідність загальної обчислювальної потужності системи поточному навантаженню. Якщо інтенсивність надходження перевищує сумарну здатність серверів обробляти задачі, то у системі починають накопичуватися черги, що призводить до зростання затримок та зниження якості обслуговування. Тому першочерговим критерієм є забезпечення стабільності системи, за якої кожен сервер здатен обробляти закріплену за ним частину потоку без накопичення нескінченної черги.

Важливою метрикою є коефіцієнт завантаження окремого сервера. Він показує, яку частку свого потенціалу використовує вузол у конкретний момент часу. У гетерогенній системі одні сервери можуть працювати майже на межі своїх можливостей, тоді як інші залишаються частково недовантаженими. Такий дисбаланс негативно впливає на загальну ефективність. Тому при оцінюванні системи аналізують не лише середнє завантаження, а й рівномірність його розподілу між вузлами. Оптимальною вважається ситуація, коли навантаження розподіляється пропорційно швидкості серверів, тобто швидші вузли отримують більшу частку потоків.

Наступною суттєвою характеристикою є середній час перебування потоку в системі. Він охоплює як час очікування в черзі, так і безпосередній час обробки на сервері. У гетерогенних системах цей показник може істотно відрізнятися залежно від того, на який вузол потрапляє завдання. Якщо алгоритм планування не враховує різницю швидкостей, повільні сервери можуть створювати «вузькі місця», що збільшує загальну затримку. Тому середній час обробки є одним із ключових критеріїв оцінювання ефективності політики розподілу потоків.

Пропускна здатність системи відображає кількість задач, які завершуються за певний проміжок часу. У стабільному режимі вона приблизно дорівнює інтенсивності надходження запитів. Проте у випадку перевантаження фактична пропускна здатність може знижуватися через накопичення черг і збільшення часу очікування. У гетерогенному середовищі досягнення максимальної пропускної здатності залежить від здатності балансувальника навантаження враховувати різну продуктивність серверів. Якщо потоки розподіляються без урахування цих відмінностей, частина ресурсів може використовуватися неефективно.

Для аналізу рівномірності використання ресурсів застосовують метрики дисбалансу навантаження. Вони показують, наскільки відрізняється завантаження окремих серверів від середнього по системі. Чим менша ця різниця з урахуванням продуктивності вузлів, тим ефективніше організовано розподіл потоків. У гетерогенних системах абсолютна рівність завантаження не є ціллю; натомість важливо забезпечити пропорційність між швидкістю сервера та обсягом оброблюваних ним задач.

Ще однією вагомою метрикою є ефективність використання ресурсів. Вона відображає, яку частину свого потенціалу система реально використовує для виконання корисної роботи. Якщо сервери простоюють або працюють нерівномірно, загальна ефективність знижується. У сучасних розподілених середовищах керування контейнерами та мікросервісами, таких як Kubernetes, подібні показники використовуються для автоматичного масштабування та оптимального розміщення навантаження. У системах потокової обробки даних, наприклад у Apache Kafka, метрики продуктивності застосовуються для балансування партицій між брокерами та контролю швидкості обробки повідомлень.

Окремо аналізуватимемо показник прискорення, який дозволяє порівняти продуктивність гетерогенної системи з базовою конфігурацією, наприклад із виконанням на одному сервері. Ця метрика демонструє, наскільки ефективно система використовує паралелізм. В ідеальному випадку збільшення кількості серверів приводить до пропорційного скорочення часу виконання задачі, проте в реальних умовах існують накладні витрати на координацію, обмін повідомленнями та синхронізацію, що знижує фактичний виграш.

Крім того, важливою характеристикою є масштабованість системи. Вона показує, як змінюється продуктивність при зростанні кількості потоків або при додаванні нових серверів. У добре спроєктованій системі продуктивність зростає майже лінійно до досягнення апаратних або мережевих обмежень. У гетерогенному середовищі масштабованість ускладнюється різницею в швидкості вузлів і потребує адаптивних алгоритмів балансування.

Розглядаючи підходи до організації роботи систем залежно від кількості потоків, можна виділити два принципово різні режими. У першому випадку кількість потоків не перевищує кількість серверів або є близькою до неї. Тут головною метою є мінімізація часу завершення кожного окремого завдання. Раціональним рішенням стає призначення більш складних або ресурсоємних задач на швидші сервери. У другому випадку, коли потоків значно більше, ніж серверів, формується чергова модель обслуговування. Основний акцент переноситься на підтримання стабільності системи, мінімізацію середнього часу очікування та забезпечення високої пропускної здатності. У цьому режимі критично важливим стає пропорційний розподіл навантаження відповідно до продуктивності вузлів.

Отже, система метрик оцінювання виконання потоків у гетерогенних розподілених середовищах охоплює показники навантаження, затримки, пропускної здатності, ефективності, масштабованості та рівномірності використання ресурсів. Їхній комплексний аналіз дозволяє обґрунтовано вибирати алгоритми планування, політики балансування та стратегії масштабування, що особливо важливо в умовах зростаючої складності сучасних обчислювальних інфраструктур.

Метод уникнення взаємоблокувань завдань в розподілених системах на основі протоколу міжпроцесної взаємодії

У розподілених системах міжпроцесна взаємодія завдань відбувається за відсутності спільної глобальної пам’яті та єдиного фізичного годинника, що ускладнює координацію доступу до ресурсів і створює передумови для виникнення взаємоблокувань. Взаємоблокування виникає тоді, коли множина процесів перебуває в стані циклічного очікування. Кожен процес утримує певні ресурси та одночасно очікує на ресурс, який утримує інший процес з тієї ж множини. Саме усунення можливості утворення такого циклу або гарантування його розриву лежить в основі більшості протоколів уникнення взаємоблокувань. Більшість протоколів у розподілених системах спрямовані саме на усунення умови циклічного очікування або на обмеження утримання ресурсів під час очікування інших.

Послідовність основних кроків розробленого методу уникнення взаємоблокувань завдань у розподілених системах та логічні зв’язки між ними. Метод побудований як циклічний керуючий процес із вбудованими механізмами зворотного зв’язку.

Крок 1. Формування подієво-орієнтованої моделі системи.

На початковому етапі кожен вузол формує локальну модель процесів, ресурсів і подій. Визначаються типи подій: створення процесу; запит ресурсу; виділення ресурсу; звільнення ресурсу; передача повідомлення; завершення процесу. Для кожної події фіксується логічна часова мітка та ідентифікатор процесу. Сформована подієва модель створює інформаційну основу для обчислення метрик і побудови локального графа очікування.

Крок 2. Обчислення локальних і глобальних метрик.

На основі потоку подій у вузлі розраховуються такі метрики: середній час очікування ресурсу; інтенсивність надходження запитів; коефіцієнт утримання ресурсів; довжина ланцюга залежностей процесу; локальний індекс ризику блокування. Періодично вузли обмінюються агрегованими показниками для формування глобального індексу циклічної загрози. Метрики є похідними від подієвої моделі. Результати оцінювання використовуються для прийняття рішення щодо дозволу або відкладення запиту ресурсу в межах протоколу міжпроцесної взаємодії.

Крок 3. Реєстрація наміру доступу до ресурсу.

Перед фактичним захопленням ресурсу процес ініціює подію «запит наміру» і надсилає відповідне

повідомлення іншим вузлам або координатору. У повідомленні містяться дані про ідентифікатор процесу, перелік утримуваних ресурсів, перелік запитуваних ресурсів, логічна мітка часу та локальні метрики. Передача наміру активується на основі поточного рівня ризику, визначеного метриками. Отримані повідомлення формують основу для узгодження порядку доступу.

Крок 4. Узгодження глобального порядку доступу.

Використовуючи часові мітки, пріоритети та показники навантаження, формується частковий або повний порядок обслуговування запитів. У системах із координатором рішення приймає центральний вузол. У децентралізованих системах застосовується алгоритм розподіленого узгодження. Порядок визначається на основі отриманих запитів наміру. Після визначення порядку виконується перевірка на можливість утворення циклу очікування.

Крок 5. Локальне та глобальне виявлення потенційних циклів.

Кожен вузол підтримує локальний граф очікування. За потреби виконується розподілена агрегація часткових графів для формування глобального графа залежностей. Використовується алгоритм пошуку циклів. Якщо цикл прогнозується або вже існує, формується сигнал ризику. Граф будується з урахуванням узгодженого порядку доступу. Результат аналізу визначає, чи буде запит дозволено, відкладено або ініційовано процедуру розв'язання конфлікту.

Крок 6. Прийняття рішення щодо доступу до ресурсу.

Можливі такі варіанти: підтвердження доступу; відкладення запиту; примусовий відкат одного з процесів; коригування пріоритетів. Рішення базується на мінімізації глобального індексу ризику та збереженні ациклічності графа очікування. Вибір дії визначається результатом перевірки на циклічність. Після виконання рішення оновлюється подієвий журнал і метрики.

Крок 7. Оновлення стану та адаптація параметрів.

Після кожної завершеної операції або зміни стану перераховуються метрики, оновлюється граф очікування, коригуються порогові значення ризику. Система переходить до наступного циклу аналізу. Оновлення відображає наслідки прийнятого рішення. Цикл замикається так: нові події знову потрапляють у подієву модель, ініціюючи наступний цикл оцінювання.

Між кроками існують прямі функціональні зв'язки, тобто результат попереднього кроку є вхідними даними наступного, та зворотні зв'язки, тобто метрики й пороги коригуються залежно від результатів розв'язання конфліктів.

Метод представляє собою замкнену систему керування доступом до ресурсів у розподіленому середовищі, де подієва модель забезпечує інформаційну повноту, метрики, тобто кількісну оцінку ризику, алгоритми виявлення забезпечують структурний контроль циклів, а протокол міжпроцесної взаємодії забезпечує механізм узгодженої реалізації рішень. Таким чином, розроблений метод інтегрує подієво-орієнтовану модель розподілених систем, систему кількісних метрик, алгоритми розподіленого виявлення циклів та адаптивний протокол координації процесів. Це створює основу для побудови масштабованих і стійких до взаємоблокувань розподілених обчислювальних середовищ.

ЕФЕКТИВНІСТЬ ТА ЕКСПЕРИМЕНТ

Методика оцінювання ефективності розробленого методу уникнення взаємоблокувань завдань у розподілених системах на основі протоколу міжпроцесної взаємодії повинна бути спрямованою не лише на емпіричну перевірку працездатності, а й на підтвердження теоретичних положень, покладених в основу запропонованого підходу. Її призначення полягає у встановленні причинно-наслідкових зв'язків між структурою графа залежностей, правилами розподілу реакторів, особливостями міжпроцесної взаємодії та кінцевими характеристиками функціонування системи. Такий підхід дозволяє розглядати оцінювання не як суто прикладне тестування, а як інструмент верифікації коректності моделі та перевірки її узгодженості з реальними умовами функціонування гетерогенних розподілених середовищ.

Розглянемо методику оцінювання ефективності розробленого методу уникнення взаємоблокувань завдань у розподілених системах на основі протоколу міжпроцесної взаємодії. Вона спрямована на формальне підтвердження коректності запропонованого підходу, кількісне вимірювання його впливу на продуктивність системи та дослідження масштабованості в умовах гетерогенного розподіленого середовища.

Оцінювання функціональної коректності здійснимо через визначення ймовірності виникнення взаємоблокування так:

$$P_{deadlock} = \frac{N_{deadlock}}{N_{runs}}, \tag{12}$$

де $P_{deadlock}$ - ймовірність виникнення взаємоблокування; $N_{deadlock}$ - кількість експериментальних запусків, у яких було зафіксовано хоча б один цикл очікування ресурсів; N_{runs} - загальна кількість незалежних експериментальних запусків системи.

Змінна $N_{deadlock}$ є критично важливою, оскільки відображає реальні випадки порушення прогресу виконання задач. Змінна N_{runs} забезпечує статистичну репрезентативність результатів, дозволяючи оцінити стійкість методу до варіацій навантаження та конфігурацій. Якщо занчення $P_{deadlock}$ прямує до нуля при

зростанні кількості запусків, то це підтверджує коректність протоколу міжпроцесної взаємодії.

Продуктивність системи аналізуємо через середній час виконання задач:

$$T_{exec} = \frac{\sum_{i=1}^{N_{completed}} T_i}{N_{completed}}. \quad (13)$$

де T_{exec} - середній час виконання задач; T_i - фактичний час виконання і-ї задачі від моменту ініціації до завершення; $N_{completed}$ - кількість успішно завершених задач у межах експерименту.

Змінна T_i дозволяє врахувати індивідуальні особливості виконання задач різної складності, а $N_{completed}$ характеризує фактичну результативність системи. Середній час виконання відображає вплив протоколу координації на затримки доступу до ресурсів.

Пропускну здатність визначимо так:

$$T_k = \frac{N_{\text{completed}}}{T_{total}}, \quad (14)$$

де T_k - кількість завершених задач за одиницю часу; T_{total} - загальна тривалість експериментального інтервалу; $N_{\text{completed}}$ - відображає реальний рівень корисної роботи системи в кількості задач.

Змінна T_{total} задає часову нормалізацію вимірювання, а $N_{\text{completed}}$ відображає реальний рівень корисної роботи системи.

Для оцінювання ефективності використання ресурсів вводиться коефіцієнт завантаження:

$$U = \frac{T_{base}}{T_{total}}, \quad (15)$$

де U - коефіцієнт використання ресурсу; T_{base} - сумарний час, протягом якого ресурс перебував у стані активного виконання задач; T_{total} - загальний час спостереження.

Змінна T_{base} демонструє фактичну корисну зайнятість ресурсу, а співвідношення з T_{total} дозволяє оцінити ступінь простоїв або перевантаження.

Окремо визначимо додаткові витрати протоколу міжпроцесної взаємодії так:

$$R_v = \frac{M_{\text{protocol}}}{M_{total}}, \quad 16)$$

де R_v - частка службового трафіку; M_{protocol} - кількість або обсяг службових повідомлень, що генеруються протоколом координації; M_{total} - загальний обсяг переданих повідомлень у системі.

Змінна M_{protocol} відображає витрати на підтримання узгодженості та контроль циклів, тоді як M_{total} характеризує повне комунікаційне навантаження. Аналіз цього співвідношення дозволяє оцінити наявність потенційного перевищення службових витрат допустимого рівня.

Для комплексної інтегральної оцінки застосовується узагальнений показник ефективності:

$$E_{total} = w_1 \cdot (1 - P_{deadlock}) + w_2 \cdot \frac{T_k}{T_{k,max}} + w_3 U - w_4 R_v, \quad (17)$$

де E_{total} - інтегральний показник ефективності; R_v - частка службового трафіку; U - коефіцієнт використання ресурсу; w_1, w_2, w_3, w_4 вагові коефіцієнти, що відображають пріоритетність відповідних критеріїв; T_k - кількість завершених задач за одиницю часу; $T_{k,max}$ - максимальне зафіксоване значення пропускної здатності в контрольному експерименті.

Вагові коефіцієнти дозволяють адаптувати оцінювання до специфіки застосування системи, наприклад, надаючи більшої ваги гарантованій відсутності взаємоблокувань або мінімізації накладних витрат.

Таким чином, розширена методика забезпечує всебічний аналіз розробленого методу, дозволяє кількісно підтвердити його коректність, оцінити вплив на продуктивність і встановити межі масштабованості. Деталізація змінних у формулах гарантує прозорість інтерпретації результатів та відтворюваність експериментів, що є необхідною умовою наукової обґрунтованості дослідження.

На рис. 2 - рис. 5 зображені графіки результатів експериментального дослідження ефективності розробленого методу уникнення взаємоблокувань у розподілених системах на основі протоколу міжпроцесної взаємодії. Візуалізація охоплює чотири ключові групи показників: ймовірність виникнення взаємоблокувань; середній час виконання задач; частку протокольних додаткових витрат; пропускну здатність системи залежно від масштабування. Сукупний аналіз цих залежностей дозволяє комплексно оцінити функціональну коректність, продуктивність та масштабованість запропонованого підходу.

Графік на рис. 1 відображає залежність ймовірності виникнення взаємоблокування $P_{deadlock}$ від кількості незалежних запусків системи N_{runs}. Крива для запропонованого методу демонструє стрімке зниження показника на початкових етапах експерименту з подальшою стабілізацією на рівні, близькому до нуля. Така динаміка має принципове значення. Вона свідчить про те, що зі зростанням статистичної вибірки

підтверджується системна властивість протоколу, яка полягає у відсутності формування циклів очікування за умов дотримання визначених правил взаємодії між реакторами. Початкові ненульові значення пояснюються перехідними процесами, що виникають у фазі ініціалізації або при моделюванні граничних сценаріїв конкуренції ресурсів. Однак подальша стабілізація кривої на мінімальному рівні підтверджує, що метод не просто зменшує частоту взаємоблокувань, а фактично запобігає їх системному виникненню.

Для порівняння, крива класичного підходу виявлення взаємоблокувань демонструє суттєво вищий рівень $P_{deadlock}$. Це пояснюється тим, що механізм виявлення та відновлення реагує вже на сформований цикл, а не запобігає його появі. Тому в певній частці експериментів система переходить у стан блокування, який потім усувається процедурою відновлення. Ще більш виражений негативний результат спостерігається для системи без механізму уникнення, де ймовірність залишається стабільно високою та майже не залежить від кількості запусків, що вказує на структурну схильність до формування циклів очікування.

Графік на рис. 3 демонструє середній час виконання задач T_{exec} за умов низького, середнього та високого навантаження. Для розробленого методу спостерігається помірне зростання часу виконання зі збільшенням навантаження, що є природним наслідком підвищеної конкуренції за ресурси. Водночас навіть за високого навантаження значення T_{exec} залишається нижчим порівняно з альтернативними підходами. Це означає, що протокол міжпроцесної взаємодії не створює надмірних синхронізаційних затримок і не призводить до каскадного накопичення очікувань.

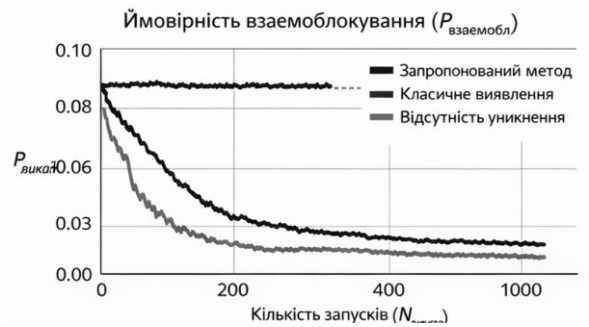

Рис. 2. Порівняльні графіки щодо ймовірності виникнення взаємоблокувань

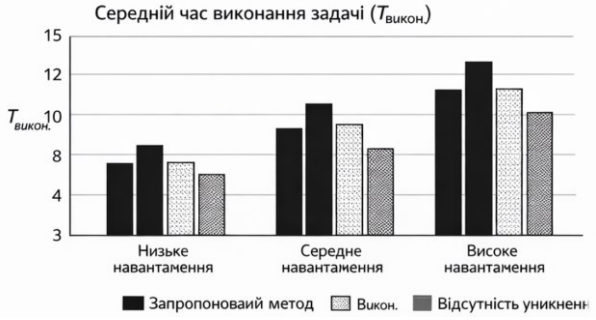

Рис. 3. Порівняльні графіки щодо середнього часу виконання завдань

У класичному методі виявлення взаємоблокувань середній час виконання зростає більш інтенсивно. Причиною цього є додаткові витрати на аналіз глобального стану та процедури відновлення після виявлення циклу. Система без механізму уникнення демонструє ще гірші результати, оскільки періодичні блокування спричиняють значні простої. Таким чином, другий графік підтверджує, що запропонований метод забезпечує не лише коректність, а й підвищену часову ефективність.

Графік на рис. 4 зображає залежність частки протокольних додаткових витрат R_v від кількості вузлів N. Для запропонованого методу спостерігається плавне зростання додаткових витрат із подальшою стабілізацією. Це свідчить про те, що механізм координації масштабуються передбачувано та не призводить до експоненційного зростання службового трафіку. Фактично протокол забезпечує локалізацію взаємодії та мінімізує глобальні синхронізації.

Натомість централізований контроль демонструє стрімке зростання додаткових витрат із масштабуванням системи. Це пов'язано з концентрацією запитів у центральному вузлі, що створює комунікаційне перевантаження. Часова стратегія має менші додаткові витрати на початкових етапах, проте зі збільшенням кількості вузлів частота повторних запитів і перезапусків зростає, що також призводить до накопичення службового трафіку. Отже, графік на рис. 4 підтверджує структурну перевагу розподіленого протоколу координації.

Графік на рис. 5 відображає залежність пропускної здатності системи від кількості вузлів. Для запропонованого методу спостерігається майже лінійне зростання продуктивності зі збільшенням числа вузлів, що свідчить про ефективну масштабованість. Кожне додавання нового вузла приводить до пропорційного зростання кількості виконаних задач за одиницю часу. Це означає, що координаційні механізми не обмежують паралелізм і не створюють вузьких місць.

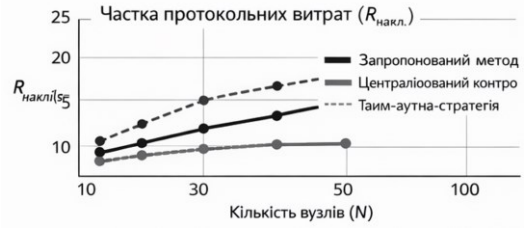

Рис. 4. Порівняльні графіки щодо частки протокольних витрат

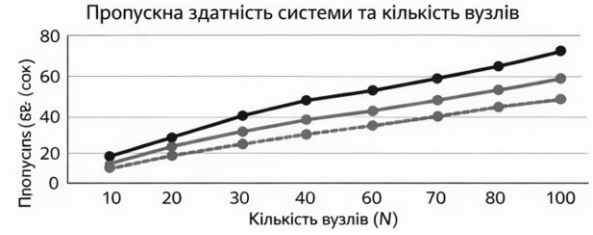

Рис. 5. Порівняльні графіки щодо пропускної здатності системи

У випадку централізованого або реактивного підходу масштабування є менш ефективним. Пропускна здатність зростає повільніше, що свідчить про накопичення координаційних витрат або періодичні втрати продуктивності через взаємоблокування.

Підсумкова табл. 1 узагальнює кількісні значення основних показників. Значення $P_{deadlock}$, близьке до нуля для запропонованого методу, підтверджує функціональну коректність. Найменший середній час виконання задач та найбільша пропускна здатність демонструють ефективність з точки зору продуктивності. Водночас частка накладних витрат залишається контрольованою і значно нижчою, ніж у централізованих підходах.

Таблиця 1

Загальні кількісні показники

Показник	Розроблений	Класичне виявлення	Відсутність уникнення
$P_{deadlock}$	0,002	0,056	0,081
Середній час виконання	5,2 с	8,4 с	11,5 с
Пропускна здатність	45,6	38,2	29,5
Частка витрат	6,7 %	15,4 %	20,3 %

У сукупності результати графічного та табличного аналізу підтверджують гіпотезу про те, що інтеграція подієво-орієнтованої моделі розподілу задач із формалізованим протоколом міжпроцесної взаємодії дозволяє забезпечити структурну відсутність циклів очікування без втрати масштабованості. Запропонований метод демонструє стабільність у широкому діапазоні навантажень, передбачуване зростання накладних витрат і лінійну динаміку продуктивності при розширенні системи. Це свідчить про його придатність до застосування в сучасних гетерогенних розподілених обчислювальних середовищах, де поєднуються вимоги високої надійності, масштабованості та ефективності використання ресурсів.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Запропонований підхід формує цілісну методологічну основу керування розподіленими обчисленнями, у якій реактор виступає ключовим елементом поєднання структурної моделі залежностей і механізмів міжпроцесної взаємодії. Інтеграція стратегії розподілу реакторів із формалізованим протоколом обміну повідомленнями забезпечує не лише теоретично обґрунтоване уникнення взаємоблокувань, а й практичну гарантованість просування виконання критичних шляхів у складних гетерогенних середовищах.

Комплексна система метрик створює кількісну основу для оцінювання стану системи, вибору алгоритмів планування, балансування та масштабування, а алгоритми виявлення циклів разом із адаптивним протоколом координації забезпечують структурний і процедурний контроль доступу до ресурсів.

Розроблений метод уникнення взаєблокувань завдань в розподілених системах представляє собою замкнену, узгоджену систему керування, що поєднує формальну модель, аналітичний інструментарій і механізми практичної реалізації, формуючи підґрунтя для побудови масштабованих, ефективних і стійких до взаємоблокувань розподілених обчислювальних середовищ.

Напрямами подальших досліджень є адаптація розробленого методу для розподілених систем з різними типами вузлів та збільшенням кількості вузлів.

References

1. Alzalab E., Abubakar U. E. H., Li Z. El-Meligy M., El-Sherbeeny A. Modeling of fault recovery and repair for automated manufacturing cells with load-sharing redundant elements using Petri nets. *Processes*. 2023, 11, 1501. **https://doi.org/10.3390/pr11051501**

2. Zhou Q., Chai B., Ran K., Guo Y., Zhou S., Wu W., Wang K., Ni Y. Research on a Multi-Dimensional Information Fusion Mechanical Wear Fault-Diagnosis Algorithm Based on Data Regeneration. *Sensors*. 2025, 25, 3745. **https://doi.org/10.3390/s25123745**

3. Chuang W., Tseng C., Tan K., Pan Y. Design of a novel transition-based deadlock recovery policy for flexible manufacturing systems. *Processes.* 2025, 13, 1610. **https://doi.org/10.3390/pr13051610**

4. Chen C., Hu H. Extended Place-Invariant Control in Automated Manufacturing Systems Using Petri Nets. *IEEE Trans. Syst. Man Cybern. Syst*. 2020, 52, 1807–1822.

5. Kaid H., Al-Ahmari A., Li Z., Davidrajuh R. Single controller-based colored petri nets for deadlock control in automated manufacturing systems. *Processes.* 2020, 8, 21. **https://doi.org/10.3390/pr8010021**

6. Feng Y., Ren S., Cao Y., Xing K., Yang Y. Deadlock control for flexible assembly systems with multiple resource requirements and separately-loaded parts. *IEEE Trans. Autom. Sci. Eng.* 2024, 22, 9275–9284. https://doi.org/10.1109/TASE.2024.3504714

7. Hu S., Li Z., Zhang Z. Design of online supervisors for enforcing diagnosability in Petri nets with unknown initial markings. *IEEE Internet Things J.* 2025, 12, 11108–11120. https://doi.org/10.1109/JIOT.2024.3515194

8. Lin X., Zhang Y., Gao Y., Chi X. Automatic construction of Petri net models for computational simulations of molecular inter-action network. *Syst. Biol. Appl*. 2024, 10, 131. https://doi.org/10.1038/s41540-024-00464-z

9. Hu H., Zhou M. A Petri net-based discrete event control of automated manufacturing systems with assembly operations. *IEEE Trans. Control Syst. Technol*. 2015, 23, 513–524. https://doi.org/10.1109/TCST.2014.2342664

10. Xing K., Wang F., Zhou M., Lei H., Luo J. Deadlock characterization and control of flexible assembly systems with Petri nets. *Automatica.* 2018, 87, 358–364. https://doi.org/10.1016/j.automatica.2017.09.001

11. Liu G., Li Z., Barkaoui K., Al-Ahmari A. Robustness of deadlock control for a class of Petri nets with unreliable resources. *Inf. Sci*. 2013, 235, 259–279. https://doi.org/10.1016/j.ins.2013.01.003

12. Cong, X. Y., Gu, C., Uzam, M., Chen, Y. F., Al-Ahmari, A. M., Wu, N. Q., Zhou, M. C., & Li, Z. W. (2018). Design of Optimal Petri Net Supervisors for Flexible Manufacturing Systems via Weighted Inhibitor Arcs. *Asian Journal of Control*, *20*(1), 511-

530. https://doi.org/10.1002/asjc.1583

13. Li, Z. W., & Zhou, M. C. (2004). Elementary Siphons of Petri Nets and Their Application to Deadlock Prevention in Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, *34*(1), 38-51. https://doi.org/10.1109/TSMCA.2003.820576

14. Huang Y., Jeng M., Xie X., Chung S. A deadlock prevention pollicy for flexible manufacturing systems using siphons. *In Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, 21–26 May 2001; pp. 541–546.

15. Piroddi L., Cordone R., Fumagalli I. Selective siphon control for deadlock preveention in Petri Nets. *IEEE Trans. Syst. Man Cybern. A. Syst. Hum*. 2008, 38, 1337–1348.

16. Bedratyuk L., Savenko O., *MATCH Commun. Math. Comput. Chem.* 2018, 79, 407–414. URL: https://match.pmf.kg.ac.rs/electronic_versions/Match79/n2/match79n2_407-414.pdf

17. Ezpeleta J., Colom J., Martinez J. A petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Trans. Robot. Autom*. 1995, 11, 173–184. https://doi.org/10.1109/70.370500

18. Liu G., Li Z., Zhong C. New controllability condition for siphons in a class of generalised Petri nets. *IET Control Theory Appl*. 2010, 4, 854–864. https://doi.org/10.1049/iet-cta.2009.0264

19. Uzam M. An Optimal Deadlock Prevention Policy for Flexible Manufacturing Systems Using Petri Net Models with Resources and the Theory of Regions. *Int. J. Adv. Manuf. Technol*. 2002, 19, 192–208. https://doi.org/10.1007/s001700200014

20. Liao H., Wang Y., Stanley J., Lafortune S., Reveliotis S., Kelly T., Mahlke S. Eliminating concurrency bugs in multithreaded software: A new approach based on discrete-event control. *IEEE Trans. Control Syst. Technol*. 2013, 21, 2067–2082. https://doi.org/10.1109/TCST.2012.2226034

21. Chen Y., Li W., Khalgui M., Mosbahi O. Design of a maximally permissive liveness enforcing Petri net supervisor for flexible manufacturing systems. *IEEE Trans. Autom. Sci. Eng*. 2011, 8, 374–393. https://doi.org/10.1109/TASE.2010.2060332