МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ

ПрАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав. кафедри _____

д.е.н., доц. С.І. Левицький

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

АНАЛІЗ МЕТОДІВ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

Виконав

ст. гр. КІ–112м _____ Д. В. Антоненко

Науковий керівник

професор _____ Н. Р. Полуектова

Запоріжжя

2024 р.

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»
Кафедра інформаційних технологій

ЗАТВЕРДЖУЮ
Зав. кафедри
д.е.н., доц. С.І. Левицький
_____
20.10.2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ
студенту гр. КІ–112м, спеціальності 123 «Комп'ютерна інженерія»
Антоненку Дмитру Вікторовичу

1.Тема: Аналіз методів балансування навантаження в комп'ютерних системах затверджена наказом по інституту № 02-25 від 05.12.2023 р.

2. Термін здачі студентом закінченої роботи: 12.01.2024 р.

3. Перелік питань що підлягають до розгляду:

1. Провести огляд предметної області, ознайомитися з літературою та інтернет-джерелами, що присвячені тематиці роботи;

2. Визначити поняття та основні методи балансування навантаження в комп'ютерних мережах;

3. Визначити основні проблеми існуючих методів балансування навантаження;

4. Запропонувати модель, яка поліпшує показники роботи системи за рахунок балансування навантаження;

5. Провести експериментальне дослідження продуктивності пропонованої моделі;

6. Проаналізувати результати експерименту і порівняти продуктивність різних рішень, у різних середовищах та умовах застосування;

7. Зробити висновки;

8. Оформити звіт за результатами роботи.

4. Календарний графік магістерських дипломних робіт

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Корегування теми кваліфікаційної магістерської роботи, збір практичного матеріалу за темою кваліфікаційної магістерської роботи	04.09.23-17.10.23		
2	І атестація І розділ кваліфікаційної магістерської роботи	23.10.23-28.10.23		
3	ІІ атестація ІІ розділ кваліфікаційної магістерської роботи	20.11.23-25.11.23		
4	ІІІ атестація ІІІ розділ кваліфікаційної магістерської роботи, висновки та рекомендації, додатки, реферат, перевірка програмою «Антиплагіат»	18.12.23-23.12.23		
5	Доопрацювання кваліфікаційної магістерської роботи, підготовка презентації, отримання відгуку керівника і рецензії	25.12.23-06.01.24		
6	Попередній захист кваліфікаційної магістерської роботи	09.01.24-10.01.24		
7	Подача кваліфікаційної магістерської роботи на кафедру	**за 3 дні до захисту**		
8	Захист кваліфікаційної магістерської роботи	17.01.24-18.01.24		

Дата видачі завдання: 20.10.2023 р.

Керівник магістерської дипломної роботи _____ Н. Р. Полуектова

Завдання отримав до виконання _____ Д. В. Антоненко

РЕФЕРАТ

Кваліфікаційна магістерська робота містить 77 сторінок, 4 таблиці, 22 рисунки, 44 бібліографічні посилання.

Об’єктом дослідження є розподілені високонавантажені комп’ютерні мережі. Предметом дослідження є методи балансування навантаження в розподілених комп’ютерних мережах.

Метою роботи є аналіз існуючих методів балансування навантаження та розробка ефективної методики балансування за результатами аналізу проблем існуючих підходів.

В першому розділі здійснено детальний огляд предметної області. Досліджено поняття розподілених комп’ютерних систем та основні проблеми їх використання, а також поняття балансування навантаження в комп’ютерних мережах. Оглянуто основні стратегії балансування навантаження в розподілених комп’ютерних мережах.

У другому розділі проведено аналіз статичного та динамічного підходів до балансування навантаження в комп’ютерних мережах. Досліджено особливості статичного та динамічного балансування навантаження. Здійснено порівняльний аналіз підходів цих підходів за такими напрямками: збільшення веб-трафіку, вибір ресурсів та розподіл завдань, вимірювання навантаження, оптимізація витрат, відмовостійкість, питання сумісності та проблеми QoS.

В третьому розділі запропонована модель балансування навантаження в комп’ютерних мережах. Визначені принципи програмно-конфігурованих мереж (SDN) та досліджена архітектура системи керування трафіку (TE). Описаний евристичний алгоритм балансування та модифікований алгоритм на основі ACS (EDAFT) та здійснена оцінка його працездатності.

SDN, TE, ACS, EDAFT, БАЛАНСУВАННЯ НАВАНТАЖЕННЯ,
КОМП’ЮТЕРНА МЕРЕЖА, ЕФЕКТИВНІСТЬ, ПРОДУКТИВНІСТЬ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Слово / словосполучення	Скорочення
алгоритм балансування навантаження	АБН
статичні алгоритми балансування навантаження	СБН
алгоритми динамічного балансування навантаження	ДБН
елемента обробки	ЕО
балансування навантаження (англ. Load Balancing)	LB
програмно-конфігурована мережа (англ. Software-defined network)	SDN
конструювання трафіка (англ. Traffic Engineering)	TE
центр обробки даних (англ. Data Center Network)	DCN
ієрархічний алгоритм балансування навантаження (англ. Hierarchical Load Balancing Algorithm)	HLBA
оптимізований алгоритм колонії мурашок (англ. Ant Colony Optimization)	ACO
удосконалена динамічна відмовостійкість на основі ACS (англ. Enhanced Dynamic ACS-based Fault Tolerance)	EDAFT
оптимізований алгоритм колонії мурашок з відмовостійкістю (англ. ACO with Fault Tolerance algorithm)	ACOwFT

ВСТУП

Широко використовувані технології мають тенденцію до зростання кількості користувачів. Масове зростання неконтрольованого мережевого трафіку в непередбачуваних об'ємах, може виявити вузькі місця в деяких каналах, а інші – недостатньо навантажити, що призведе до нерівномірного розподілу навантаження та невідповідність умовам якості QoS. Таким чином, є потреба у створенні механізму ефективної обробки та передачі трафіку без втрат з підвищенням швидкості передачі за рахунок рівномірного розподілу навантаження, тобто балансування. Відтак, аналіз методів балансування навантаження в комп'ютерних системах постає дуже актуальним завданням.

Сучасні програми повинні обробляти запити мільйонів користувачів одночасно, а також швидко і надійно повертати кожному користувачеві правильний текст, відео, зображення та інші дані. Для обробки таких значущих обсягів трафіку в більшості програм використовується безліч серверів ресурсів з дублюванням даних між ними. Балансування навантаження направляє і контролює інтернет-трафік між серверами додатків і їх відвідувачами або клієнтами. У результаті підвищується доступність, масштабність, безпека та продуктивність додатків.

Тож мета цього дослідження: розглянути основні методи балансування навантаження в розподіленій комп'ютерній мережі та визначити рекомендації до застосуванні оптимальних методів балансування навантаження для конкретних умов.

Об'єктом дослідження є розподілені високонавантажені комп'ютерні мережі.

Предметом дослідження є методи балансування навантаження в розподілених комп'ютерних мережах.

Метою роботи є аналіз існуючих методів балансування навантаження та розробка ефективної методики балансування за результатами аналізу проблем існуючих підходів.

Для досягнення мети роботи було поставлено такі завдання:

- визначити поняття та основні методи балансування навантаження в комп’ютерних мережах;

- визначити основні проблеми існуючих методів балансування навантаження;

- дослідити принципи програмно-конфігурованих мереж;

- дослідити методи балансування навантаження, які базуються на алгоритмі ACS;

- запропонувати та дослідити ефективність модифікації алгоритму ACS для покращення показників якості обробки завдань.

Для виконання поставлених завдань було застосовано такі методи дослідження: експеримент, вимірювання та порівняння.

Отримані результати можуть бути при розробці архітектури високонавантажених розподілених комп’ютерних мереж.

Апробація. Основні положення досліджень роботи доповідалися на щорічній загальноінститутській науковій конференції в Запорізькому інституті економіки і інформаційних технологій у рамках секції «Інформаційні технології» з публікацією у збірці тез конференції.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття розподілених комп’ютерних систем та основні проблеми їх використання

Розподілена система – це набір комп’ютерних програм, що використовують обчислювальні ресурси кількох окремих обчислювальних вузлів для досягнення загальної мети [1]. Її також називають розподіленими обчисленнями чи розподіленою базою даних. Розподілена система ґрунтується на окремих вузлах, які обмінюються даними та виконують синхронізацію у загальній мережі. Зазвичай вузли є окремими фізичними апаратними пристроями, але це можуть бути і окремі програмні процеси або інші рекурсивні інкапсульовані системи. Розподілені системи спрямовані усунення вузьких місць чи єдиних точок відмови у системі .

Розподілені системи – це складна концепція, заснована на програмних архітектурах (особливо мікросервісах). Фізично розподілена система являє собою сукупність фізичних машин, які обмінюються даними мережевими каналами. Іншими словами, розподілена система складається з програмних процесів, які взаємодіють через механізми IPC та розміщуються на комп’ютерах.

Також можна сказати, що розподілена система – це набір слабко пов’язаних компонентів, які можна розгортати і масштабувати незалежно один від одного, які називаються сервісами [1].

Розподілена система призначена виконати різні, унікальні та жорстокі завдання: комунікація; координація; масштабованість; стійкість; обслуговування.

Вузли, машини або служби повинні взаємодіяти між собою по мережі.

Концептуальна основа (фактично модель OSI) дозволяє краще зрозуміти складні взаємодії, що відбуваються [2]. Мережеві протоколи організовані як

стеки, де кожен рівень будується на абстракції, що забезпечується нижчим рівнем, а нижні рівні ближче до устаткування.

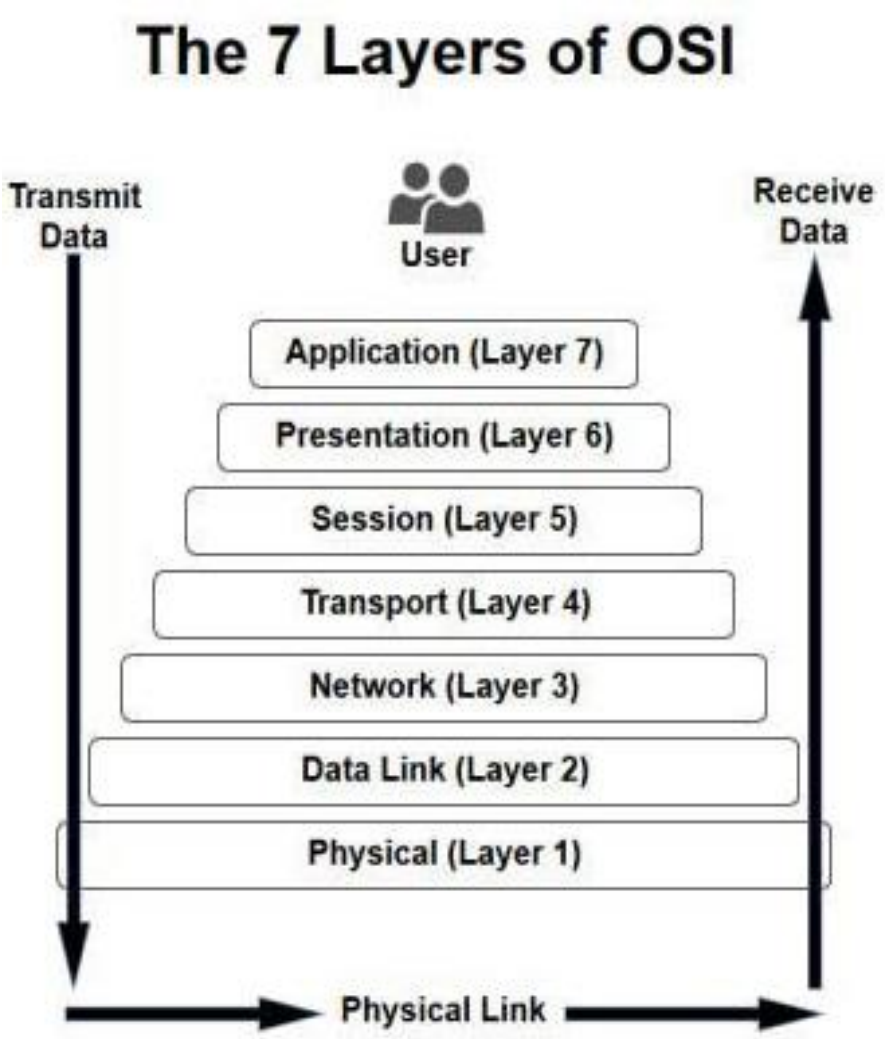


Рис. 1.1 – Модель OSI

Мережеві протоколи організовані як стек, де кожен рівень будується на абстракції, що забезпечується нижчим рівнем, а нижні рівні ближче до устаткування.

Якщо потрібно побудувати розподілену систему, необхідно створити надійну та безпечну модель зв’язку. Крім того, необхідно враховувати свій стиль спілкування між клієнтами та послугами. Необхідно вирішити свою стратегію, розглядаючи свій власний бізнес. Можна використовувати пряму чи непряму (або обидві) комунікаційну модель(і).

Необхідно ретельно обирати технології IPC. Послуги надають свої операції своїм споживачам через набір інтерфейсів, що реалізуються його бізнес-логікою. Клієнти не можуть отримати доступ до цих операцій безпосередньо. Необхідно перетворити повідомлення, отримані від механізмів IPC, на інтерфейсні виклики.

Необхідно точно визначити, що може і не може відбуватися у розподілених системах. З іншого боку, треба побудувати модель системи, яка кодує очікування щодо поведінки процесів, каналів зв'язку та часу. Є тільки дев'ять моделей для побудови розподіленої системи. Необхідно знати кожну модель, щоб вирішити свою власну.

Розглянемо моделі каналів зв'язку [1-2]:

- модель зв'язку із справедливими втратами: канали доставляють будь-яке повідомлення, відправлене з ненульовою ймовірністю (без мережевих розділів).
- модель надійного зв'язку: канали, які доставляють будь-яке повідомлення, відправлене рівно один раз
- модель надійного зв'язку з автентифікацією: те саме, що і модель надійного зв'язку, але додатково передбачає, що одержувач може аутентифікувати відправника.

Порахуємо види процесів моделей [1-2]:

- модель довільної помилки: також відома як візантійська модель, передбачає, що процес може довільно відхилятися від свого алгоритму, що призводить до збою або несподіваної поведінки через помилки або зловмисну активність.
- модель відновлення після збою: передбачає, що процес не відхиляється від свого алгоритму, але може будь-якої миті зависнути і перезапуститися, втративши свій стан у пам'яті.
- модель аварійної зупинки: передбачається, що процес не відхиляється від свого алгоритму, але у разі збою він ніколи не повертається до ладу.

- синхронна модель (не є реальною): передбачається, що надсилання повідомлення або виконання операції ніколи не займає певний час.

- асинхронна модель (має багато проблем): передбачається, що надсилання повідомлення або виконання операції над процесом може тривати необмежену кількість часу.

- частково синхронна модель (практична): передбачається, що більшість часу система веде себе синхронно, але іноді може повернутися до асинхронної моделі.

Масштабована служба або програма можуть збільшувати свою продуктивність у міру збільшення навантаження. Найпростіший спосіб зробити це – масштабувати і запускати службу або додаток на більш дорогому устаткуванні, але це тільки призводить до того, що програма в кінцевому підсумку досягає продуктивності. стеля.

Альтернативою масштабування є масштабування шляхом розподілу навантаження на кілька вузлів. Ви можете використовувати масштабування лише трьома способами: функціональна декомпозиція, поділ та дублювання.

Продуктивність розподіленої системи показує, наскільки ефективно вона справляється з навантаженням, і зазвичай вимірюється пропускною здатністю та часом відгуку. Пропускна спроможність – це кількість операцій, що обробляються за секунду, а час відповіді – це загальний час, витрачений на кожен запит [1].

Розподілена система стійка, якщо вона може продовжувати виконувати свою роботу навіть у разі збоїв. Будь-яка невдача, яка може статися, рано чи пізно станеться. Кожен компонент системи має можливість відмови. Незалежно від того, наскільки мала ця ймовірність, що більше компонентів і чим більше операцій виконує система, то вище стає абсолютна кількість відмов.

Якщо система не стійка до збоїв, які тільки збільшуються в міру масштабування програми для обробки більшого навантаження, її доступність неминуче зменшиться. Ім’я невдачі не має значення.

Збої обладнання, збої програмного забезпечення, витоку пам’яті або щось інше. Необхідно гарантувати щонайменше дві дев’ятки.

Найбільш поширені причини збоїв:

- єдина точка відмови;
- ненадійна мережа;
- повільні процеси;
- несподіване навантаження;
- каскадні збої.

Ліквідація проблем полягає в наступному.

- відмовостійкість низхідного потоку: тайм-аут(и), механізми повторних спроб, автоматичний вимикач;
- відмовостійкість висхідного потоку: скидання навантаження, вирівнювання навантаження, обмеження швидкості, перебирання, кінцеві точки працездатності, сторожові пси [1].

Насправді технічне обслуговування є складнішою проблемою, ніж інші.

Спочатку необхідно перевірити тестову піраміду. Чим більше часу витрачається на виявлення помилки, тим дорожче обходиться її виправлення. Тестування полягає у виявленні помилок якомога раніше, що дозволяє розробникам змінювати реалізацію з упевненістю, що існуюча функціональність не зламається, збільшує швидкість рефакторингу, доставки нових функцій та інших змін.

Іншим ключовим моментом є безперервний моніторинг, він здебільшого використовується для виявлення збоїв, які впливають на користувачів у виробничому середовищі, та ініціювання повідомлень (попереджень), що надсилаються операторам-людям, відповідальним за їх усунення. Іншим важливим варіантом використання моніторингу є надання загального огляду стану системи за допомогою інформаційних панелей.

Також треба врахувати:

- метрики – числове подання інформації, виміряної за інтервал часу та поданої у вигляді часового ряду, наприклад кількість запитів, оброблених

службою. Метрики з високою кардинальністю спрощують нарізування та нарізування даних, а також усувають витрати на інструменти, пов’язані з ручним створенням метрики для кожної комбінації міток.

- індикатори рівня обслуговування: (SLI) – це метрика, яка вимірює один із аспектів рівня обслуговування, який надає служба своїм користувачам, наприклад, час відгуку, частота помилок або tps. SLI зазвичай агрегуються за ковзним тимчасовим вікном і видаються зведеною статистикою.
- об’єкти рівня служби: (SLO) визначає діапазон допустимих значень для SLI, у межах якого вважається, що служба перебуває у працездатному стані. (також ви можете встановити очікування)
- оповіщення
- панелі приладів
- спостерігається: розподілена система ніколи не буде справною (100%) у будь-який момент часу.

Отже, потрібні журнали та трасування, щоб зробити гіпотезу та спробувати її перевірити.

1.2 Поняття балансування навантаження в комп’ютерних мережах

Використання балансування навантаження (Load Balancing) визначається як оптимізація виконання розподілених (паралельних) обчислень шляхом рівномірного розподілу завдань між обчислювальними вузлами (процесорами багатопроцесорних ЕОМ або комп’ютерами у мережі) [3].

Розвиток та впровадження у повсякденне життя населення інформаційних систем зумовило різке збільшення кількості запитів на обробку, збільшення навантаження на обробне обладнання (сервери).

Використання високонавантажених систем, послугами яких користується велика кількість користувачів, що вимагає застосування в якості апаратної платформи серверних груп або кластерів. Кластер складається з кількох комп’ютерів, об’єднаних високошвидкісне з’єднання. Для

користувачів кластер виглядає як один комп’ютер, а всередині він є різновидом мережі, яка може бути розподіленою або локальної.

Важлива ланка кластера зосереджена на пристрої або програмному забезпеченні, що розподіляє навантаження (потік запитів) між комп’ютерами кластера. Цей пристрій (програмне забезпечення) називається балансувальником навантаження.

Проблемою, пов’язаною з балансуванням, є питання про те, як розподіляти навантаження найефективніше. Для цього необхідно формалізувати методику оцінки якості роботи балансувальника, яка залежить від параметрів системи та параметрів вхідного потоку запитів.

Оскільки навантаження на інформаційні системи постійно зростатиме, завдання балансування будуть набувати все більш важливого значення для підвищення ефективності інформаційних систем.

У термінології комп’ютерних мереж, балансування навантаження (або вирівнювання навантаження) представляє собою стратегію розподілу завдань між численними мережевими пристроями, такими як сервери, з метою оптимізації використання ресурсів, скорочення часу обслуговування запитів, горизонтального масштабування кластера (динамічне додавання або вилучення пристроїв), а також забезпечення відмовостійкості (резервування) [3].

У сфері комп’ютерів балансування навантаження означає розподіл навантаження між різними обчислювальними ресурсами, такими як комп’ютери, кластери, мережі, центральні процесори або диски.

Основною метою балансування навантаження є оптимізація використання ресурсів, максимізація пропускної здатності, скорочення часу відгуку та уникнення перевантаження будь-якого окремого ресурсу. Використання кількох компонентів балансування навантаження, замість одного, може підвищити надійність і доступність за рахунок резервування.

Зазвичай балансування навантаження передбачає використання спеціального програмного забезпечення або апаратних засобів, таких як

багаторівневий комутатор або система доменних імен, як серверний процес.Балансування навантаження відрізняється від фізичного з’єднання тим, що балансування навантаження ділить трафік між мережевими інтерфейсами на мережевий сокет (модельOSI рівень 4) основі, у той час як з’єднання каналу перед бачає поділ трафіку між фізичними інтерфейсами на нижчому рівні, або в пакет (модель OSI рівень 3) або каналом зв’язку (модель OSI рівень 2).

Приклади пристроїв, до яких застосовується балансування: серверні кластери; проксі-сервери; міжмережові екрани; комутатори; сервери інспектування вмісту; сервери DNS; мережні адаптери.

Балансування навантаження може бути використане для розширення можливостей ферми серверів, що складається з більш ніж одного сервера. Вона також може дозволити продовжувати роботу навіть за умов, коли кілька виконавчих пристроїв (серверів) вийшли з ладу. Завдяки цьому зростає стійкість до відмов, і з’являється можливість динамічно регулювати використовувані обчислювальні ресурси за рахунок додавання/видалення виконавчих пристроїв в кластері.

Балансування навантаження в розподілених комп’ютерних системах - це процес перерозподілу робочого навантаження між процесорами в системі підвищення продуктивності системи. Однак спроба досягти цього - непросте завдання. У недавніх дослідженнях та літературі було запропоновано різні підходи для досягнення цієї мети.

Розглянемо різні підходи до вирішення проблеми [4-10].

Існує безліч методів та алгоритмів, які можна використовувати для інтелектуальне балансування навантаження запитів клієнтів на пули серверів.

Вибраний метод буде залежати від типу послуги або програми, що обслуговується, а також від стану мережі та серверів на момент запиту.

Методи, описані нижче, будуть використовуватися в комбінації, щоб визначити найкращий сервер для обслуговування нових запитів.

Поточний рівень запитів до балансувальників навантаження часто визначає метод, який використовується. Коли навантаження невелике, тоді буде достатньо одного з простих методів балансування навантаження. У періоди високого навантаження використовуються складніші методи забезпечення рівномірного розподілу запитів.

Основним завданням балансувальника є визначення того, який із серверів з бекенд-пулу може найефективніше обробити вхідний пакет даних. Для цього у балансувальника (Load Balancer, LB) є кілька алгоритмів, наявність та можливість застосування яких залежить від типу LB та його налаштувань.

Групи балансування навантаження використовують алгоритми для прийняття рішень про розподіл навантаження. Рішення визначає, якому віддаленому серверу надіслати нове з’єднання.

Групи балансування навантаження підтримують виважені та незважені алгоритми.

Виважений алгоритм використовує вагу (або перевагу), щоб допомогти визначити, який сервер отримує наступний запит. Сервер з більшою вагою отримує більше трафіку, ніж сервер із меншою вагою. Відсоток трафіку на кожен сервер приблизно дорівнює його вазі, поділеному на сукупну вагу всіх серверів групи.

Незважений алгоритм передбачає, що ємність всіх серверів групи еквівалентна. Хоча незважені алгоритми зазвичай швидше, ніж зважені алгоритми, деякі незважені алгоритми, такі як алгоритм хешування, можуть надсилати більше трафіку деякі сервери. Якщо групі є сервери з різними потужностями, обробка неспроможна оптимізувати потужності всіх серверів.

Вибираючи конкретний алгоритм, потрібно виходити, по-перше, зі специфіки конкретного проекту, а по-друге – з цілей, яких ми плануємо досягти.

Серед поставлених цілей, які досягаються за допомогою балансування, можна визначити наступні:

- Справедливість: гарантує, що системні ресурси виділяються для обробки кожного запиту, і уникнення ситуацій, коли один запит обробляється, а інші чекають в черзі;

- Ефективність: досягається, коли всі сервери, які обробляють запити, максимально завантажені. Бажано уникати ситуацій, коли один із серверів залишається незайнятим в очікуванні обробки запитів (хоча це може бути складно досягти на практиці);

- Скорочення часу виконання запиту: забезпечує мінімальний час між початком обробки запиту (або його постановкою в чергу на обробку) та його завершенням;

- Скорочення часу відгуку: мінімізує час відповіді на запит користувача.

Додатково до цього, бажано, щоб алгоритм балансування володів такими характеристиками:

- Передбачуваність: забезпечує чітке розуміння умов і навантажень, при яких алгоритм буде ефективним для вирішення завдань;

- Рівномірне завантаження ресурсів системи;

- Масштабованість: здатність алгоритму зберігати ефективність при збільшенні навантаження.

1.3 Основні стратегії балансування навантаження в розподілених комп’ютерних мережах

Розглянемо типи виконання балансування, диференційовані на момент виконання і, як наслідок, що оперують різним обсягом інформації про систему. В даному випадку розрізняють статичне та динамічне балансування.

На рис. 1.2 наведено основні типи систем балансування навантаження.

Статичне балансування провадиться перед початком роботи розподіленої системи. Такий підхід часто ґрунтується на знаннях про попередні відпрацювання цільової системи або подібних систем і дозволяє розподілити відповідальність за обробку запитів між вузлами на основі цих

знань. Статичне балансування може продемонструвати переваги в системах із передбачуваним навантаженням.

Рис. 1.2 – Основні типи систем балансування навантаження

У разі систем, що відрізняються стохастичною невизначеністю виконання завдань, даний підхід не дозволяє ефективно розподілити обробку з кількох причин: динамічна зміна середовища виконання, навантаження на вузлах у подібних системах не може бути передбачене однозначно на початку роботи [5].

На відміну від статичного балансування, що виконується перед початком роботи системи, динамічний підхід дозволяє орієнтуватися на стан системи в даний момент та враховувати дані метрики під час здійснення розподілу навантаження. Так як в даному випадку нас цікавить гетерогенна розподілена система, навантаження в окремий момент часу якої схильна до стохастичної невизначеності, динамічний підхід дозволяє ефективно проводити балансування на вузлах такої системи. На етапі виконання існує можливість зняття показників системи, що дозволяють зробити рішення, таких як поточне навантаження на вузлах, доступність вузлів, пропускну здатність каналів передачі і т.д.

Деякі дослідження динамічного балансування навантаження показують, що алгоритми, що приймають рішення про таке балансування з використанням

великої кількості докладно зібраних даних про систему, не дають якогось суттєвого приросту продуктивності системи щодо алгоритмів, що належать при винесенні рішення на мінімальний набір інформації.

Алгоритми можуть відрізнятись за типом ініціаторів. Існують підходи, під час використання яких ініціатива передачі завдання надходить від власника завдання. Даний підхід хороший тим, що можлива мінімізація інформаційного обміну між вузлами системи у випадках, коли вузол-власник вирішує виконувати завдання локально, а також балансування ініціюється лише тоді, коли є завдання на розподіл. Як альтернатива ініціатором балансування можуть виступати вузли-виконавці, створюючи заявки, що позначають готовність вузла прийняти додаткове навантаження.

При використанні даної стратегії балансування навантаження перебуває у стані часткового виконання якщо у системі немає поточних завдань, оскільки заявки ініціюються зі звільненням вузлів, на відміну попереднього підходу. Існуючі дослідження показують велику ефективність стратегії ініціатора-виконавця в системах, що зазнають сильного навантаження, на відміну від стратегії ініціатора-власника, яка досить добре працює в помірно навантажених системах.

Балансування навантаження може здійснюватися тільки на прибулих завданнях, або в балансуванні може брати участь безліч активних завдань, що ускладнює сам процес балансування, але згідно з деякими дослідженнями такий підхід дає кращий розподіл [4-10].

За типом пристрою системи балансування навантаження може бути як централізованими, і розподіленими. Централізована схема передбачає розташування керуючого компонента єдиному вузлі. Інакше схема, в якій кожен вузол бере участь у процесі балансування навантаження, збирає дані про поточне навантаження, приймає рішення щодо обробки запиту, а також виконує міграцію даних, має розподілену структуру.

Централізована схема балансування навантаження зазвичай тягне за собою мінімальні витрати на комунікацію вузлів, оскільки вся інформація про

стан системи концентрується в єдиному місці. Мінусом цього підходу може бути низька стійкість до відмов, оскільки при виході з ладу центрального вузла, відповідального за балансування навантаження, така схема перестає працювати. Також у цьому випадку централізація розподілу завдань чи підключень як така може викликати перевантаження центральної ланки.

Розподілена схема балансування позбавлена недоліків централізованої і своєю чергою може виконуватися з допомогою кооперативного і кооперативного підходів. Якщо говорити про кооперативний підхід, при якому рішення про передачу завдань виконується на основі колективного рішення, у цьому випадку може виникнути високе навантаження системи зв’язку, пов’язане з витратами на комунікацію між вузлами. Вирішенням цієї проблеми бачиться мінімізація надмірної взаємодії вузлів у рамках розподіленого алгоритму балансування навантаження.

У межах некооперативних розподілених стратегій здійснюються індивідуальні рішення вузлів про розподіл навантаження, з урахуванням інформації про загальний стан системи. Некооперативний підхід дозволяє знизити витрати на комунікацію, а також скористатися основними перевагами розподіленого підходу, такими як відмовостійкість та раціональне використання ресурсів.

В якості альтернативи може використовуватися змішана, частково розподілена схема балансування навантаження, де безліч всіх вузлів поділено на підмножини (кластери), серед яких у свою чергу панує централізована політика. Ця модифікація дозволяє скористатися перевагами як розподіленої, і нерозподіленої схеми.

Найпростішою стратегією балансування навантаження є випадковий вибір. Використовуючи даний підхід у найпростішому випадку немає потреби збирати дані про навантаження системи та стан компонентів. Визначення вузла, який отримає завдання на обробку запиту, провадиться на основі випадкового вибору.

Перевагою даного алгоритму є простота його реалізації та низьке навантаження на систему балансування, але цінність даних переваг є сумнівною з урахуванням недоліку у вигляді розподілу навантаження без урахування інформації про завантаженість вузлів, що може бути критичним і призвести до серйозних наслідків, у вигляді часткової або повної відмови системи. Випадкова стратегія дає непогані результати щодо систем, що не використовують балансування, і часто використовується як відправна точка для порівняння інших алгоритмів балансування навантаження. На рис. 1.3 зображено роботу стратегії випадкового вибору з використанням централізованої схеми.

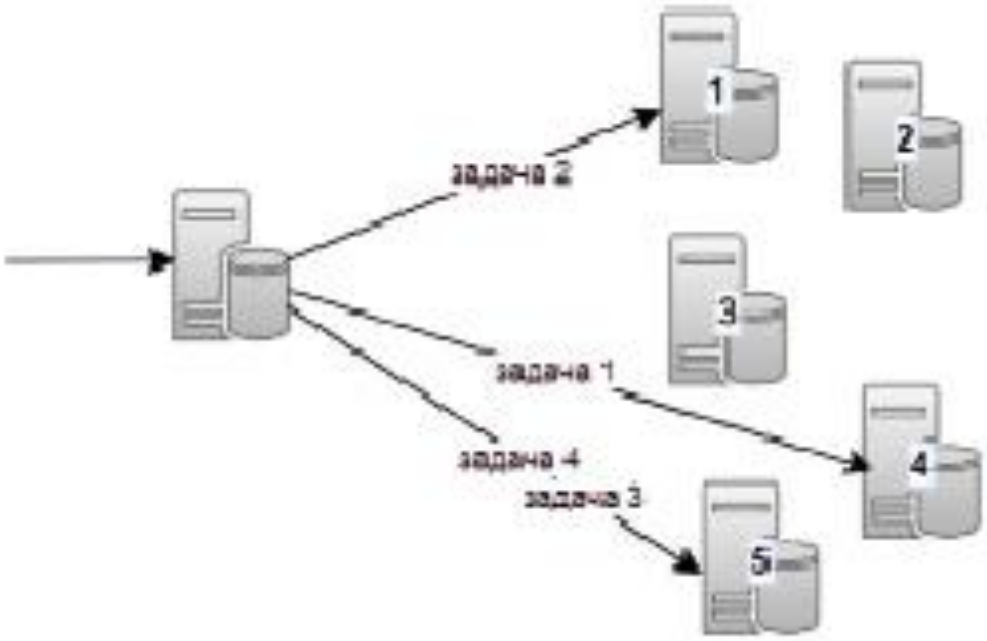


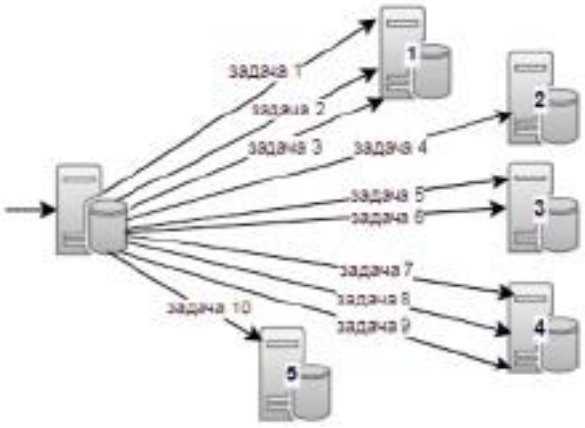
Рис. 1.3 – Централізована схема стратегії випадкового вибору

Циклічний алгоритм має на увазі розподіл завдань по обробникам з використанням циклічного або кругового підходу [11]. Циклічна стратегія зазвичай застосовується, коли завдання за своєю складністю та структурою є типовими. Такий алгоритм досить простий у виконанні та передбачає наївне балансування навантаження, розраховуючи на припущення про те, що коли черга на обробку повторно звернеться до вузла, він встигне звільнити

достатньо ресурсів для обробки наступного запиту. В даному випадку використання кругового алгоритму в гетерогенній розподіленій системі є невиправданим, оскільки облік навантаження системи та складності завдань не здійснюється, внаслідок чого різко зростає ризик відмови системи.

Зважений циклічний алгоритм є вдосконаленою версією циклічної стратегії та орієнтований на роботу у гетерогенному середовищі [12]. Суть алгоритму в тому, що кожен вузол маркується певною вагою, що означає продуктивність оброблювача. Залежно від величини даної ваги вузол отримує більше завдань. На рис. 1.4 зображено роботу зваженого циклічного алгоритму з використанням централізованої схеми.

При надходженні нового запиту балансувальник навантаження використовує стратегію найменшого навантаження та визначає кількість запитів, що виконуються на вузлах і здійснює з’єднання з найменш навантаженим з них. Цей алгоритм успішно підходить для систем, у яких навантаження більшою мірою визначається кількістю виконуваних завдань.

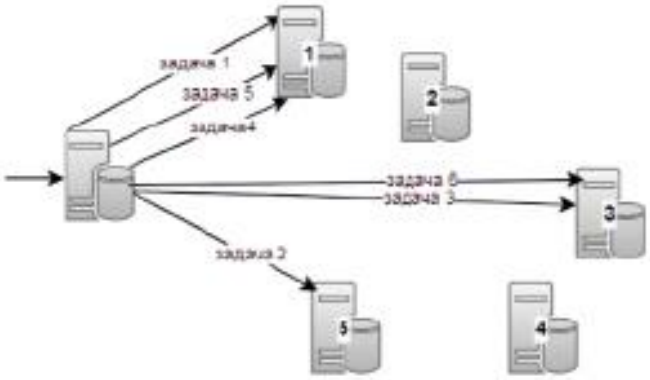


Начальное состояние системы

Узел	Мощность
1	100
2	20
3	70
4	100
5	20

Рис. 1.4 – Централізована схема зваженого циклічного алгоритму

Як і у випадку з циклічною стратегією балансування навантаження, цей алгоритм має розширену версію, яка використовує інформацію про продуктивність сервера. В даному випадку вдосконалена версія оперує не тільки поточним навантаженням, але також враховує ваги вузлів при здійсненні балансування. На рис. 1.5 зображено роботу виваженої стратегії.

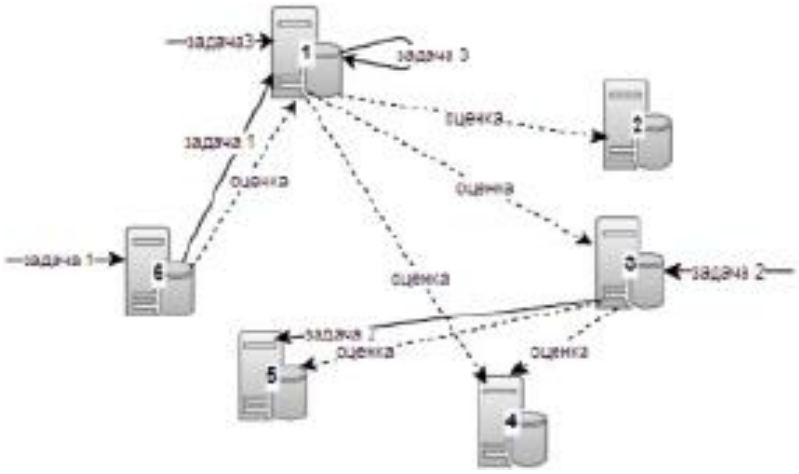


Начальное состояние системы

Узел	Мощность	Нагрузка
1	100	0
2	20	90
3	70	40
4	100	100
5	20	40

Рис. 1.5 – Централізована схема виваженої стратегії найменшого навантаження

На рис. 1.6 продемонстрована розподілена схема жадібної стратегії зондування, в якій максимальним числом оцінок встановлено 3, а як поріг використовується рівень навантаження власника.



Начальное состояние системы

Узел	Нагрузка
1	10
2	40
3	80
4	100
5	30
6	60

Рис. 1.6 – Розподілена схема жадібної стратегії зондування

Існують алгоритми балансування, що базуються на концепції переговорів. У стратегії переговорів навантажений вузол ініціює запит-заявку, де описує своє навантаження, і навіть завдання, які він хотів би передати. Віддалений вузол перевіряє даний запит і порівнює навантаження вузла ініціатора зі своїм, якщо вона менше, цей віддалений вузол відправляє заявку на виконання завдання ініціатору. Інакше вузол ігнорує повідомлення. У

результаті вузол-ініціатор отримує заявки прийняття завдань і вибирає їх най незавантажені вузли.

Проблемою цього підходу може бути ситуація, в якій вузли з меншим завантаженням можуть стати перевантаженими внаслідок перемоги у багатьох одночасних переговорах. Позбутися цієї проблеми можна, використовуючи обмеження кількості заявок. Існує інша версія стратегії переговорів. Вузли розділені на 3 групи: слабо навантажені, помірно навантажені та сильно навантажені. Вузли спостерігають за своїм навантаженням і періодично змінюють свої статки, сповіщаючи про це інші вузли. Таблиця стану вузлів зберігається у кожного вузла.

Слабо навантажені вузли надсилають заявки на прийняття завдань усім завантаженим вузлам. Вузли, що відчувають сильну завантаженість, відправляють відповідь, у якій міститься інформація про завдання, якими цей вузол міг би поділитися. Коли вузол-ініціатор збирає всі відповіді, він приймає рішення, які завдання може прийняти, відправляючи новий запит навантаженим вузлам. Навантажені вузли своєю чергою відправляють завдання, зменшуючи власне навантаження. Стратегія переговорів може ефективно використовуватись у розподіленій схемі балансування. На рис. 1.7 продемонстровано розподілену схему підходу переговорів, ініціатором у цьому випадку виступає виконавець.

Існує стратегія балансування навантаження, заснована на кластеризації вузлів мережі, що мають схожі характеристики, такі як пропускна здатність мережі, кількість та частота процесорів, дисковий простір тощо. Такий поділ можна використовувати для відображення груп завдань на окремі кластери. Ця стратегія добре поєднується з частково розподіленою схемою балансування.

Балансування навантаження також може здійснюватися з використанням відомих алгоритмів, натхненних природою, теорії ігор та генетичних алгоритмів .

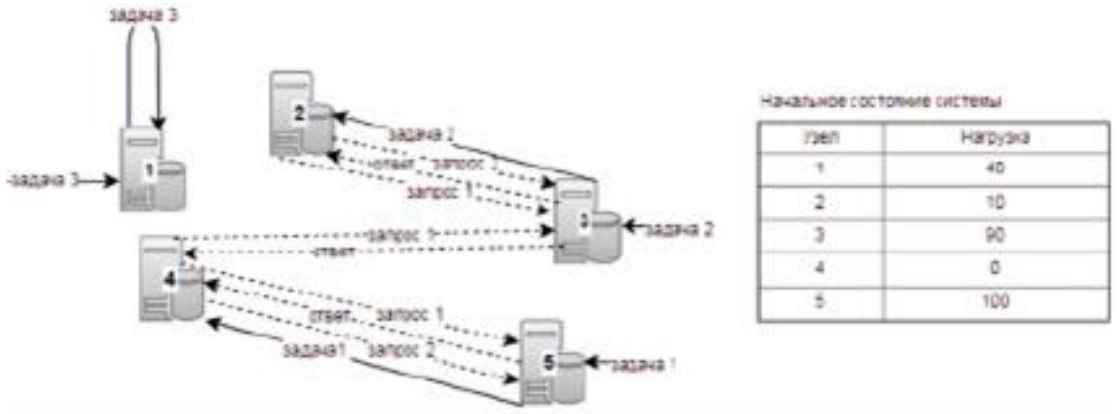


Рис. 1.7 – Розподілена схема стратегії переговорів

У таблиці 1.1 представлені переваги та недоліки розглянутих методів динамічного балансування навантаження.

Таблиця 1.1 – Переваги та недоліки розглянутих методів

Метод	Переваги	Недоліки
1	2	3
Випадковий вибір	Низьке навантаження на системи зв’язку та балансування	Низька ефективність Немає врахування поточного навантаження Немає врахування продуктивності вузлів
Циклічний алгоритм	Простота реалізації Низьке навантаження на системи зв’язку та балансування Спрямована зміна вузлів Врахування продуктивності вузлів (зважена версія)	Низька ефективність Немає врахування поточного навантаження
Стратегія найменшого навантаження	Облік поточного навантаження Висока ефективність (щодо стратегії випадкового вибору та циклічного алгоритму) Врахування продуктивності вузлів (зважена версія)	Витрати комунікацію (збір динамічної інформації)
Стратегія зондування	Облік поточного навантаження Висока ефективність Можливість урахування продуктивності вузлів Орієнтованість на розподілену схему	Витрати комунікацію (збір динамічної інформації)
Стратегія переговорів	Облік поточного навантаження Висока ефективність Варіанти ініціаторів Можливість урахування продуктивності вузлів Орієнтованість на розподілену схему	Витрати комунікацію (збір динамічної інформації) Складність реалізації

1.4 Висновок до першого розділу

Балансування навантаження в розподілених системах – це процес розподілу робочого навантаження системи між обчислювальними ресурсами, що розділяються. З появою високошвидкісних каналів зв’язку все частіше в якості вузлів розподілених систем використовуються автономні комп’ютерів, об’єднаних високошвидкісними лініями зв’язку. Основною перевагою таких систем є висока продуктивність та доступність при низьких витратах.

Внаслідок цього розподілені обчислення набувають все більшого значення як переважний метод обчислень порівняно з централізованою обробкою даних.

Сьогодні існує безліч підходів до вирішення проблеми балансування навантаження у розподілених системах, зокрема розподілені файлові системи не є винятком. Ці підходи можуть використовуватися для покращення якості обслуговування клієнтів, підвищення продуктивності системи, а також більш раціонального використання ресурсів. Як переваги ефективного балансування навантаження варто відзначити підтримку відмовостійкості та масштабованості системи.

РОЗДІЛ 2
АНАЛІЗ СТАТИЧНОГО ТА ДИНАМІЧНОГО ПІДХОДІВ ДО БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

На продуктивність будь-якого веб-сервера зазвичай впливає веб-трафік, і час обробки запитів сервером збільшується, оскільки він перевантажується. Через збільшення трафіку через Інтернет сервер стикається з проблемами обслуговування великої кількості клієнтів з мінімальною затримкою. Тому існує концепція конфедерації ресурсів. Популярний сервер Google працює за такою ж концепцією. Він розподіляє запит користувача по різних серверах, які географічно розташовані в різних місцях. Сервер вимагає кількох механізмів роботи з пов'язаними відкритими даними (LOD) для виконання запиту користувача.

Балансування навантаження відіграє важливу роль у роботі розподілених та паралельних обчислень. Цей підхід розподіляє вхідне робоче навантаження на менші завдання, призначені обчислювальним ресурсам для одночасного виконання. Навантаженням може бути ємність процесора, об'єм пам'яті, навантаження на мережу, затримка тощо. Причина балансування навантаження – обробка запитів кількох користувачів без погіршення продуктивності сервера. Балансувальник навантаження отримує запити від користувача, визначає навантаження на наявні ресурси та надсилає запит на сервер, який злегка завантажується.

Основні функції балансувальника навантаження такі [15]:

- Розподіляти вхідний трафік між кількома обчислювальними ресурсами;
- Визначати наявність ресурсів та їх надійність для виконання завдань;
- Покращувати використання ресурсів;
- Підвищувати задоволеність клієнтів;
- Забезпечувати відмовостійкість та гнучку структуру шляхом додавання або віднімання ресурсів у міру виникнення попиту.

Використання алгоритмів балансування сприяє досягненню вищої пропускної здатності та покращує час відгуку у розподілених системах. Однак кожен із цих алгоритмів має свої переваги та недоліки [16]. Основні методи балансування навантаження наведені на рис. 2.1.

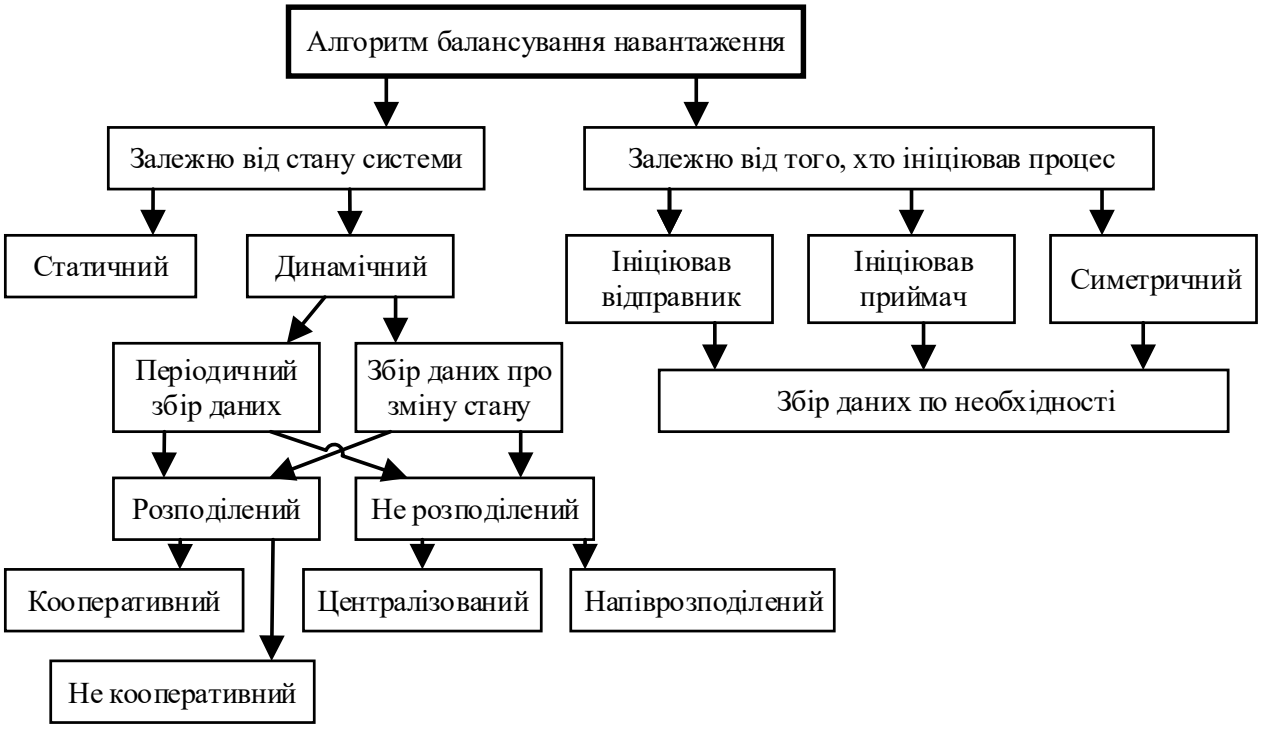


Рис 2.1 – Класифікація алгоритмів балансування навантаження

Ми пропонуємо кристуватися більш спрощеною схемою алгоритму балансування навантаження (АБН) і перш за все можна використовувати класифікацію таку, як показано на рис. 2.2 [16]. Статичні алгоритми (СБН) вимагають попередньої інформації про характеристики системи, такі як можливості обробки, пам’ять, кількість активних з’єднань тощо, тоді як алгоритми динамічного балансування навантаження (ДБН) використовують поточний стан системи для прийняття рішень щодо планування.

Щоб отримати згадані вище переваги від балансування, важливо вибрати відповідний АБН для веб-ресурсів у розподіленому середовищі.

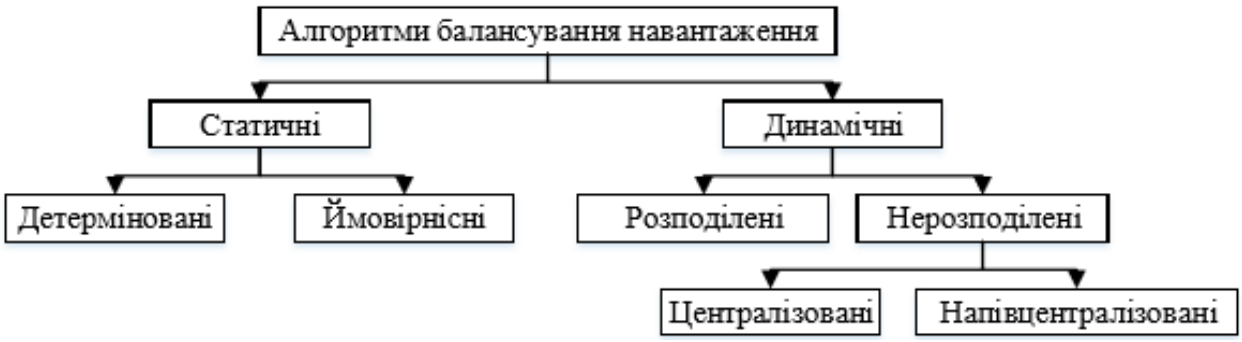


Рис. 2.2 – Класифікація алгоритмів балансування навантаження

На основі характеристик ініціації процесу АБН можна ввести додаткову, уточнюючу класифікацію на три підкатегорії [16]:

- Ініційовані відправником;
- Ініційовані одержувачем;
- Ініційовані симетрично.

Алгоритми, ініційовані відправником та алгоритми, ініціатором в яких є одержувач, використовують різні політики передачі та призначення для реалізації балансування навантаження. Симетрично ініційовані алгоритми усувають умову випередження, яка є характерною для алгоритмів, ініційованих приймачем, і пропонують два алгоритми: алгоритм вище середнього (above-average) та адаптивний алгоритм. Алгоритм вище середнього використовує прийнятний діапазон для прийняття рішення про те, чи є вузол ініційований відправником або одержувачем. Адаптивний алгоритм приймає рішення на основі певного набору правил та показників, що зазвичай забезпечує більшу гнучкість.

У таблиці 2.1 наведено порівняння існуючих алгоритмів балансування навантаження по можливості їх оцінювання за різними показниками продуктивності [17-19].

Таблиця 2.1 – Порівняльний аналіз алгоритмів за показниками продуктивності

Алгоритм / Метрика	Опис алгоритму	Пропускна здатність	Час міграції	Час відповіді	Відмовостійкість	Витрати	Використання ресурсів	Масштабованість	Продуктивність
1	2	3	4	5	6	7	8	9	10
Round Robin Sheduling	Управління завданнями в системах з розподілом часу – алгоритм, що використовує кільце черги з фіксованим квантом часу. Кожне завдання виконується протягом цього проміжку та у порядку черги. При великому кванті часу алгоритм Round Robin подібний до FCFS Scheduling. За невеликого кванту часу він відомий як алгоритм Processor Sharing.	так	ні	так	ні	так	так	ні	так
Max-Min	У цьому алгоритмі спочатку визначається мінімальний час виконання завдань, обираючи максимальне значення серед усіх завдань та ресурсів. Потім завдання з максимальним часом призначається на конкретно обраний вузол. Після цього час виконання всіх завдань на цьому вузлі перераховується, додаючи час виконання поставленого завдання до часу виконання інших завдань на цьому вузлі. Крім того, поставлене завдання видаляється зі списку системи.	так	ні	так	ні	так	так	ні	так
Compare and Balance	Використовується для забезпечення рівноваги та управління незбалансованим навантаженням системи. У цьому алгоритмі, відповідно до ймовірності (залежно від номера віртуальної машини, яка працює на поточному хості та в цілій хмарній системі), поточний хост випадковим чином обирає хост та порівнює їх навантаження для передачі додаткового навантаження на обраний вузол. Алгоритм балансування навантаження також спроектований та реалізований для скорочення часу міграції віртуальних машин, використовуючи спільну пам’ять з метою зменшення часу міграції віртуальних машин.	так	так	так	так	так	ні	так	так

Продовження таблиці 2.1

1	2	3	4	5	6	7	8	9	10
Ant Colony Optimization	Це розподілений алгоритм, де інформація про ресурси системи динамічно оновлюється при кожному русі мурах. Кожен вузол розсилає кольорові колонії по мережі, щоб уникнути руху мурах одним маршрутом і забезпечити їх розподіл по всіх вузлах системи. Кожен мураха виступає як мобільний агент, передаючи оновлену інформацію про баланс навантаження до наступного вузла.	так	ні	ні	ні	так	так	так	так
Shortest Response Time First	Основна ідея алгоритму – пряма переадресація. Кожному процесу встановлюється пріоритет для запуску, а процеси з однаковими пріоритетами обслуговуються в порядку «перший прийшов, перший обслужений» (FCFS). SJF – це варіант алгоритму пріоритетного планування, де пріоритет залежить від тривалості активності процесора (CPU). Політика SJF сприяє виконанню завдань з найменшим часом обробки, що призводить до пріоритетного виконання коротших завдань перед довшими. Важливо точно визначити час обробки для кожного завдання, що є головною трудністю SJF.	ні	так	так	ні	так	так	ні	так
Active Clustering	Алгоритм самоагрегації оптимізує завдання, об’єднуючи подібні послуги через локальне перегрупування на основі формування груп схожих вузлів за допомогою концепції рефері вузла. Цей метод подолає неоднорідність, адаптується до динамічного середовища, має високу стійкість до помилок і демонструє ефективну масштабованість для поліпшення продуктивності системи.	ні	так	ні	ні	так	так	ні	ні
Join-Idle-Queue	Алгоритм балансування навантаження для динамічно масштабованих веб-сервісів розподіляє навантаження між відправниками з урахуванням їхньої масштабованості. Починаючи з оцінки доступності процесорних ресурсів на кожному відправнику, він призначає завдання процесорам для зменшення середньої довжини черги на кожному процесорі. В процесі видалення завдань балансування навантаження з критичного шляху обробки запиту ефективно зменшує навантаження системи, уникавши комунікаційного навантаження на нові завдання, і не збільшує фактичний час відгуку.	так	так	так	ні	так	ні	так	так

Продовження таблиці 2.1

1	2	3	4	5	6	7	8	9	10
Central queuing	Цей алгоритм динамічного розподілу обробляє нові задачі, які надходять до менеджера черг. Кожна задача додається до черги. При виконанні запиту на обробку, менеджер видаляє першу задачу з черги і передає її виконання запитуючій стороні. У випадку відсутності готових завдань у черзі, запит буферизується до появи нової доступної задачі.	ні	так	так	ні	так	так	ні	так
Connection mechanism	Алгоритм балансування навантаження мінімізує кількість з’єднань у рамках динамічного алгоритму планування. Цей метод використовується для розрахунку кількості з’єднань для кожного сервера та для динамічної оцінки навантаження. Балансувальник навантаження відслідковує кількість з’єднань на кожному сервері, що збільшується при надходженні нового з’єднання та зменшується при завершенні або перериванні з’єднання.	так	ні	ні	так	так	так	так	ні
Least connections	Алгоритм мінімізації з’єднань направляє запити на сервер, який в даний момент має найменшу кількість активних підключень. Балансувальник навантаження відслідковує кількість з’єднань на серверах і направляє наступний запит на той сервер, який має мінімальну кількість з’єднань. Ефективність алгоритмів балансування навантаження оцінюється за декількома критеріями.	ні	ні	ні	так	так	ні	ні	ні

2.1 Статичне балансування навантаження

Підходи до балансування статичного навантаження використовують попередню інформацію про завдання, обчислювальні ресурси чи елемент обробки та деталі мережі, як показано на рис. 2.3.

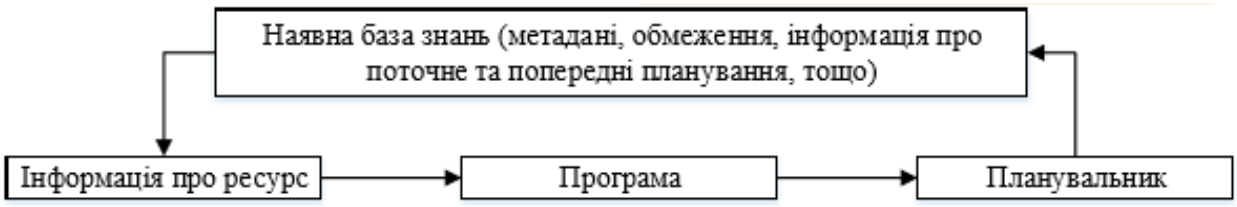


Рис 2.3 – Схема циклічного процесу статичного балансування навантаження

Завдання можна делегувати до будь-якого елемента обробки двома методами: без врахування стану системи (stateless, ймовірнісні) та з врахуванням стану системи (stateful, детерміновані).

У stateless методі без врахування стану вибір елемента обробки (ЕО) здійснюється без будьякого оцінювання системного середовища, тоді як у stateful методі вибір ЕО вимагає інформації про стан системи. Stateless методи прості у реалізації, але вони забезпечують взаємодію між клієнтом та сервером лише у форматі один до одного, як показано на рис. 2.4.

На рис. 2.5 схематично зображено процес статичного балансування навантаження зі stateful методом. Балансувальник навантаження відстежує всі сеанси, а рішення приймаються на основі завантаження серверів. Існують різні stateful методики для вибору такого елемента обробки, як Round Robin, зважений Round Robin та алгоритм випадкового розподілу. Однак, ці алгоритми мають обмежену сферу застосування через здебільшого динамічний характер розподілених середовищ.

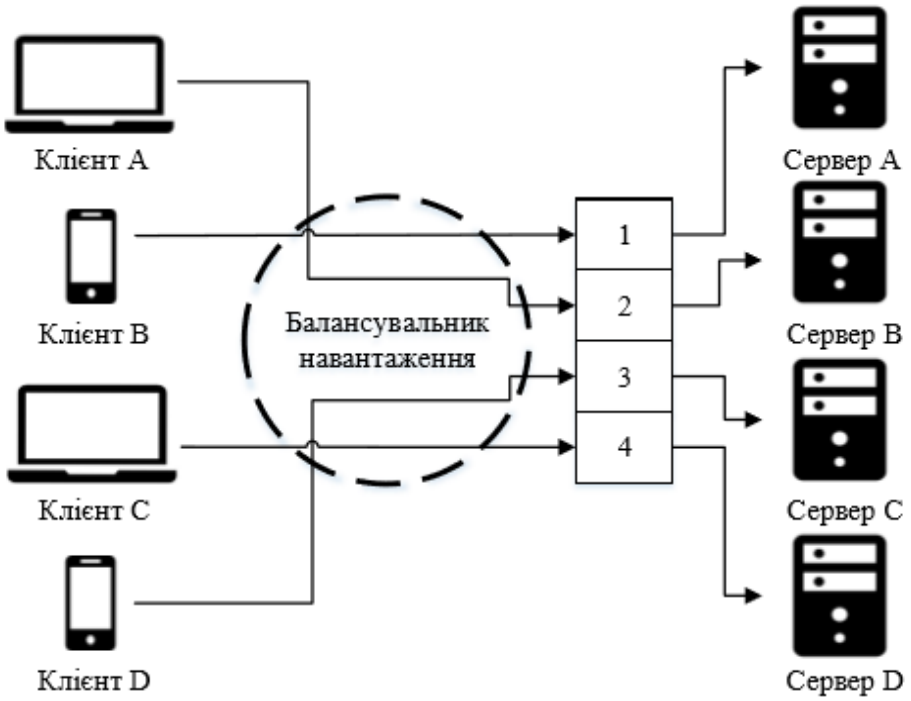


Рис 2.4 – Схема статичного stateless балансування навантаження в системі

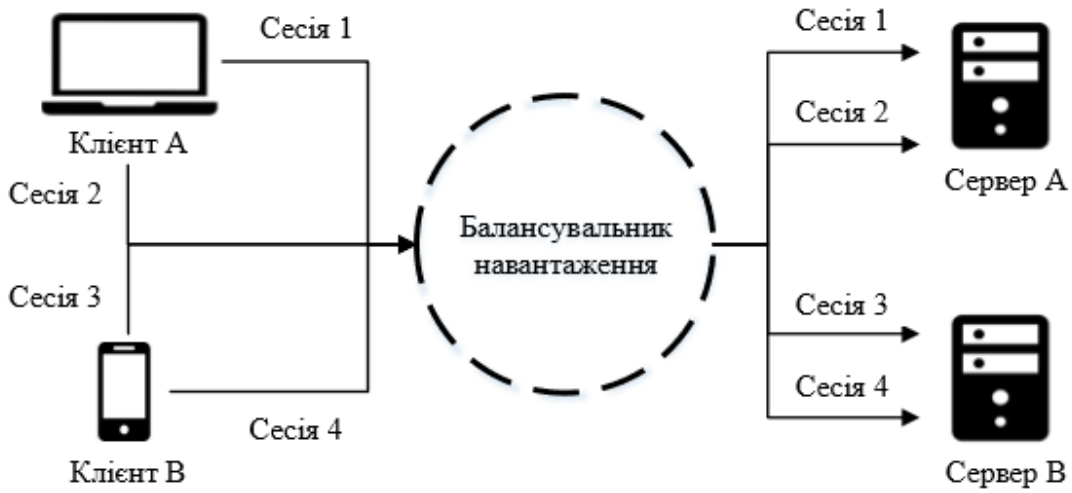


Рис. 2.5 – Схема статичного stateful балансування навантаження в системі

Статичний метод розподілу трафіку надає послідовний розподіл на сервери і потребує попереднього знання системних ресурсів. Цей підхід не залежить від поточного стану системи. Навпаки, динамічні алгоритми приймають рішення, опираючись на фактичний стан системи, і дозволяють рухати трафік в режимі реального часу, переміщаючи його з перевантажених машин на невикористані.

Алгоритм змішаного завантаження фокусується на симетричному розподілі обчислювального завдання для зменшення вартості зв’язку між розподіленими обчислювальними вузлами. Важливо розуміти, що хмарні обчислення відбуваються в динамічному середовищі, і тому динамічні алгоритми навантаження важливі для відгуку на зміни в завантаженні. Такий алгоритм може бути класифікований як періодичне планування пакетного режиму та планування негайного режиму.

Алгоритм планування повинен впливати на кілька характеристик [16; 20]:

- мінімізація часу очікування;
- мінімізація часу відгуку;
- максимізація пропускної здатності;
- максимізація завантаженості процесора.

2.2 Динамічне балансування навантаження

Динамічне балансування навантаження являє собою розвиток stateful алгоритмів СБН, в яких запити клієнтів розподіляються між доступними ресурсами під час виконання [16]. LB призначає запит на основі динамічної інформації, зібраної з усіх ресурсів, як показано на рис. 2.6.

Алгоритми ДБН можна розділити на дві категорії – розподілені та нерозподілені.

У розподіленому алгоритмі ДБН всі обчислювальні ресурси однаково відповідають за балансування навантаження. В нерозподілених алгоритмах кожен ресурс виконує свою роботу самостійно для досягнення спільної мети. Як правило, розподілені алгоритми ДБН генерують більше службових повідомлень, ніж нерозподілені алгоритми ДБН, завдяки своїй безпосередній взаємодії з усіма ресурсами системи. Розподілені алгоритми ДБН працюють краще в умовах відмови вузлів, оскільки випадкова несправність вузла

погіршує лише отриманий розподіл сумарної задачі в системі замість зведення нанівець загальної продуктивності системи

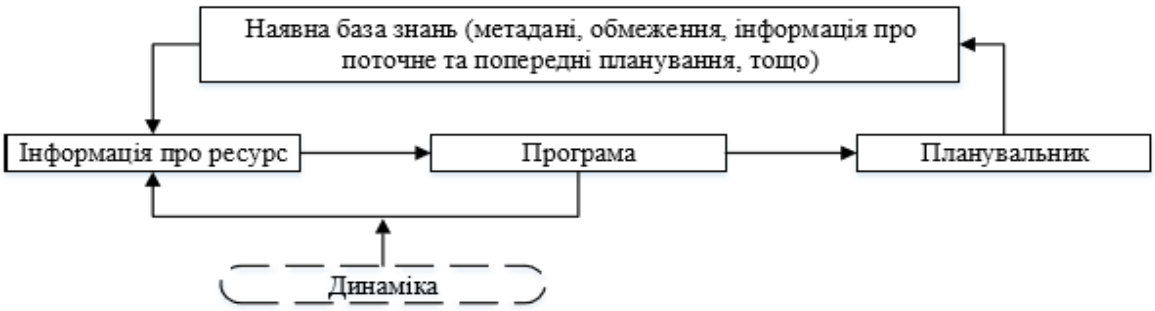


Рис. 2.6 – Схема циклічного процесу динамічного балансування навантаження

Нерозподілені алгоритми ДБН в свою чергу також поділяються на дві категорії – централізовані та напівцентралізовані. У централізованому алгоритмі центральний сервер відповідає за виконання алгоритму балансування навантаження. У напівцентралізованих сервери розташовані в кластерах, а балансування навантаження всередині кластера управляється централізовано.

Динамічне балансування передбачає перерозподіл обчислювального навантаження між вузлами під час виконання програми. Програмне забезпечення, що реалізує динамічне балансування, враховує різні фактори, такі як завантаження обчислювальних вузлів, пропускну здатність ліній зв’язку та частоту обмінів повідомленнями між логічними процесами розподіленого додатка.

Балансування навантаження може використовувати як апаратні, так і програмні інструменти. Застосування динамічного балансування з урахуванням поточного завантаження серверів дозволяє створити хмарне середовище, яке оптимально використовує наявні ресурси.

Алгоритми динамічного навантаження не потребують попередніх знань про дії в роботі або глобальний стан системи. Вони базуються на існуючому або поточному стані системи. У розподіленій системі такі алгоритми

виконуються всіма вузлами, які присутні в системі, і розподіляють завдання балансування навантаження між ними. Взаємодія між вузлами для здійснення балансування може бути кооперативною або некооперативною.

У кооперативній формі вузли працюють спільно для досягнення спільної мети, такої як покращення часу відгуку. У некооперативній формі кожен вузол працює незалежно для досягнення локальних цілей. Динамічні алгоритми, що мають розподілений характер, можуть генерувати більше повідомлень, оскільки кожен вузол повинен взаємодіяти з кожним іншим вузлом. Проте це забезпечує продуктивність системи, навіть якщо окремі вузли виявляються неактивними.

У нерозподіленому типі балансування навантаження виконується або одним вузлом, або групою вузлів. Алгоритми динамічного розподілу навантаження нерозподіленого характеру можуть бути централізованими або напіврозподіленими. У централізованому підході розподіл навантаження відбувається через один вузол – центральний вузол. У напіврозподіленому варіанті вузли поділяються на кластери, кожен з яких має централізований центр. Обидва типи мають свої переваги і недоліки, зокрема централізовані алгоритми можуть викликати вузькі місця на центральному вузлі, тоді як напіврозподілені алгоритми забезпечують більшу масштабованість.

2.3 Порівняльний аналіз підходів

Балансувальник навантаження реалізує кілька алгоритмів балансування навантаження для визначення відповідного ресурсу. Проте він стикається з кількома проблемами при розподілі навантаження між доступними ресурсами. У наступному розділі представлено кілька основних проблем із відповідними рішеннями.

2.3.1 Збільшення веб-трафіку

За останні кілька років веб-трафік дуже швидко збільшується завдяки численним зареєстрованим веб-сайтам та онлайн-транзакціям. Зі збільшенням кількості запитів відповідь сервера стає повільною через обмежену кількість відкритих з'єднань. Запити додаються до загальної можливості обробки ресурсів. Коли вхідні запити перевищують можливості ресурсу, ресурс аварійно завершує роботу або виникає помилка. Кілька авторів проаналізували та запропонували рішення для вирішення проблеми. Перше рішення - це оновлення сервера, в якому на деякий час запити обробляються більш потужним сервером. Але з цим рішенням пов'язані проблеми масштабованості, переривання та обслуговування. Іншим рішенням є аутсорсинг, при якому запити надсилаються на інший відповідний сервер для швидкої відповіді. Але цей підхід коштує дорого і має обмежений контроль над проблемами якості обслуговування. Розмір вебсторінки та кількість користувачів виказують значний вплив на час відгуку системи.

Найбільш сприятливим рішенням є використання декількох серверів з ефективним балансуванням навантаження, яке врівноважує навантаження між серверами. Продуктивність цих серверів аналізується за допомогою моделей черги або моделей ліній очікування. Загалом, два типи моделей балансування навантаження використовуються для аналізу продуктивності сервера. Кожен підхід має свої переваги, застосування та обмеження.

Модель централізованої черги для балансування навантаження

У цьому механізмі використовуються однорідні сервери з кінцевими розмірами буфера. Балансувальник навантаження отримує запит від клієнта і перенаправляє запит між серверами, використовуючи одну з таких політик маршрутизації:

- Випадкова політика;
- Політика RR;
- Політика найкоротших черг.

Порівняння цих політик з точки зору частоти відхилень та часу відгуку системи показує, що ці алгоритми добре працюють, коли рух невеликий, але коли веб-трафік стає високим, політика найкоротших черг працює краще, ніж політика випадкових та RR. Кількість відхилень у RR і випадковій політиці збільшується зі збільшенням трафіку. Існує запропонований алгоритм черги для вимірювання перевантаження та обслуговування сервісних можливостей сервера в середовищі з ДБН. Алгоритм працює краще як в однорідному, так і в гетерогенному середовищі, ніж алгоритми залишкової ємності (RC) та алгоритми черги на основі вмісту сервера (QSC).

Модель розподіленої черги для балансування навантаження

Ці механізми також вирішують проблему затримки мережі, що дозволяє уникнути перевантаження мережі. Моделі черги дотримуються певних правил прибуття та розподілу для розподілу запитів. Моделі розподіленої черги добре працюють в умовах інтенсивного руху [5]. Рішення про маршрутизацію приймаються на основі різниці в довжині черги веб -серверів. Зібрана інформація використовується для розподілу трафіку для покращення продуктивності веб-серверів. Час виконання завдання безпосередньо впливає на довжину черги веб-сервера, тому доцільним є застосування моделі на основі коефіцієнту співвідношення середнього часу виконання завдання.

Для нормалізації затримки мережі існує підхід до балансування навантаження з контролем затримки для покращення продуктивності мережі. Однак підхід має обмежену застосовність і підходить для стабільних станів шляху.

Також існує підхід на основі теорії черг для моніторингу мережевого трафіку, при якому моделювання виконується в однорідному, а також в гетерогенному мережевому середовищах. Моніторинг трафіку необхідний для розрахунку параметрів достовірності та ефективності від постійної роботи мережі. Виходячи з теорії черг і закону Малта, перевантаженість мережі збалансована.

2.3.2 Вибір ресурсів та розподіл завдань

Багато дослідників зверталися до проблеми вибору ресурсів та розподілу завдань для справедливого балансування навантаження. Балансувальник навантаження бере на себе всю відповідальність за відображення ресурсу та завдання до фактичного виконання.

Управління ресурсами складається з двох основних функцій: забезпечення ресурсами та планування ресурсів. Під час надання ресурсів користувач надсилає брокеру завдання з різними заздалегідь визначеними обмеженнями якості обслуговування. Брокер несе відповідальність за пошук відповідного ресурсу для виконання завдань. Планування ресурсів - це відображення та виконання завдання на відповідному ресурсі. Він складається з трьох основних функцій: відображення ресурсів, виконання ресурсів та моніторинг ресурсів.

Класифікувати ресурс можна за трьома категоріями - недовантажений, нормально завантажений та перевантажений. Планувальник позначає завдання лише на недовантажені або нормально завантажені ресурси. З іншого боку, можна розділити ресурси на дискретні рівні для вибору найбільш придатного ресурсу для виконання завдань. Також можна представити планування бюджетних завдань та розрахувати придатність усіх ресурсів для відбору ресурсів.

Для виконання завдання можна вивести функцію вартості для вибору ресурсу, значення якої обчислюється з використанням коефіцієнта завершення та історичної інформації про ресурс. Для мінімізації передачі даних між ресурсами можна використати гіперграф, який ідентифікує залежність завдань та даних. Завдання, які використовують схожі дані, призначаються одному і тому ж ресурсу для непрямого зниження вартості.

Перспективним є метод вибору ресурсів на основі бюджету для подільних навантажень. Метод призначає відповідний ресурс з точки зору вартості зі списку доступних ресурсів. Метод розподілу ресурсів для подільних навантажень існує також на основі лінійного програмування.

Робота класифікується у відповідних розмірах для розподілу за наявними ресурсами. Існує також вдосконалення методу вибору ресурсів шляхом визначення навантаження на завдання та наявності ресурсів відповідно.

Різні дослідники запропонували численні методи розподілу неоднорідних завдань. Наприклад удосконалення традиційного планування завдань кругової роботи (RR), яке добре працює, коли всі ресурси мають однакову пропускну здатність. У гетерогенному середовищі це не дає помітних результатів. Тому кожному серверу призначається вага, яка відображає також пріоритет вибору сервера. Алгоритм справляється з розподілом навантаження більш ефективно, ніж класичний алгоритм планування RR.

Існує також алгоритм планування завдань в хмарному та туманному середовища, на основі асоціації між вузлами туману та хмарними вузлами, щоб зменшити пропускну здатність та ціну хмарних ресурсів. Якщо обчислення неможливе на туманному вузлі, то завдання виконуються на хмарному вузлі. Кілька обмежень, таких як термін та бюджет, можуть підвищити ефективність та придатність алгоритму.

В рамках інтернету речей (IoT) також існують алгоритми планування завдань в ДБН. Наприклад, для вбудованих систем для підвищення продуктивності реальних додатків у парадигмі CloudIoT можна застосувати підхід з використанням мереж реального часу, де чітко дотримуються часових обмежень. Алгоритм підвищує швидкість планування виконання завдань у режимі реального часу в гетерогенних веб-серверах. Для багатохмарного середовища IoT існує алгоритм планування завдань, при якому найменш завантажений сервер вибирається глобальним диспетчером для планування завдань IoT.

Існують також генетичні алгоритми для вирішення проблеми планування завдань. Наприклад, за допомогою мета-евристичних методів генетичний алгоритм планування здатен знижувати вартість зв’язку між процесорами.

2.3.3 Вимірювання навантаження

Вимірювання навантаження є дуже важливою діяльністю в розподіленому середовищі. Різні алгоритми балансування навантаження визначають стан завантаження ресурсів ще до реальної реалізації завдання. Різні показники продуктивності, такі як відмовостійкість, час очікування, час відгуку тощо, можуть бути ефективно оптимізовані шляхом вимірювання поточного навантаження ресурсу. Багато авторів зверталися до цього питання і представляли різні методи забезпечення ресурсами для ефективного розподілу вхідного навантаження.

Перед призначенням завдання ресурсу планувальник перевіряє поточне завантаження кожного ресурсу та вибирає недовантажений або нормально завантажений ресурс для виконання завдання. Довжина завдання, ємність оброблюваного елемента та обмеження терміну - це фактори, які враховуються для визначення поточного навантаження кожного ресурсу. Якщо ресурс перевантажується, незавершені завдання переносяться на інший відповідний ресурс для завершення їх виконання. Механізм контрольної точки використовується для збереження та відновлення стану завдання, що значно зменшує середній час відгуку та час повторного подання завдання та покращує пропускну здатність системи.

В якості показнику навантаження можна визначити динамічне порогове значення на основі стандартного відхилення для балансування навантаження та міграції робочих місць. Для міграції завдань ресурси класифікуються, а середнє навантаження кожного кластера порівнюється з пороговим значенням елемента обробки. Для балансування навантаження завдання вибираються випадковим чином з колекції недовантажених або перевантажених ресурсів.

Існує модель вибору найкращого кластеру з точки зору підвищення надійності системи та зменшення споживання енергії, а також ефективного збалансування навантаження на систему. Клієнт може визначити пріоритетність свого вибору, щоб вибрати відповідний кластер для виконання

завдань. Ефективний підхід до визначення накладних витрат на міграцію робочих процесів може підвищити адаптивність моделі до реальних сценаріїв.

2.3.4 Оптимізація витрат

Алгоритм балансування навантаження відображає завдання на різні гетерогенні ресурси на основі заздалегідь визначених цілей. Основна мета балансувальника навантаження – оптимізувати час виконання завдань, вартість ресурсів та їх використання. Тому під час балансування розглядаються питання вартості та пропонуються деякі можливі рішення для його оптимізації.

Перш за все це може бути адаптивне планування робочих процесів, з врахуванням вартості ресурсів та вартості зв’язку між завданням та ресурсами. Через неоднорідність ресурсів кінцева вартість розраховується періодично. Реалізувати це можна у форматі двофазного алгоритму планування завдань, з фазою вибору завдань та фазою вибору процесора. Для вибору завдання пріоритет визначається шляхом обчислення рангу. Для вибору процесора обчислюється цінність усіх процесорів і вибирається процесор з найвищою цінністю.

В свою чергу, методи резервування забезпечують більше зниження витрат, ніж методи на вимогу. При цьому методі, загалом, виділення ресурсів передбачає такі три етапи:

- Бронювання ресурсів;
- Розширення ресурсів;
- Ресурс на вимогу.

На першому етапі хмарний брокер заздалегідь організовує ресурси, не враховуючи вимоги клієнта. На другому етапі співставляються вимоги клієнта та вартість ресурсів, а також виявляється надмірне або недостатнє використання ресурсів. Якщо вимога клієнта перевищує зарезервовані ресурси, брокер може попросити додаткові ресурси на основі оплати за

використання. Далі розпочинається етап на вимогу, при якому клієнт повинен знати відповідні майбутні вимоги, які важко оцінити в хмарному середовищі.

Існує також метод оптимізації витрат на основі активності процесу, при якому вартість обробки та час очікування визначаються за допомогою часу діяльності, використання ресурсів та коефіцієнта мінливості для перевірки ефективності.

У разі використання хмарних середовищ, можна класифікувати хмари за трьома категоріями залежно від наявності ресурсів: загальнодоступна хмара, приватна хмара та гібридна хмара. Користувач може скористатися послугами публічної хмари за допомогою методу оплати за використання. Приватні хмари належать приватним особам і пропонують безкоштовні різноманітні послуги. У гібридній хмарі ресурси із загальнодоступної хмари агрегуються відповідно до вимог.

Для планування завдань в хмарному середовища є підхід з направленням діяльності на скорочення завдань. Цей алгоритм працює краще, ніж традиційні підходи до призначення завдань з точки зору зниження витрат.

Ефективне забезпечення ресурсами відіграє життєво важливу роль у зниженні витрат на виконання завдань. Одним з алгоритмів, які вирішує цю проблему, є алгоритм забезпечення ресурсами на основі оптимізації рою частинок (PSO). PSO прийнято для вибору відповідного ресурсу для оптимізації витрат. Оцінюються і порівнюються з іншими існуючими методами три показники ефективності: виконання завдань, використання пам’яті та вартість. Результати моделювання показують, що алгоритм на основі PSO забезпечує мінімальний час виконання та використання пам’яті з найменшими витратами, ніж інші найсучасніші методи.

Розподіл ресурсів також можливий на основі аукціону для задоволення вимог клієнта та постачальника послуг. Під час планування ресурси призначаються користувачам, які мають найвищі ставки. Алгоритм забезпечує найбільший прибуток і задовольняє як клієнтів, так і постачальників послуг за кількома критеріями. Пов’язання механізму розподілу ресурсів з економічною

проблемою попиту та пропозиції забезпечує кращу функціональність із 17% прибутку за допомогою інших існуючих методів.

Планування завдань також необхідне для програм робочого процесу. Ці програми робочого циклу складаються із залежного завдання з обмеженнями терміну виконання. Мета полягає у виборі найменш витратного хмарного ресурсу через PSO для виконання завдань на основі робочого процесу. Ефективне планування завдань скорочує час виконання, що безпосередньо впливає на кінцеву вартість. Враховуючи накладні витрати на зв'язок, ефективність моделі та її застосування можна підвищити в реальному хмарному середовищі.

Мобільні пристрої значною мірою залежать від хмарних ресурсів для доступу до даних та виконання операцій. Мета планування та забезпечення ресурсами в цьому випадку полягає в тому, щоб вибрати оптимальний ресурс за мінімальну вартість. Постачальник послуг виконує завдання на відповідних ресурсах, щоб отримати максимальний прибуток.

2.3.5 Відмовостійкість

Відмовостійкість – це механізм, який забезпечує оцінювані результати якості навіть за наявності несправностей в системі. Система з її компонентами та службами може вважатися надійною лише за умови, що вона має відмовостійкість. Таким чином, проблема відмовостійкості привертає увагу дослідників протягом останніх десятиліть.

Методи відмовостійкості можна розділити на дві категорії: проактивні та реактивні. Проактивні методи – це методи запобігання, які визначають контрольований стан на відмовостійкість до їх виникнення. Системи постійно контролюються для оцінки несправностей. Проактивна відмовостійкість може бути реалізована трьома способами: самовідновлення, попереджувальна міграція та омолодження системи. У процесі самовідновлення періодично застосовуються процедури відновлення несправностей для автономного відновлення. Під час попереджувальної міграції завдання переносяться з

ймовірно несправного ресурсу на інший ресурс. Омолодження системи - це механізм, за допомогою якого періодично створюються резервні копії для очищення та видалення помилок із системи.

Інша категорія – це реактивні підходи, які мають справу з несправностями після їх виникнення. Реактивна відмовостійкість також може бути реалізована трьома способами: реплікації завдань, міграція завдань та контрольна точка. Під час реплікації завдань кілька екземплярів або копій одного завдання стають доступними на різних ресурсах. Якщо один екземпляр не працює, завдання виконується на іншому екземплярі. При міграції завдань завдання мігруються на інший відповідний ресурс для завершення його виконання. У контрольній точці стани завдань періодично зберігаються і перезапускаються з останнього збереженого стану, а не з самого початку. Існує кілька запропонованих механізмів відмовостійкості та рішень для відновлення.

На основі розгляду проблеми з недостатністю ресурсів представлено механізм відновлення на основі контрольних точок для виконання завдань. Якщо завдання не завершує виконання у встановлені терміни, то для завершення його вибирається інший відповідний ресурс. Перш ніж перенести його на інший відповідний ресурс, стан завдання зберігається і відновлюється для подальшого виконання через контрольну точку. Це призводить до більшого скорочення часу виконання, часу відгуку та покращення пропускної здатності, порівняно з іншими існуючими методами.

Проте, як правило, сама по собі контрольна точка збільшує час виконання, що безпосередньо впливає на вартість виконання. Тому можна використовувати процес надмірної техніки, щоб скоротити час виконання завдання. Ця методика досить хороша і зменшує накладні витрати на контрольнопропускні пункти до 40%.

Щоб забезпечити планування відмовостійкості в хмарному середовищі також необхідною є ідентифікація зловмисників. Будь-який користувач, який користується лише хмарними службами та відхиляє інші запити,

розглядається як зловмисний користувач. Репутація розраховується для визначення шкідливих користувачів. Такий підхід може бути реалізовано для підвищення надійності мережі та часу виконання завдань у хмарній парадигмі.

Стійкість до помилок на основі реплікації призводить до витрачання великої кількості ресурсів, а також йде на компроміс з найменшим можливим часом виконання. Для вирішення цієї проблеми існує механізм планування відмовостійкості, що забезпечує успішне завершення виконання завдання. Обмеження реплікації уникається шляхом перенесення завдання на подальше виконання. Якщо планувальник виявляє помилку, він переназначає завдання на інший відповідний ресурс і зберігає втрату ресурсів. Цей механізм зменшує споживання ресурсів та час виконання завдань. Однак передбачаються витрати на впровадження планування, що обмежує застосовність моделі в реальному сценарії.

Для ведення історії ресурсів Інформацію про несправності в системі можна представити у форматі індексу несправності. Індекс несправності визначається на основі успішного та невдалого виконання завдання на конкретному ресурсі. На основі значення індексу несправності, брокер повторює завдання, яке можна використовувати при виникненні несправності. Бюджетні та часові обмеження також враховуються під час планування виконання завдань. Такий механізм задовольняє різним вимогам якості обслуговування, підвищує надійність, а також послідовно працює за наявності несправності.

Поєднання двох методів відмовостійкості: гібридизацію альтернативного завдання за допомогою механізму повторної спроби та контрольної точки дозвляє успадкувати сприятливі аспекти їх обох. Результати моделювання показують, що альтернативне завдання з механізмом контрольної точки працює краще і покращує пропускну здатність системи, ніж інші існуючі методи.

Хмара полегшує зберігання та доступ до неоднорідних даних у розподіленій віддаленій мережі. Через динамічність, перевантаження мережі

та системні збої є ключовими факторами виникнення несправностей. Запобігання перевантаженню мережі та вибір відповідних серверів дозволяє уникнути умов несправності. Для забезпечення цього запропоновано концепцію квадратного матричного множення для управління мережевим трафіком та уникнення перевантаження мережі. Монітор ресурсів прогнозує умови несправності та використовує політику міграції, щоб уникнути збою системи. Представлений механізм відмовостійкості забезпечує знижену вартість з меншим споживанням енергії.

В результаті спостереження за умовами виникнення помилок було також запропоновано генетичний алгоритм для визначення ресурсних об’ємів для планування завдань. Представлений підхід підвищив надійність системи та скоротив час виконання завдань у мережевому середовищі.

2.3.6 Питання сумісності

Взаємосумісність означає ефективну міграцію та інтеграцію неоднорідних програм та даних для отримання безперебійних послуг у різних доменах. Існують різні розподілені програми для надання мільйонів послуг, що відрізняються за пропонованими послугами:

- Розподілені обчислення – це сукупність різних неоднорідних компонентів, розташованих у віддалених місцях, які координуються між собою шляхом передачі повідомлень. Кожен компонент або процесор має власну пам’ять.

- Сіткові (grid) обчислення – це мережа комп’ютерних ресурсів, пов’язаних для вирішення складної проблеми. Кожен ресурс слабо пов’язаний і виконує незалежне завдання для досягнення спільної мети. Grid обчислення можна класифікувати на основі масштабу та функціональних можливостей. На основі масштабу grid обчислення можна класифікувати на дві категорії, тобто кластерну сітку та сітку підприємства. Кластер означає групу однотипних сутностей. Тож кластерна сітка надає послуги груповому чи відомчому рівням.

Інший тип - це сітка підприємства, яка забезпечує спільне використання ресурсів у межах підприємства.

- Хмарні обчислення надають на вимогу комп'ютерні ресурси, такі як сховище чи обчислювальні ресурси, без безпосередньої участі користувачів. Він має ефективну систему управління даними та обчислювальної бази для паралельного виконання завдань для покращення різних показників якості обслуговування (QoS).

- Обчислення в тумані – це розширення хмарних обчислень, яке складається з декількох вузлів туману, безпосередньо підключених до фізичних пристроїв. Різниця між обома технологіями полягає в тому, що хмара - це централізована система, а туман – децентралізована.

- CloudIoT – це інноваційна тенденція, яка з'єднує та управляє мільйонами пристроїв, які розповсюджені по всьому світу, дуже економічно ефективними способами. Хмара може отримувати прибуток від IoT, щоб мати справу з реальними речами, використовуючи спільний ресурс високообчислювальних ресурсів, замість того, щоб мати локальні сервери чи персональні пристрої для обробки програм.

Пов'язані відкриті дані (LOD) надають новий вимір для різних проблем гетерогенної сумісності на основі архітектур веб-серверів. Ці питання потребують уваги для підтримки неоднорідних принципів опису, необхідних для роботи з різними даними з веб-ресурсів. Сумісність LOD дотримується підходу знизу вгору для встановлення міцних зв'язків між наборами даних. Різні дослідники вирішували питання сумісності LOD та представляли відповідні рішення для задоволення попиту користувачів.

2.3.7 Проблеми QoS

Користувач надсилає завдання з різними обмеженнями якості обслуговування (час виконання, витрата енергії, затримка тощо) для покращення продуктивності в розподіленому середовищі. Дослідники

вирішили кілька проблем забезпечення якості обслуговування та запропонували рішення для досягнення поставленої мети.

Виділена на основі спостереження за різними проблемами якості обслуговування (вартість, надійність, безпека та час для забезпечення ресурсами в мережевому середовищі) угода про рівень обслуговування (SLA) зменшила складність надання ресурсів, підтримуючи актуальну інформацію про всі ресурси. Цей підхід працює краще з точки зору використання ресурсів, вартості та задоволеності клієнтів.

В таблиці 2.2 згруповано найкращі з оглянутих підходів, технік та методів, розкрито їх сильні та слабкі або поки нереалізовані сторони.

Таблиця 2.2 – Основні переваги та недоліки деяких основних сучасних методів балансування навантаження

Основа методу та посилання на опис	Сильні сторони	Слабкі сторони / подальша робота
Розподіл ресурсів на основі витрат [16]	Вартість	Оптимізація продуктивності
Розподілення ресурсів [15]	Вартість	Бюджетні та часові обмеження
Паралельний розподіл ресурсів на основі виконання [17]	Вартість	Моделювання в реальній хмарі
Розподіл ресурсів на основі витрат [28]	Вартість	Обмеження часу
Передбачення часу виконання [11]	Час виконання	Відмовостійкість
Механізм відбору ресурсів [10]	Час виконання в найкращому випадку, показник завершеності, час повторного подання	Розгляд терміну виконання завдання
Механізм відбору ресурсів [12]	Час виконання, вартість	Відмовостійкість
Механізм планування завдань та вибору ресурсів [43]	Час виконання	Оптимізація витрат
Модель черги [6]	Час відгуку, швидкість падіння, використання сервера	Відмовостійкість
Модель вибору ресурсів [18]	Час відгуку, показник відкидання, використання сервера	Розгляд інших показників оптимізації
Вибір ресурсів та планування завдань [36]	Час виконання, час відгуку, пропускна здатність, час повторного подання	Механізм відмовостійкості та пов'язані з ним накладні витрати

2.4 Висновок за другим розділом

Підводячи підсумок, можна сказати, що сучасна система веб-серверів повинна завжди використовувати один або кілька методів ефективного балансування вхідного навантаження для розподілу між доступними їй веб-

ресурсами. Ці ресурси з кожним днем стають все дорожчими, тому необхідні ефективні механізми оптимізації витрат, особливо в рамках підтримки такої системи в невеликій організації.

В роботі було виявлено кілька проблем з балансуванням навантаження для ефективного використання веб-ресурсів у розподіленому середовищі та проведено огляд та аналіз цілого ряду сучасних алгоритмів та підходів до вирішення цих проблем. Був проаналізований детальний опис цих підходів, їх сильних та слабких сторін, перспектив застосування та обмежень, які вони накладають.

На основі вищезгаданих проблем балансування навантаження та виявлених недоліків сучасних алгоритмів балансування навантаження, було визначено кілька майбутніх вимірів, які будуть корисними для дослідницької спільноти для досягнення різних цілей:

- розробка моделі розподілу ресурсів, яка враховує характеристики як ресурсу, так і конкретного завдання для оптимізації різних показників якості обслуговування;
- розробка моделі збалансованості навантаження з відмовою для частково виконаних завдань через збій ресурсів та побудова політики вибору ресурсів для виконання завдань;
- аналіз контекстуальних відносин між проблемами CloudIoT та оптимізація їх за допомогою ефективного планування;
- розробка моделі прогнозування часу виконання для ефективного забезпечення ресурсами, вибору та планування.

РОЗДІЛ 3
ПРОПОНОВАНА МОДЕЛЬ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

3.1 Принципи програмно-конфігурованих мереж (SDN)

Для спрощення розширення мережевої інфраструктури CDN можна використовувати принципи програмованої конфігурації мереж SDN (Software Defined Networks, SDN) – це новий підхід до архітектури мережі, який надає гнучкість, керованість та економічну ефективність. Цей метод забезпечує потрібну швидкість, адаптабельність та автоматизацію, щоб постачальники послуг зв'язку (CSPs) залишалися конкурентоспроможними на сучасному ринку зв'язку. SDN передбачає відокремлення функцій управління мережею від функцій пересилання даних, централізацію інтелекту та абстрагування базової архітектури від додатків та служб. Інтегроване програмне забезпечення для управління надає CSPs можливість керувати та налаштовувати SDN через централізовану програмну утиліту, що дозволяє організовувати ресурси інтелектуально.

Цікавість IT-компаній щодо SDN пояснюється можливістю підвищення ефективності мережевого обладнання на 25-30%, зменшення витрат на експлуатацію мереж на 30%, перетворення управління мережами з мистецтва в інженерію, підвищення рівня безпеки та надання користувачам можливості програмно створювати та швидко впроваджувати нові сервіси у мережеве обладнання.

Основні дослідження в області SDN ведуться учасниками програми GENI (Global Environment for Network Innovations) для вивчення майбутнього Інтернету, Стенфордським та Берклійським об'єднаним центром з досліджень і розробок в галузі Internet2, а також за участю сьомої рамкової програми досліджень Європейського Союзу Ofelia та проекту FEDERICA.

Основні завдання SDN включають розділення процесів передачі та управління даними, створення уніфікованого інтерфейсу між рівнями управління і передачі даних, централізоване управління мережею через контролер з мережевою операційною системою та віртуалізацію фізичних ресурсів мережі.

3.2 Архітектура системи керування трафіку (TE)

На рис. 3.1 відображено загальну архітектуру системи, яку пропонується. Основні складові цієї системи включають:

Мережева інфраструктура дата-центру (DCN): Це основна мережева структура, яка визначається в межах центру обробки даних і забезпечує інфраструктуру для обміну даними між різними компонентами системи.

SDN контролер: Це логічне об’єднання, яке відповідає за керування мережевою інфраструктурою. Він взаємодіє з різними компонентами системи та забезпечує централізоване управління мережею.

Менеджер керування трафіком: Цей компонент відповідає за ефективний розподіл трафіку в мережі, забезпечуючи оптимальні умови для передачі даних.

Клієнт для віддаленого керування менеджером системи: Це інтерфейс, який дозволяє віддалено керувати роботою менеджера системи, забезпечуючи зручний інструмент для взаємодії з системою.

Система керування трафіком спрямована на ефективне управління мережею DCN, яка включає в себе значну кількість серверів. SDN роутери, розташовані в цій мережі, передають інформацію про свій трафік та виявлені помилки контролеру за допомогою взаємодійного інтерфейсу, такого як OpenFlow. Контролер, у свою чергу, агрегує надходжені дані для подальшого використання. Менеджер системи керування трафіком регулярно отримує агреговану інформацію для проведення аналізу.

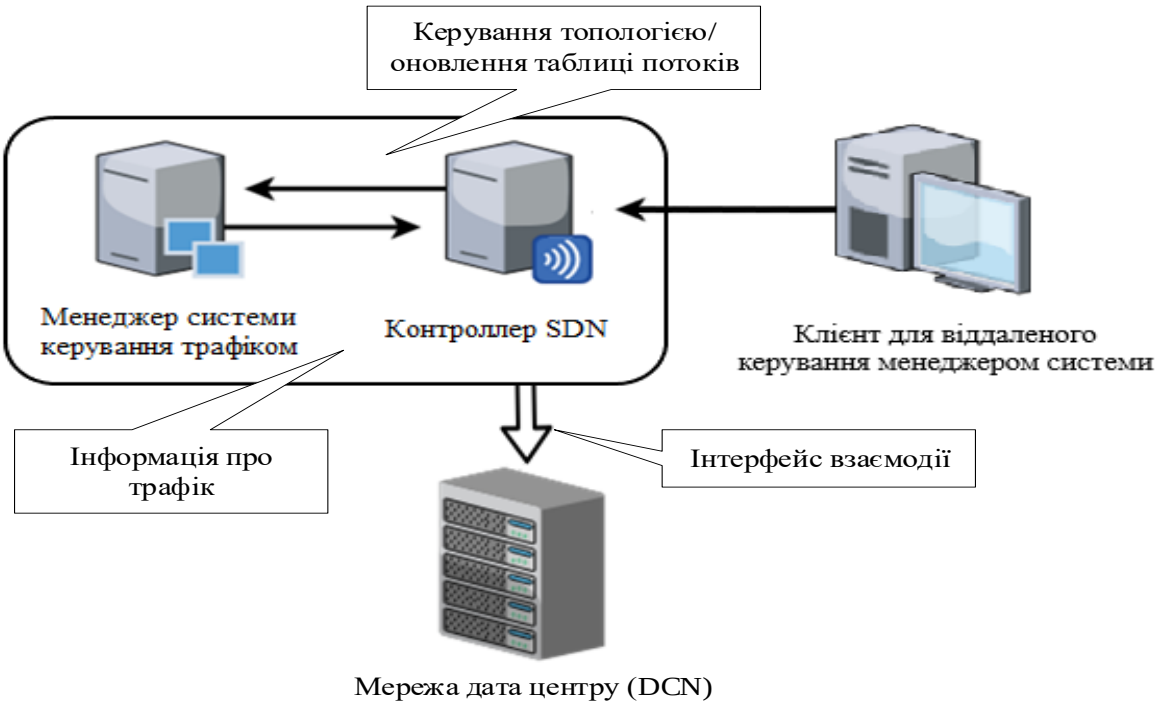


Рис. 3.1 – Загальна архітектура системи ТЕ для SDN

На підставі результатів аналізу менеджер приймає рішення стосовно керування трафіком та топологією DCN. Зміни, що стосуються керування трафіком, передаються контролеру, який, в свою чергу, впроваджує необхідні зміни до мережі DCN. Це може включати активацію або вимкнення SDN роутерів та налаштування зв'язків між ними з метою ефективного використання електроенергії та уникнення перевантаження окремих каналів зв'язку.

3.3 Опис евристичного алгоритму балансування

Успішність алгоритму балансування навантаження визначається його здатністю вчасно виконувати завдання. У цьому алгоритмі всі завдання розподіляються за принципом, схожим на алгоритм «Мурашине гніздо» [44]. Високорівневий алгоритм балансування (HLBA) використовує ідею поведінки колонії мурах при пошуку їжі та є дієвим методом розв'язання багатьох задач оптимізації комбінаторного характеру.

Схематично евристичний алгоритм балансування навантаження наведений на рис. 3.2.

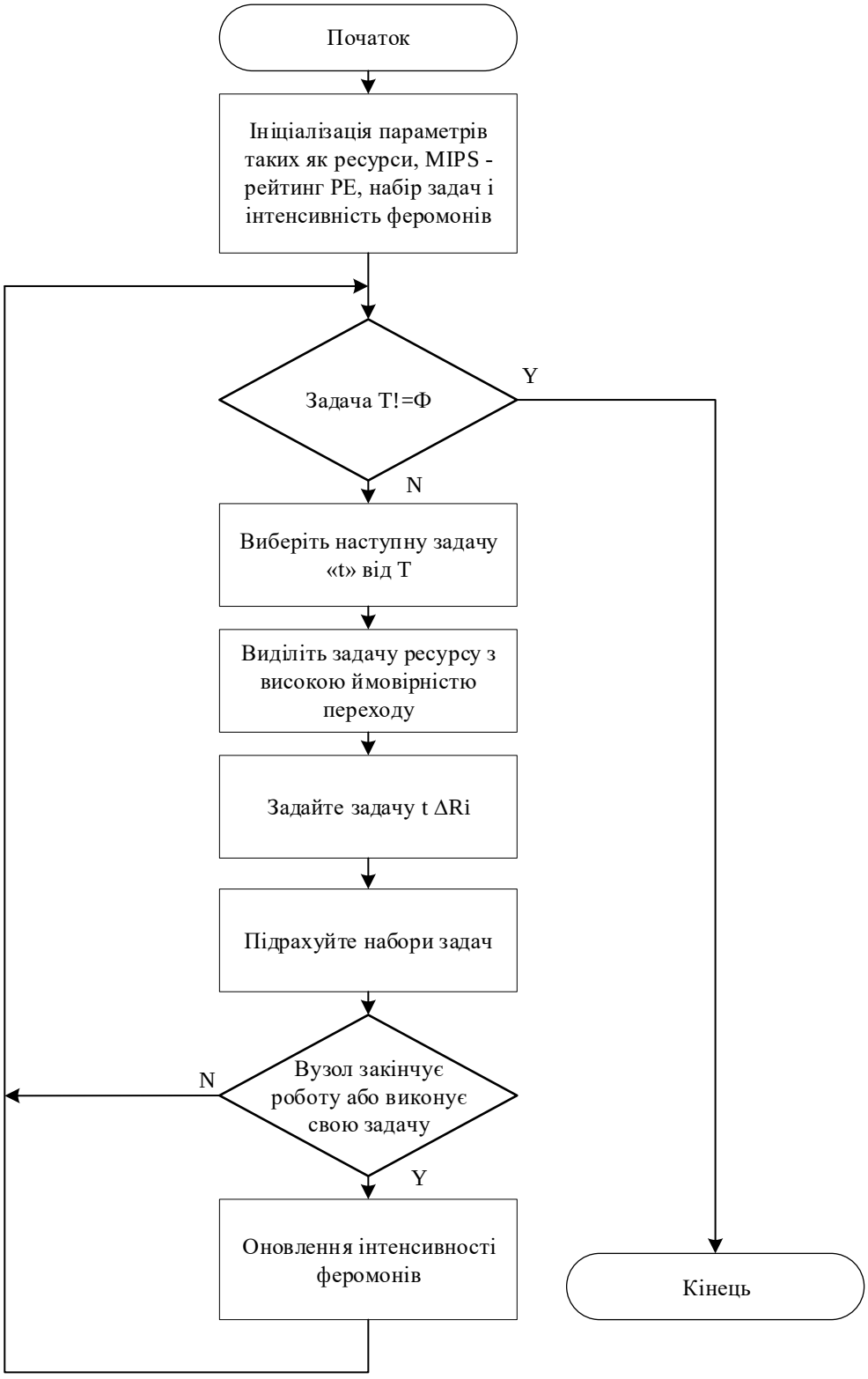


Рис. 3.2 – Евристичний алгоритм балансування навантаження

Наукові дослідження доводять, що мурахи мають унікальну здатність знаходження оптимального шляху від гнізда до їжі. Вони залишають на землі феромони під час руху, що служить позначкою для інших мурах. Коли мураха зустрічає раніше покладений слід, вона виявляє його і ймовірності слідувати за ним, тим самим зміцнюючи шлях своїми феромонами. В HLBA феромон пов'язаний з ресурсами, а не з конкретними шляхами. Кількість феромону збільшується чи зменшується в залежності від стану завдання у ресурсах.

Основною метою цього алгоритму є зниження загальної вартості і часу виконання. Процес балансування навантаження, який ґрунтується на евристичних принципах, описано нижче: Нехай T – кількість завдань (мурах) у наборі завдань, який обслуговується агентом завдань, P – кількість завдань, асоційованих із агентом завдань, та Q – загальна кількість доступних ресурсів.

Коли новий ресурс Ri реєструється у системі, він ініціює свій феромон, використовуючи формулу:

$$\tau_i(0) = N \times M, \tag{3.1}$$

де N – кількість обчислювальних елементів, а M – рейтинг MIPS обчислювального елемента.

Кожного разу, коли нове завдання призначається або завдання повертається з Ri, феромон Ri змінюється за допомогою формули:

$$i\tau_i^{new} = \rho \times \tau_i^{old} + \Delta\tau\,, \tag{3.2}$$

де $\Delta\tau$ представляє дисперсію феромону і ρ, 0 <ρ <1 є параметром розпаду феромону.

При призначенні завдання Ri феромон зменшується, тобто $\Delta\tau$ = -C, де C – обчислювальна складність завдання, а при успішному повереннні завдання Ri феромон збільшується і маємо $\Delta\tau$ = Ф × C, де фактор Ф є параметром заохочення.

Ймовірність призначення наступного завдання ресурсу Rj обчислюється за формулою:

$$p_j(t) = \frac{[\tau_j(t)]\ \alpha \times [\eta_j]\ \beta}{\sum r\ [\tau_j(t)]\ \ \alpha \times [\eta_j]}, \quad (3.3)$$

де j, r – індекси доступних ресурсів, $\tau_j(t)$ – поточний феромон ресурсу Rj, а ηj – початковий феромон Rj, тобто ηj = τj (0). Параметри α і β визначають відносну ефективність поточної інтенсивності феромонів та відносну важливість початкових характеристик продуктивності. Отже, алгоритм спроектований для впливу на продуктивність системи, що залежить від кількох факторів, які можуть бути спрощені для зменшення складнощів використаного методу.

Всі завдання розподіляються між ресурсами відповідно до ймовірності феромону або ймовірності переходу, що залежить від статусу завдання на конкретному ресурсі. Ресурс з найвищою інтенсивністю феромонів отримає завдання першим, до того, як будь-який інший ресурс отримає завдання з меншою цією інтенсивністю.

3.4 Опис модифікованого алгоритму на основі ACS (EDAFT) та оцінка його працездатності

Покращена динамічна стійкість відмов на основі ACS (EDAFT) представляє собою розширену версію алгоритму, інспірованого поведінкою мурашиного колонії у пошуках джерела їжі, шляхом побудови оптимального маршруту між гніздом і джерелом їжі. Цей аналогічний процес використовується для побудови оптимального маршруту між завданнями і ресурсами в гратчастій обчислювальній системі. У даному алгоритмі цей процес розширено так, що мурахи можуть проводити дослідження ресурсів під час повторних спроб на основі контрольних точок, з метою призначення будь-

якого невиконаного завдання альтернативним ресурсам з вищою ймовірністю успіху. Для додаткового удосконалення механізму відновлення феромону вводиться фактор довіри, щоб винагороджувати придатні ресурси або штрафувати непридатні ресурси, враховуючи історію виконання, з метою регулювання зменшення або збільшення рівня феромону. Очікується, що поліпшена формула оновлення феромону ефективно керуватиме призначенням завдань на основі придатності ресурсів, що в кінцевому підсумку може знизити ймовірність виникнення помилок.

Стандартний алгоритм ACS включає глобальне та локальне оновлення феромонів. У вдосконаленій версії EDAFT було удосконалено локальне оновлення феромону за допомогою введення фактора довіри. Таким чином, додатковий феромон додається, коли ресурс завершує виконання завдання або випаровує існуючий феромон, покращуючи ефективність алгоритму. Робочий процес високого рівня EDAFT зображено на рис. 3.3, де покращений процес виділено для зрозумілості.

Для кожного завдання у черзі генерується мураха для пошуку ресурсів, використовуючи значення феромону. Перед відправкою першого завдання у черзі розраховується початкове значення феромону для оцінки стану всіх ресурсів. Вибір ресурсу базується на рівнях феромону, отриманих з початкового розрахунку або процесу оновлення феромону. Коли завдання призначається будь-якому ресурсу, мураха використовує глобальне оновлення феромону, щоб зменшити кількість феромону, зробивши ресурс менш привабливим для наступного мурахи. Кожне призначене завдання розбивається на тимчасові контрольні точки, які фіксуються під час виконання. У випадку невдачі завдання піддається процесу перепланування і призначається альтернативний ресурс з останньої збереженої контрольної точки. Локальне оновлення феромону із штрафом застосовується до ресурсу, який не зумів зменшити інтенсивність феромону. У випадку успіху виконання, локальне оновлення феромону із заохоченням використовується для

збільшення феромона. Після виконання або невдачі ресурс звільняється для наступного призначення завдання після процесу оновлення феромону.

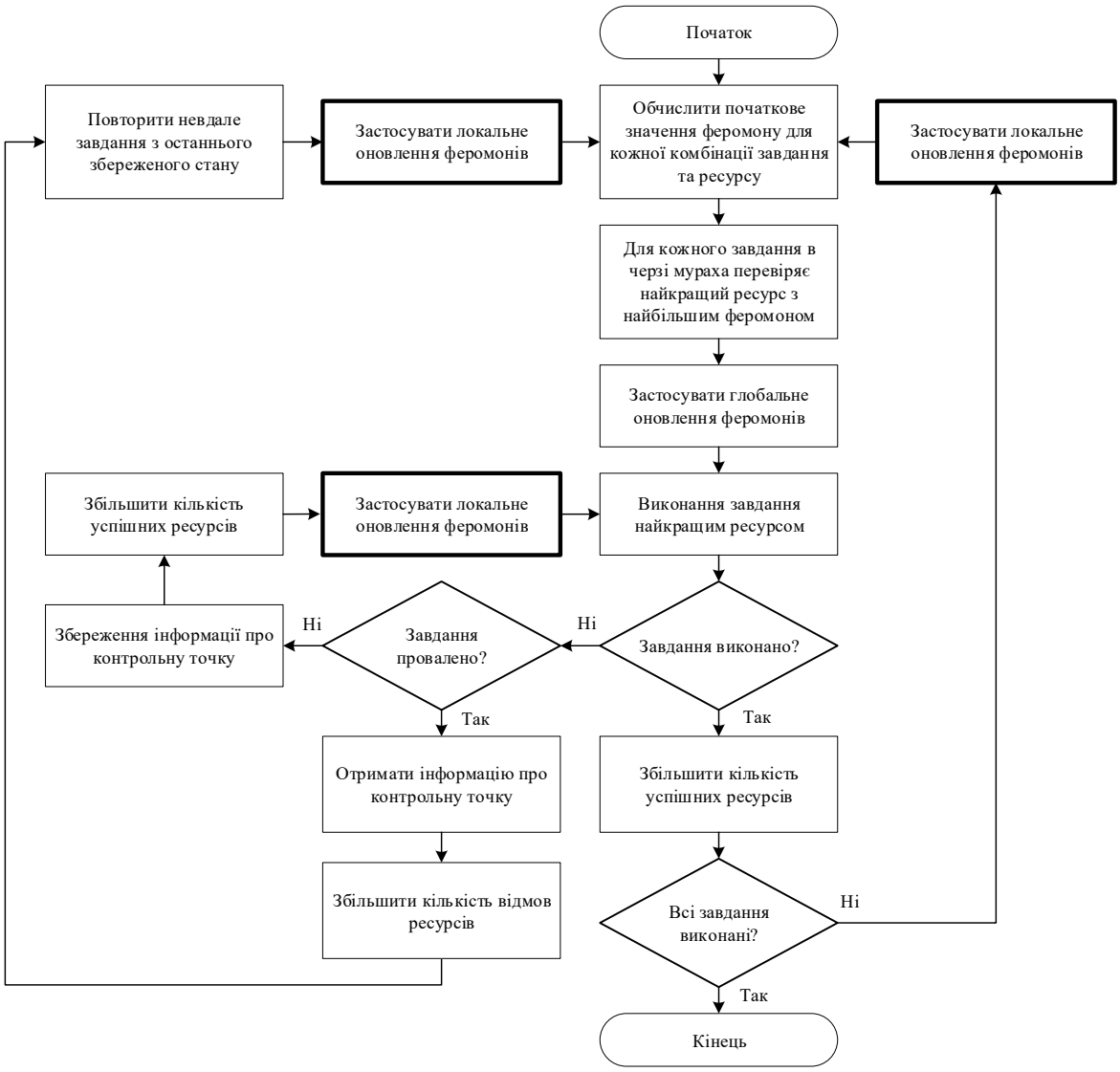


Рис. 3.3 – Модифікований алгоритм EDAF

Для визначення продуктивності запропонованого алгоритму можна застосовувати різні метрики, які включають час виконання, пропускну здатність, час виконання, затримку, балансування навантаження і коефіцієнт успішності.

Для перевірки ефективності запропонованого алгоритму EDAFT при виникненні відмов використовується псевдовипадковий алгоритм для призначення придатності ресурсів у межах конкретного діапазону. У цьому

випадку діапазон придатності ресурсу встановлюється між 50% і 100%. Важливо визначити діапазон придатності ресурсів для оцінки ефективності алгоритму відмовостійкості при різних сценаріях відмов. З іншого боку, інші ресурси і параметри завдань наведені в таблиці 3.1.

Таблиця 3.1 – Параметри моделювання

Параметр	Значення
Кількість ресурсів	200
Кількість завдань	8000
Рейтинг PE	80 MIPS
Пропускна здатність	8000 B/S
Машина на ресурс	1
PE на машину	2
Довжина завдання	80000
Розмір файлу	200 + (10-40%)
Розмір виводу	450 + (10-50%)

Алгоритм, який ми пропонуємо, був порівняний з алгоритмами TACO [16], FTACO [18], ACOwFT і ACO [19], які всі були адаптовані на основі опублікованих псевдокодів, формулювань і блок-схем. Вони були обрані для порівняння через їхню придатність до використання при оцінці, оскільки алгоритм EDAFT взятий за основу всіма цими алгоритмами з точки зору методів відмовостійкості. Всі експерименти моделювалися в імітаторі, який базується на мові JAVA і відомий як GridSim Toolkit, оскільки він забезпечує комплексне середовище для імітації сіток, включаючи більшість компонентів. Кожен алгоритм виконувався 10 разів для кожного діапазону несправностей, і було взяте середнє значення для більш точного вимірювання.

У переліку показників продуктивності враховувалися час виконання, пропускна здатність, середній час виконання і затримка завдань, а також балансування навантаження і коефіцієнт успішності виконання.

При відсутності помилок (рис. 3.4) час виконання для всіх алгоритмів практично однаковий. Однак при збільшенні частоти відмов час виконання для EDAFT виявляється найменшим порівняно з іншими, супроводжуючись ACOwFT і FTACO відповідно. Результати також свідчать, що ACO і TACO

виявляють значно більший час виконання, оскільки використання методу контрольних точок дозволяє істотно скоротити час виконання, уникаючи повторного виконання невдалого завдання з самого початку. Цей підхід також виявився ефективним, особливо при великому розмірі кожного завдання, і коли очікуваний час для повного виконання кожного завдання є значним.

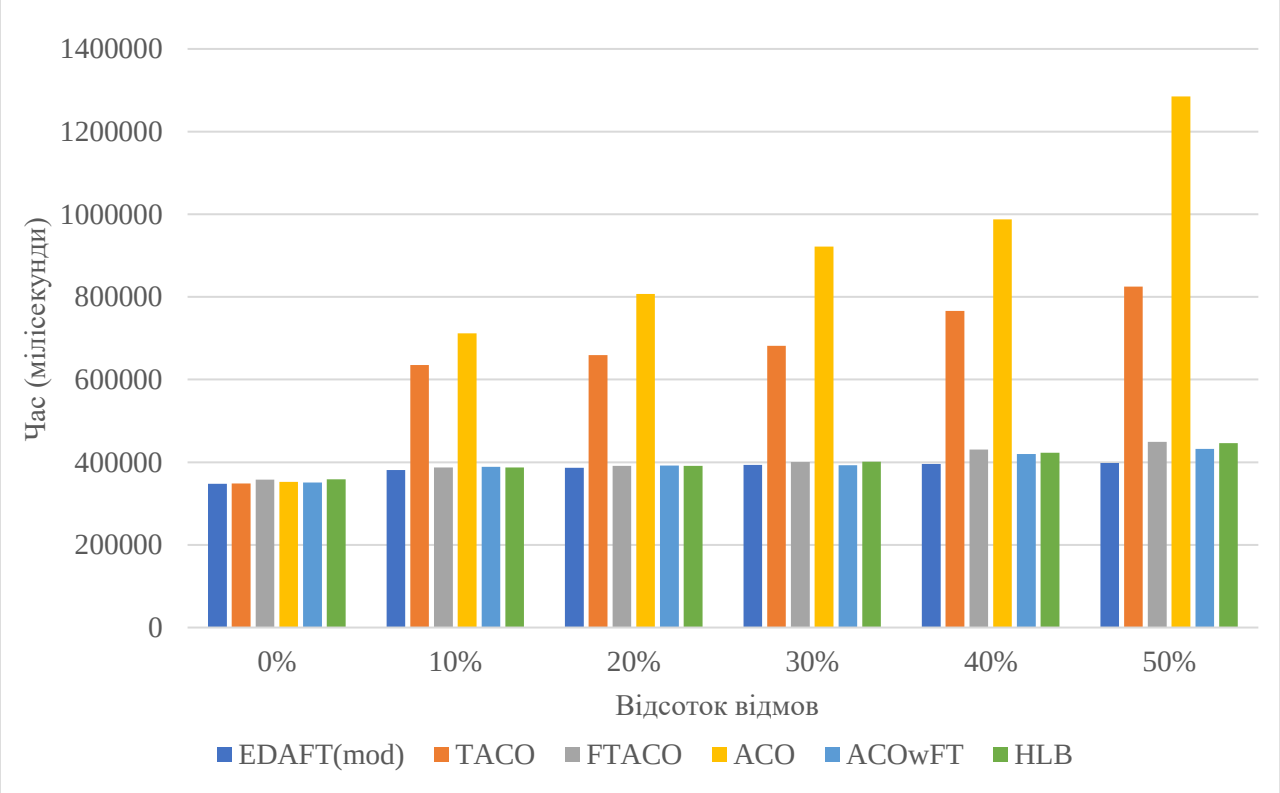
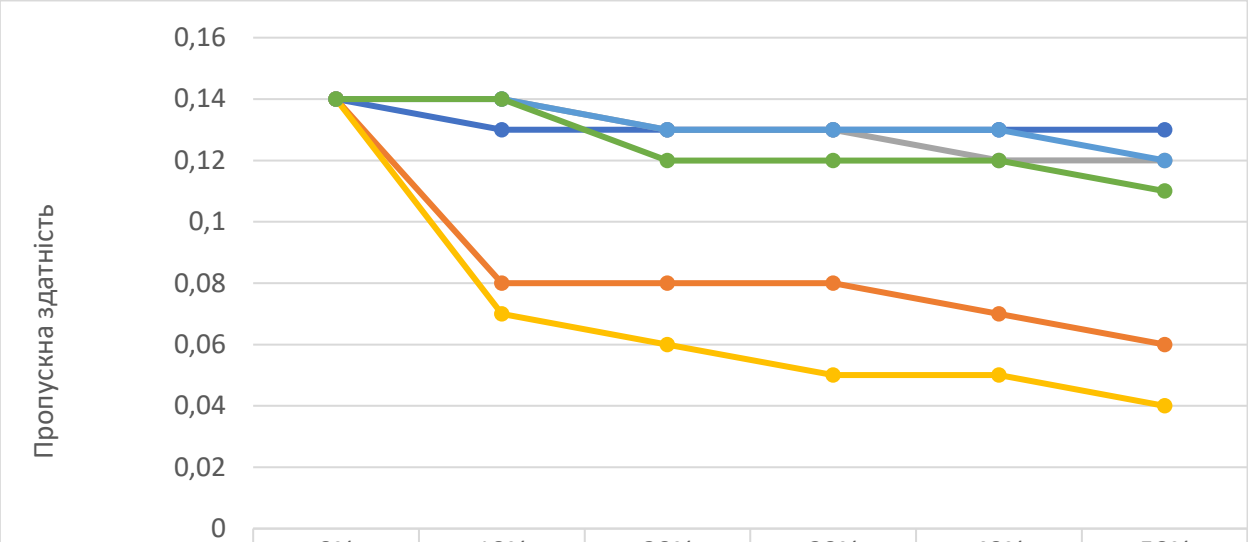


Рис. 3.4 – Результати часу виконання

Графік на рис. 3.5 відображає результати щодо пропускної здатності для всіх розглянутих алгоритмів. Пропускна здатність в основному визначається часом виконання і загальним обсягом виконаної роботи. У цьому випадку спостерігається відповідність між пропускною здатністю та часом виконання, який представлений на рис. 3.4. При частоті відмов 0%, пропускна здатність для всіх алгоритмів максимальна і поступово зменшується при збільшенні частоти відмов. Результати також свідчать, що EDAFT виявляє найменше зниження пропускної здатності від 10% до 50% частоти відмов порівняно з іншими алгоритмами. Збереження часу виконання важливо навіть у випадку

відмови, щоб забезпечити збереження пропускної здатності. На рисунку 3.3 видно, що середні часи виконання для EDAFT, FTACO і ACOwFT практично ідентичні у порівнянні з TACO і ACO. В даному випадку важливу роль відіграє метод контрольних точок, дозволяючи виконувати кожне невиконане завдання з останнього збереженого стану, що в кінцевому підсумку зменшує час виконання кожного окремого завдання. Результати також підтверджують, що EDAFT, FTACO і ACOwFT ефективно контролюють час виконання за допомогою техніки контрольних точок при виявленні ситуацій відмов, що забезпечує своєчасне завершення кожної індивідуальної задачі.

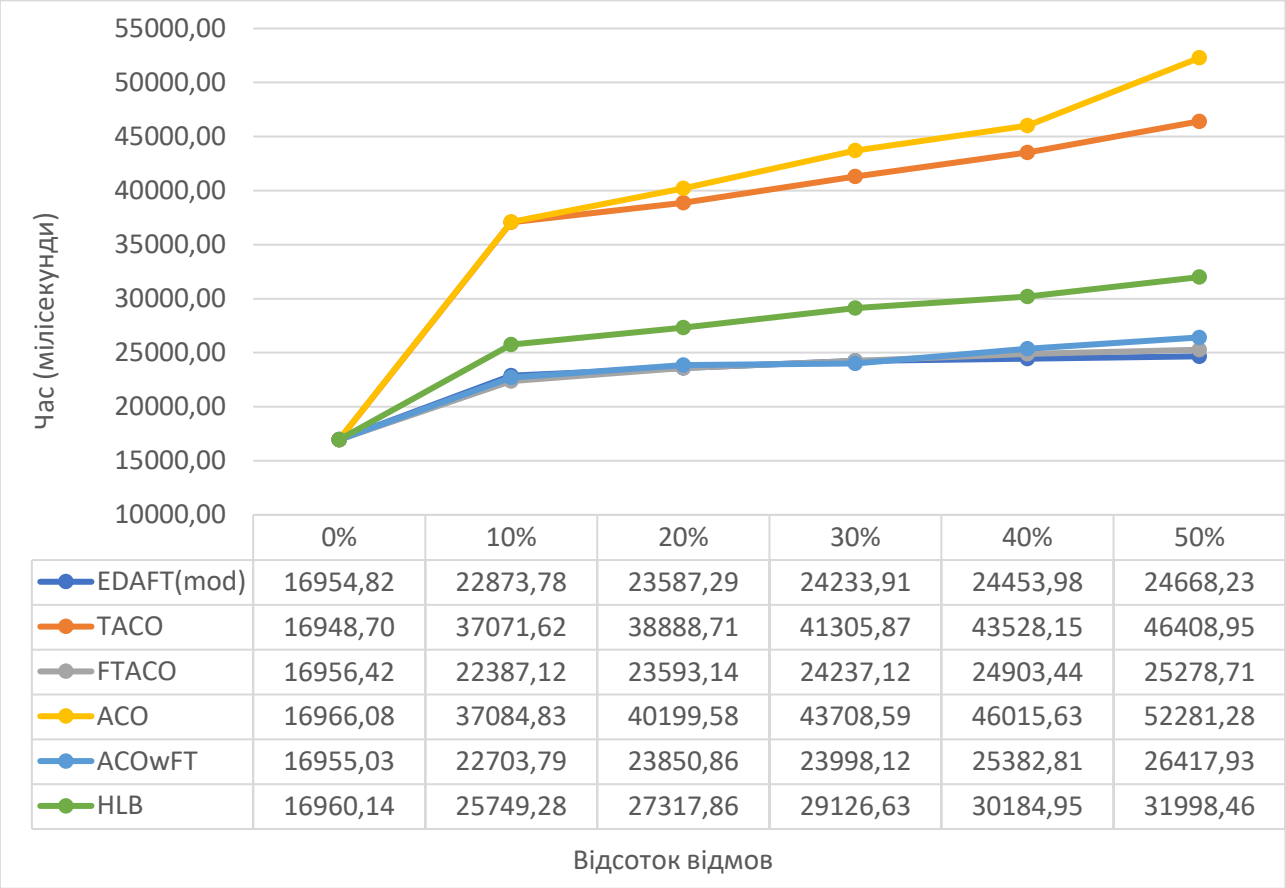


	0%	10%	20%	30%	40%	50%
EDAFT(mod)	0,14	0,13	0,13	0,13	0,13	0,13
TACO	0,14	0,08	0,08	0,08	0,07	0,06
FTACO	0,14	0,14	0,13	0,13	0,12	0,12
ACO	0,14	0,07	0,06	0,05	0,05	0,04
ACOwFT	0,14	0,14	0,13	0,13	0,13	0,12
HLB	0,14	0,14	0,12	0,12	0,12	0,11

Рис. 3.5 – Результати пропускної здатності

Середня затримка, представлена на рис. 3.7, демонструє аналогічну динаміку, що й середній час обороту на рис. 3.6. Отримані результати дають підстави вважати, що ACO проявляє найвищі значення затримки, і йому слідує TACO, де обидва алгоритми не використовують техніку контрольних точок.

Ефективне керування затримкою можливе шляхом правильного розподілу завдань між усіма доступними ресурсами, уникаючи ситуацій перевантаження, коли черги на окремих ресурсах перевищують середню довжину черги.



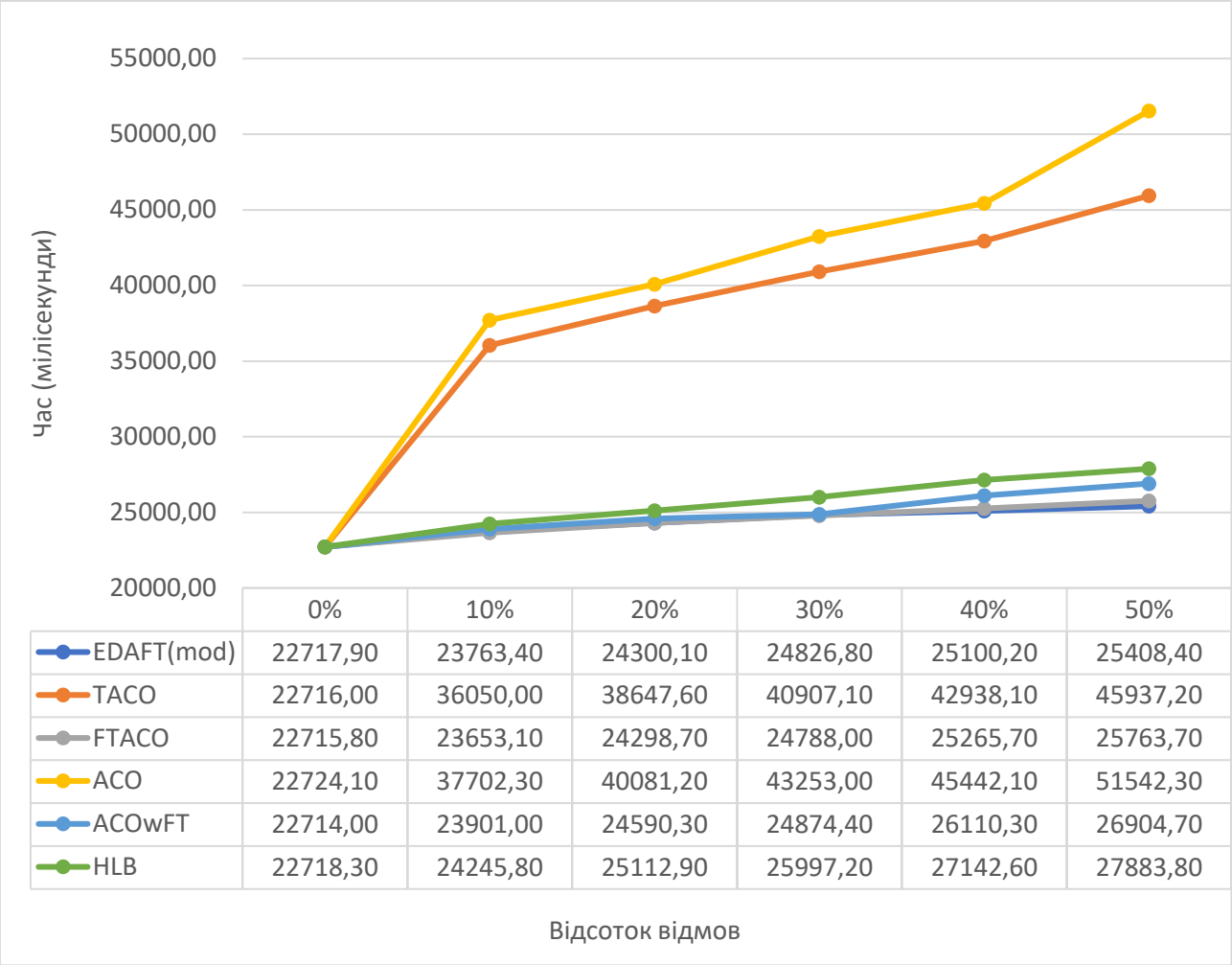
	0%	10%	20%	30%	40%	50%
EDAFT(mod)	16954,82	22873,78	23587,29	24233,91	24453,98	24668,23
TACO	16948,70	37071,62	38888,71	41305,87	43528,15	46408,95
FTACO	16956,42	22387,12	23593,14	24237,12	24903,44	25278,71
ACO	16966,08	37084,83	40199,58	43708,59	46015,63	52281,28
ACOwFT	16955,03	22703,79	23850,86	23998,12	25382,81	26417,93
HLB	16960,14	25749,28	27317,86	29126,63	30184,95	31998,46

Відсоток відмов

Рис 3.6 – Результати середнього часу виконання завдання

Баланс навантаження визначає, наскільки ефективно розподілені завдання. Як показано на рис. 3.8, ACOwFT та ACO демонструють найменше стандартне відхилення, оскільки кожне завдання призначається враховуючи поточне навантаження кожного ресурсу. Щодо EDAFT, яка практично вирівнюється з ACOwFT за продуктивністю, завдання виконуються з урахуванням історії виконання при визначенні придатності ресурсу та балансуванні навантаження. Чим ближче стандартне відхилення до нуля, тим ефективніше балансування навантаження, тобто запропонований алгоритм виявляється здатним застосовувати евристичні підходи до визначення

придатності на основі історії виконання, що сприяє урівноваженому використанню ресурсів, навіть без прямого знання про придатність конкретного ресурсу.



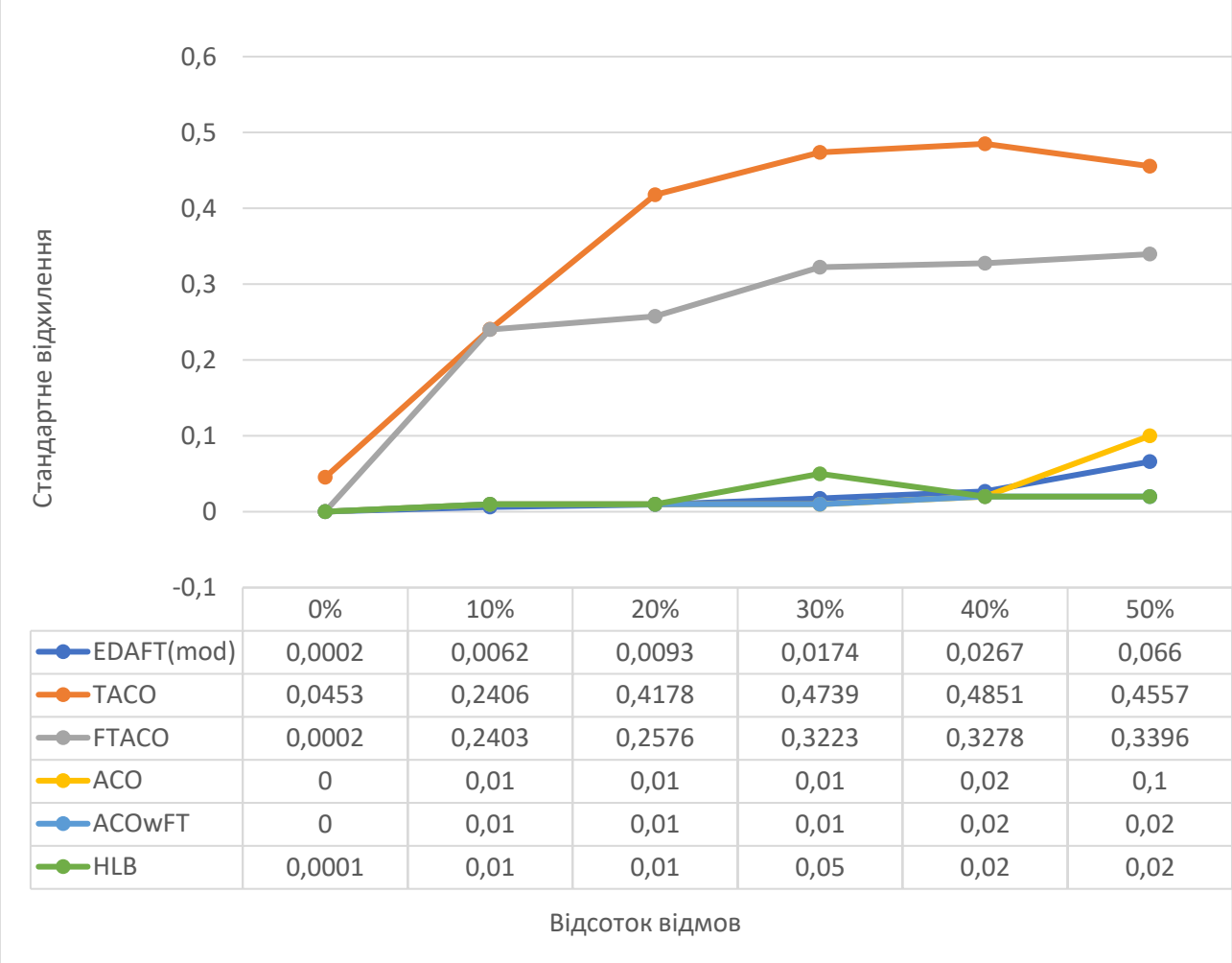
	0%	10%	20%	30%	40%	50%
EDAFT(mod)	22717,90	23763,40	24300,10	24826,80	25100,20	25408,40
TACO	22716,00	36050,00	38647,60	40907,10	42938,10	45937,20
FTACO	22715,80	23653,10	24298,70	24788,00	25265,70	25763,70
ACO	22724,10	37702,30	40081,20	43253,00	45442,10	51542,30
ACOwFT	22714,00	23901,00	24590,30	24874,40	26110,30	26904,70
HLB	22718,30	24245,80	25112,90	25997,20	27142,60	27883,80

Рис 3.7 – Результати середньої затримки на завдання

Додатково, фактор довіри є корисним для винагороди придатних ресурсів або за штрафи за непридатні ресурси на основі виконання завдання. Поєднання обох елементів забезпечує більш ефективний процес оновлення феромонів, що сприяє вирішенню питань балансування навантаження та успішності виконання.

У системі, яка має високу ступінь стійкості до відмов, головною метою є підтримка рівня успішності виконання без втрат продуктивності. Згідно з рис. 3.9, EDAFT проявляє вищий рівень успіху порівняно з іншими

алгоритмами. Це може здатися дивним, але TACO займає друге місце за кількістю успішних спроб, оскільки призначає більшу частину завдань відповідно до наявних ресурсів, а не з непридатними. Очевидно, що кожен раз, коли більшість завдань розподіляється на найбільш підходящі ресурси, це збільшує ймовірність успішного виконання.

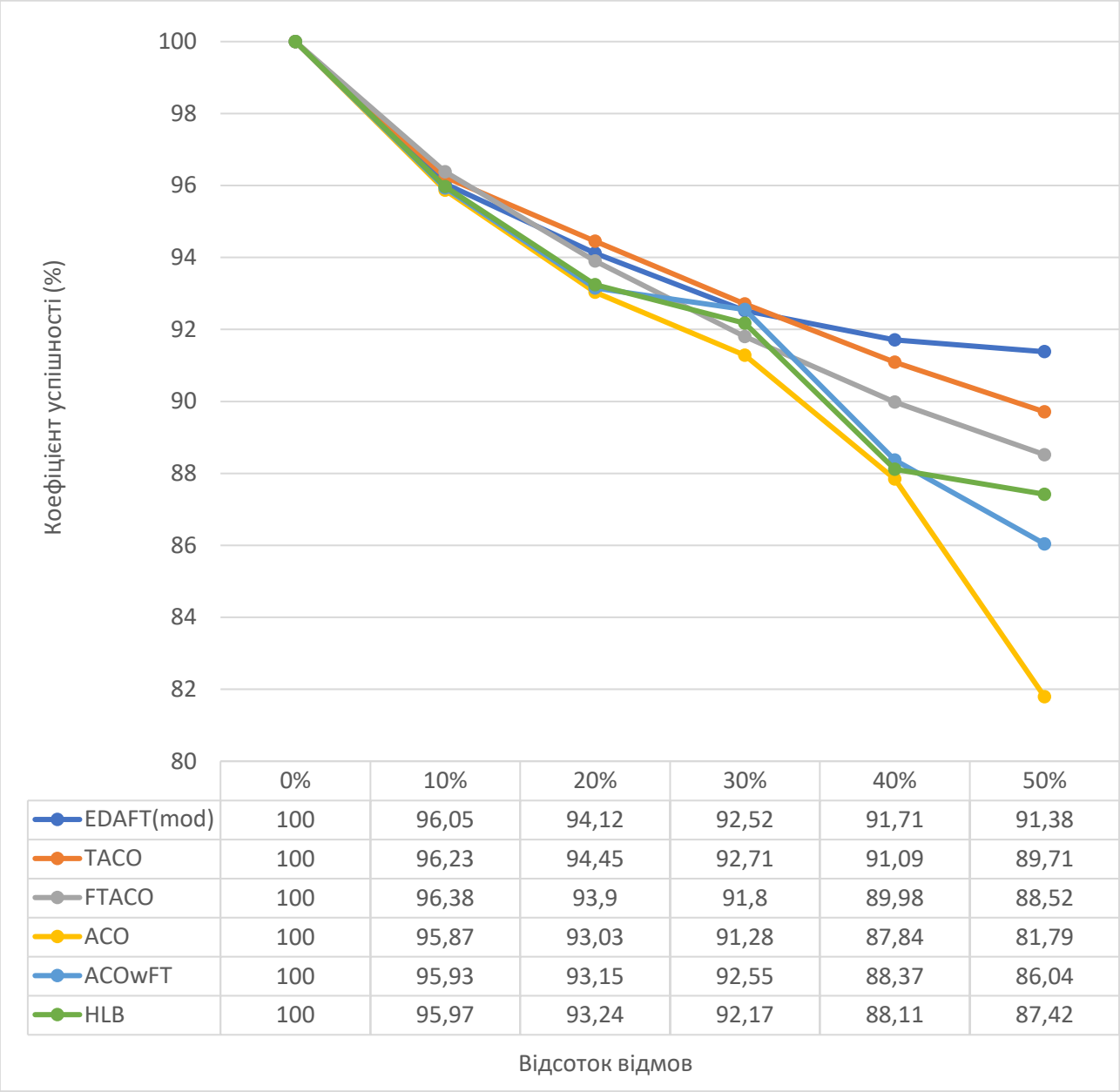


	0%	10%	20%	30%	40%	50%
EDAFT(mod)	0,0002	0,0062	0,0093	0,0174	0,0267	0,066
TACO	0,0453	0,2406	0,4178	0,4739	0,4851	0,4557
FTACO	0,0002	0,2403	0,2576	0,3223	0,3278	0,3396
ACO	0	0,01	0,01	0,01	0,02	0,1
ACOwFT	0	0,01	0,01	0,01	0,02	0,02
HLB	0,0001	0,01	0,01	0,05	0,02	0,02

Відсоток відмов

Рис 3.8 – Результати балансування навантаження

Недолік TACO полягає в тому, що затримка і тривалість виконання будуть значно вищими, оскільки відповідні ресурси матимуть довші черги. Крім того, ACOwFT і ACO мають найнижчий показник успіху, оскільки лише посилання на навантаження недостатнє для визначення придатності ресурсу. Деякі ресурси можуть мати низьке навантаження через відсутність активних завдань.



	0%	10%	20%	30%	40%	50%
EDAFT(mod)	100	96,05	94,12	92,52	91,71	91,38
TACO	100	96,23	94,45	92,71	91,09	89,71
FTACO	100	96,38	93,9	91,8	89,98	88,52
ACO	100	95,87	93,03	91,28	87,84	81,79
ACOwFT	100	95,93	93,15	92,55	88,37	86,04
HLB	100	95,97	93,24	92,17	88,11	87,42

Рис. 3.9 – Результати успішності

Проте для EDAFT властива здатність підтримувати частоту успіху, забезпечуючи також високу пропускну здатність, краще балансування навантаження та менші затримки та час виконання. Також виявлено, що зниження значення довіри C у разі відмови може збільшити рівень успіху для EDAFT, але при цьому може збільшити стандартне відхилення балансування навантаження. Таким чином, важливо визначити оптимальне значення довіри

для досягнення найкращого результату виконання або оптимального балансування навантаження.

3.5 Висновок до третього розділу

Симуляція, описана в даному розділі, надає можливість порівняти базовий алгоритм балансування навантаження, а саме Ant Colony System (ACS), із розробленою модифікацією та іншими розглянутими алгоритмами.

Після проведеного порівняння було встановлено, що розроблений алгоритм EDAFT виявився більш продуктивним загалом порівняно з іншими алгоритмами. Зокрема, він показав кращі результати щодо часу виконання (в середньому на 35%), пропускної здатності (на 38%), середнього часу виконання (на 26%), середньої затримки і швидкості успішного виконання. Проте, щодо розподілу навантаження ACOwFT продемонстрував кращі показники з невеликою різницею в порівнянні з ACO та EDAFT.

Навіть при високій продуктивності EDAFT є можливість його подальшого вдосконалення, додавши тимчасове припинення, яке уникне призначення завдань на недавно відмовлених ресурсах, поки вони не будуть відновлені після збою. Це може бути особливо ефективним, коли розмір окремого завдання значний, а більшість ресурсів зайняті обробкою великого завдання. Важливо забезпечити, щоб система не надавала завдання негайно на ресурсі, що нещодавно відмовився, щоб зменшити ймовірність подальших збоїв і вибирала ресурс, який може завершити поточне виконання швидше та має високу придатність.

ВИСНОВОК

Метою роботи є аналіз існуючих методів балансування навантаження та розробка ефективної методики балансування за результатами аналізу проблем існуючих підходів.

Для досягнення мети роботи було поставлено і вирішено такі завдання:

- визначиено поняття та основні методи балансування навантаження в комп’ютерних мережах;

- визначено основні проблеми існуючих методів балансування навантаження;

- досліджені принципи програмно-конфігурованих мереж;

- досліджені методи балансування навантаження, які базуються на алгоритмі ACS;

- запропонувані та досліджені ефективність модифікації алгоритму ACS для покращення показників якості обробки завдань.

Балансування навантаження в розподілених системах – це процес розподілу робочого навантаження системи між обчислювальними ресурсами, що розділяються. З появою високошвидкісних каналів зв’язку все частіше в якості вузлів розподілених систем використовуються автономні комп’ютерів, об’єднаних високошвидкісними лініями зв’язку. Основною перевагою таких систем є висока продуктивність та доступність при низьких витратах.

Внаслідок цього розподілені обчислення набувають все більшого значення як переважний метод обчислень порівняно з централізованою обробкою даних.

Сьогодні існує безліч підходів до вирішення проблеми балансування навантаження у розподілених системах, зокрема розподілені файлові системи не є винятком. Ці підходи можуть використовуватися для покращення якості обслуговування клієнтів, підвищення продуктивності системи, а також більш раціонального використання ресурсів. Як переваги ефективного балансування

навантаження варто відзначити підтримку відмовостійкості та масштабованості системи.

Сучасна система веб-серверів повинна завжди використовувати один або кілька методів ефективного балансування вхідного навантаження для розподілу між доступними їй веб-ресурсами. Ці ресурси з кожним днем стають все дорожчими, тому необхідні ефективні механізми оптимізації витрат, особливо в рамках підтримки такої системи в невеликій організації.

В роботі було виявлено кілька проблем з балансуванням навантаження для ефективного використання веб-ресурсів у розподіленому середовищі та проведено огляд та аналіз цілого ряду сучасних алгоритмів та підходів до вирішення цих проблем. Був проаналізований детальний опис цих підходів, їх сильних та слабких сторін, перспектив застосування та обмежень, які вони накладають.

На основі вищезгаданих проблем балансування навантаження та виявлених недоліків сучасних алгоритмів балансування навантаження, було визначено кілька майбутніх вимірів, які будуть корисними для дослідницької спільноти для досягнення різних цілей: розробка моделі розподілу ресурсів, яка враховує характеристики як ресурсу, так і конкретного завдання для оптимізації різних показників якості обслуговування; розробка моделі збалансованості навантаження з відмовою для частково виконаних завдань через збій ресурсів та побудова політики вибору ресурсів для виконання завдань; аналіз контекстуальних відносин між проблемами CloudIoT та оптимізація їх за допомогою ефективного планування; розробка моделі прогнозування часу виконання для ефективного забезпечення ресурсами, вибору та планування.

Описана в роботі симуляція надає можливість порівняти базовий алгоритм балансування навантаження, а саме Ant Colony System (ACS), із розробленою модифікацією та іншими розглянутими алгоритмами. Після проведеного порівняння було встановлено, що розроблений алгоритм EDAFT виявився більш продуктивним загалом порівняно з іншими алгоритмами.

Зокрема, він показав кращі результати щодо часу виконання (в середньому на 35%), пропускної здатності (на 38%), середнього часу виконання (на 26%), середньої затримки і швидкості успішного виконання. Проте, щодо розподілу навантаження ACOwFT продемонстрував кращі показники з невеликою різницею в порівнянні з ACO та EDAFT.

Навіть при високій продуктивності EDAFT є можливість його подальшого вдосконалення, додавши тимчасове припинення, яке уникне призначення завдань на недавно відмовлених ресурсах, поки вони не будуть відновлені після збою. Це може бути особливо ефективним, коли розмір окремого завдання значний, а більшість ресурсів зайняті обробкою великого завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глоба Л. С. Розподілені системи та мережі. Том 1: «Розподілені системи», «Розподілені системи. Поняття розподіленого середовища», «Зв'язок», «Процеси», «Іменування», «Синхронизація». Київ : НТУУ «КПІ», 2011. URL : https://moodle.znu.edu.ua/pluginfile.php/486881/mod_resource/content/1/Глоба_книга_Том1-21-31.pdf.
2. Мережева модель OSI для чайників. DIGITAL-ЛАБОРАТОРІЯ IT ТЕХНОЛОГІЙ. URL : https://kr-labs.com.ua/blog/model-osi.
3. Іванісенко І. М. Методи балансування навантаження у розподілених системах з урахуванням самоподібних властивостей вхідних потоків : дис. … канд. техн. наук : 05.13.05 / Харківський національний університет радіоелектроніки. Харків, 2017. 184 с. URL : https://nure.ua/wp-content/uploads/2018/Dissertation/dissertatsiya_Ivanisenko_ukr_.pdf.
4. Іванісенко І.М. Балансування навантаження з урахуванням рівня дисбалансу системи. *Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління* : матеріали V міжнар. наук.-техн. конф., 21–22 квіт. 2016 р. Полтава-Баку-Кіровоград-Харків, Україна, 2016. С. 32.
5. Игорь Савчук. Балансировка нагрузки сервера по методу SLB. 2012. URL: http://blogerator.ru/page/high-load-balansirovka-nagruzki-servera-pometodu-sticky-load-balancing.
6. Тарасов В. Н., Полежаев П. Н., Шухман А. Е. Математические модели облачного вычислительного центра обработки данных с использованием Openflow. *Вестник ОГУ*. 2012. № 9 (145). С. 150–155.
7. Чжоу Тао. Системы балансировки нагрузки Web-серверов. *Windows IT Pro*. 2000. № 03. URL : http://www.osp.ru/win2000/2000/03/174228/.
8. Радивилова Т. А., Иванисенко И. Н. Динамический метод оценки загрузки узлов распределенной системы. *Проблеми телекомунікацій*. 2016. № 1

(18). С. 42–51. URL : http://pt.journal.kh.ua/2016/1/1/161_radivilova_utilization.pdf.

9. Іванісенко І. М. OPEN SOURCE продукти балансування навантаження. *Free and Open Source Software* : матеріали VII Всеукр. наук.-практ. конф., 24–27 лист. 2015 р. Харків, Україна, 2015. С 85.
10. Kirichenko Lyudmila, Ivanisenko Igor, Radivilova Tamara. Dynamic load balancing algorithm of distributed systems. *Modern Problems of Rradio Engineering, Telecommunications and Computer Science* : Abstract Book of XIIIth Intern. Conf. TCSET'2016, February 23–26, 2016. Lviv-Slavsko, Ukraine, 2016. P. 515–518.
11. Тарнавський Ю. А., Кузьменко І. М. *Організація комп'ютерних мереж* : підручник для студ. спеціальності 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки». Київ : КПІ ім. Ігоря Сікорського, 2018. 259 с. URL : https://ela.kpi.ua/bitstream/123456789/25156/1/Tarnavsky_Kuzmenko_Org_Komp_merej.pdf.
12. Карпенко М. Ю., Макогон Н. В. Комп'ютерні мережі. Харків : Харківський національний університет міського господарства імені О. М. Бекетова. URL : https://eprints.kname.edu.ua/52081/1/2017%20%D1%80%D0%B5%D0%BF%20249%D0%9B%20%D0%BB%D0%BA%D0%9A%D0%BE%D0%BC%D0%BF%D0%A1%D0%B5%D1%82%D0%B8.pdf.
13. Організація комп'ютерних мереж. Топологія комп'ютерних мереж. URL : https://kremenetskyy.blogspot.com/2017/10/blog-post.html.
14. Абрамов В. О. *Комп'ютерні мережі* : навчальний посібник. Київ : Київ. ун-т ім. Б. Грінченка, 2010. 108 с. URL : https://elibrary.kubg.edu.ua/id/eprint/8382/1/V_Abramov_V_Chehrenets_KM_NP_2010_IS.pdf.
15. Балансування навантаження та мікропослуги. URL : https://hashdork.com/uk/load-balancing-microservices.
16. Демчик В. В. Аналіз сучасних методів планування обчислень та балансування навантаження в розподілених комп'ютерних системах. *The scientific heritage*. 2021. № 72. С. 30–39.

17. Иванисенко И. Н., Радивилова Т. А. Анализ методов балансировки нагрузки в распределённых системах. *Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління* : матеріали V міжнар. наук.-техн. конф., 23–25 квіт. 2015 р. Полтава-Баку-Кіровоград-Харків, Україна, 2015. С. 20–21. URL : http://www.foibg.com/ijitk/ijitkvol09/ijitk09-04-p04.pdf.
18. Бабенко О. А., Гіоргізова-Гай В. Ш. Балансування навантаження «хмарних» додатків. *Системний аналіз та інформаційні технології SAIT 2015* : матеріали 17-ї Міжнародної науково-технічної конференції, 22–25 червня 2015 р. Київ, 2015. С. 184. URL : http://cad.kpi.ua/_attachments/043_2017_03_gg.pdf.
19. Боровський Б. М., Явіся В. С. Методи підвищення ефективності алгоритмів та методів балансування навантаження у хмарних середовищах інфокомунікаційних систем : магістерська дисертаційнаа робота зі спец. 172 «Телекомунікації та радіотехніка». Київ : НТУУ «КПІ ім. Ігоря Сікорського», 2020. 124 с. URL : https://ela.kpi.ua/bitstream/123456789/58096/1/Borovskyi_magistr.pdf.
20. Симаков Д. В. Управление трафиком в сети с высокой динамикой метрик сетевых маршрутов. *Интернет-журнал «НАУКОВЕДЕНИЕ»*. Том 8, № 1. 2016.
21. Телелейко І. С., Клятченко Я. М. Інтеграція сервіс-орієнтованої архітектури з Grid-системами. *Прикладна математика та комп'ютинг (ПМК-2016)* : матеріали VIII наук. конф. магістрантів та аспірантів, 20–22 квітня 2016 р. Київ : Просвіта, 2016. С. 60-64.
22. Телелейко І. С., Орлова М. М. Сумісність підходів хмарних обчислень та грід-технологій. *Системний аналіз та інформаційні технології» SAIT 2017* : матеріали XIX-а міжн. наук.-техн. конференції, 22–25 травня 2017 р. Київ : ННК «ІПСА» НТУУ «КПІ ім. Ігоря Сікорського», 2017. 340 с.
23. Телелейко І. С., Орлова М. М. Аналіз методів динамічного балансування навантаження в хмарному середовищі. *Прикладна математика та*

комп'ютинг : матеріали X конференції молодих вчених, 21–23 березня 2018 р. Київ : Просвіта, 2018.

24. Телелейко І. С., Орлова М. М. Спосіб динамічного балансування навантаження в хмарному середовищі. *Науковий журнал «Інтернаука»*. 2018. № 7. URL : https://www.inter-nauka.com/issues/2018/7/3687/.
25. Микитишин А. Г., Митник М. М., Стухляк. П. Д., Пасічник В. В. Комп'ютерні мережі. Львів, 2013. 373 с.
26. Порєв Г. В. Архітектура комп'ютерних мереж : методичний посібник. Вінниця : Універсум Вінниця, 2008. 98 с.
27. *Інформаційні технології і автоматизація*: матеріали XIV міжн. науково-практ. конф., 21–22 жовтя 2021 р. Одеса : Вид-во ОНАХТ, 2021. 350 с.
28. Журавська І. Гетерогенні комп'ютерні мережі критичного застосування на основі роїв та зграй БПЛА : монографія. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2019. 192 с.
29. Горбулін В. П., Додонов О. Г., Горбачик О. С., Кузнєцова М. Г. Розподілені комп'ютерні системи як складові інформаційних інфраструктур. *Реєстрація, зберігання і обробка даних*. 2008. Т. 10, № 4. С. 19–24.
30. Балансировка нагрузки. URL : https://ru.wikipedia.org/wiki/.
31. Algorithms for making load-balancing decisions. URL : https://www.ibm.com/support/knowledgecenter/SS9H2Y7.6.0/com.ibm.dp.doc/lbg_algorithms.html.
32. Comparing Load Balancing Algorithms. URL : Режим доступа: https://www.jscape.com/blog/load-balancing-algorithms.
33. Алгоритмы и методы балансировки нагрузки. URL : https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques.
34. Задача балансировки нагрузки на серверы. URL : https://cyberleninka.ru/article/n/zadacha-balansirovki-nagruzki-na-servery/viewer.
35. Дворников В.С., Долгов В.В., Венцов Н.Н. Обзор методов балансировки нагрузки в гетерогенных распределенных файловых системах. *Фундаментальные исследования*. 2017. № 9-2. С. 295-302

36. А. Г. Микитишин., М. М. Митник., П. Д. Стухляк.В., В. В. Пасічник. Комп'ютерні мережі : навчальний посібник. Львів, 2013. 373 с.
37. Буров Є. В. Комп'ютерні мережі : підручник. Львів, 2010. 262 с.
38. Городецька, О. С., Гикавий В. А., Онищук О. В. Комп'ютерні мережі : навчальний посібник. Вінниця : ВНТУ, 2017. 129 с.
39. Сучасні технології побудови комп'ютерних мереж. 2013. URL : http://m.programming.in.ua/other-files/internet/234-technology-for-creting-computer-network.
40. Климаш М. М., Шпур О. М., Селюченко М. О., Киричук Б. В., Мельник Т. В. Метод підвищення ефективності використання мережевих ресурсів інформаційно-телекомунікаційних систем. *Вісник Національного університету «Львівська політехніка» «Радіоелектроніка та телекомунікації»*. Львів, 2015. № 818. С. 137–151.
41. Бешлей М. І., Селюченко О. М., Лаврів О. А., Масюк А. Р., Холявка Г. В. Оцінка адекватності функціонування програмного маршрутизатора у процесі обслуговування мультимедійного трафіку. *Вісник Національного університету «Львівська політехніка» «Радіоелектроніка та телекомунікації»*. Львів, 2015. № 818. С. 162–173.
42. Жураковський Б. Ю., Зенів І. О. Комп'ютерні мережі. Частина 1 : навч. посіб. для студ. спец. 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології», спеціалізації «Інженерія програмного забезпечення інформаційно управляючих систем» та «Інформаційне забезпечення робототехнічних систем». Київ : КПІ ім.Ігоря Сікорського, 2020. 336 с.
43. Гаркуша І.М. Конспект лекцій з дисципліни «Комп'ютерні мережі» для студентів галузі знань 12 «Інформаційні технології» спеціальності 126 «Інформаційні системи та технології». Дніпро : НТУ «ДП», 2019. 75 с.
44. Оптимизация методом колонии муравьев. Алгоритм ACOR. URL : https://habr.com/ru/articles/163887/.