Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях

Виконав: студент 2-го курсу
групи 2ПІ-22м_ спеціальності
121 – Інженерія програмного забезпечення
Сиченко Віктор Володимирович

Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
«12» грудня 2023 р.

Опонент: к.т.н., доц. каф. ЗІ Лукічов В.В.
«12» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«12» грудня 2023 р.

Вінниця ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« _19_ » вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сиченко Віктору Володимировичу

1. Тема роботи – розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях.

Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 19 » вересня 2023 р. № 247.

2. Строк подання студентом роботи
_____5 грудня 2023 р._____

3. Вихідні дані до роботи: формалізація методів підвищення ефективності обчислювальних систем, визначення продуктивності обчислювальних систем у розподілених системах - не менш 500 запитів за секунду, визначення розміру обробки запитах на обчислювальних системах - не більш ніж 300 мс, визначення кількості відмов в обробці запитів на сервері - не більш ніж 100 відмов за секунду, визначення кількості коллізій на мережевому інтерфейсі серверу - не більш ніж 2 за 5 хвилин, визначення кількості віртуальних машин на одному серверу - не більш ніж 20.

4. Зміст розрахунково-пояснювальної записки: вступ,аналіз сучасних методів та розробки програмних засобів досліджень bottlenecks у високопродуктивних обчисленнях, задачі розробки методів та програмних засобів дослідження, аналіз сучасних методів розробки програмних засобів для дослідження bottlenecks, вимоги щодо програмного засобу дослідження bottlenecks, структурно-функціональні особливості основних модулів

програмного засобу, вхідні та вихідні параметри програмного засобу, алгорит
обчислення вхідних параметрів для обробки даних у програмному засобі, ме
вирішення задачі, висновки, економічна частина.

5. Перелік графічного матеріалу: вступ, актуальність теми, аналіз сучас
методів та розробки програмних засобів досліджень bottlenecks
високопродуктивних обчисленнях, задачі розробки методів та програм
засобів дослідження, аналіз сучасних методів розробки програмних засобів
дослідження bottlenecks, вимоги щодо програмного засобу дослідже
bottlenecks, вхідні та вихідні параметри програмного засобу, алгорит
обчислення вхідних параметрів для обробки даних у програмному засобі, ме
вирішення задачі, економічний розділ, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О. М., к.т.н., доцент кафедри ПЗ	19.09.23	01.12.23
5	Кавецький В.В., к.е.н., доц. кафедри ЕПВМ	13.11.23	01.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних методів та розробки програмних засобів досліджень bottlenecks у високопродуктивних обчисленнях.	19.09.2023-02.10.2023	вик
2	Задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях.	03.10.2023-16.10.2023	вик
3	Визначення вимог щодо програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.	17.10.2023-28.10.2023	вик
4	Розробка вхідних та вихідних параметрів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.	29.10.2023-12.11.2023	вик
5	Економічна частина	13.11.2023-01.12.2023	вик

Студент _____ **Сиченко В. В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Хошаба О. М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 519.6.

Сиченко В.В. Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях: магістерська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 130 с.

На укр. мові. Бібліогр. : 34 назв ; рис. : 8; табл. 20.

В магістерській роботі доведена необхідність використання сучасних методів та розробки програмних засобів досліджень bottlenecks у високопродуктивних обчисленнях. Змістовно описані задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях. Виконано аналіз сучасних методів розробки програмних засобів для дослідження bottlenecks. Проведено дослідження відносно особливості використання методів дослідження bottlenecks у високопродуктивних обчисленнях. Визначені вимоги щодо програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях. Запропоновані структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.

Також, реалізовано метод та програмний засіб, що дозволяє проводити дослідження bottlenecks у високопродуктивних обчисленнях. Розроблені вхідні та вихідні параметри програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях, створені алгоритми обчислення вхідних параметрів для обробки даних у програмному засобі.

Запропонований метод вирішення задачі полягав у у використанні системи сповіщень про негативні наслідки розвитку bottlenecks, на основі отримання повідомлень про критичні точки.

Ключові слова: методи підвищення ефективності, використання обчислювальних ресурсів, bottlenecks, високопродуктивні обчислення.

ABSTRACT

Sychenko V.V. Development of methods and software tools for studying bottlenecks in high-performance computing. : master's qualification thesis on the specialty 121 Software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 130 with.

In Ukrainian language. Bibliographer : 34 titles; Fig. : 8; table 20.

The master's work proved the necessity of using modern methods and developing software tools to research high-performance computing bottlenecks. The tasks of developing methods and software tools for studying bottlenecks in high-performance computing are meaningfully described. An analysis of modern software development methods for researching bottlenecks was performed. Research has been conducted regarding bottleneck research methods in high-performance computing. Defined requirements for a software tool for researching bottlenecks in high-performance computing. They proposed structural and functional features of the main modules of the software tool for researching bottlenecks in high-performance computing.

Also, a method and a software tool have been implemented, allowing for researching bottlenecks in high-performance computing. The input and output parameters of the software tool for the study of bottlenecks in high-performance computing were developed, and the algorithms for calculating the input parameters for data processing in the software tool were created.

The proposed method of solving the problem consisted of a system of notifications about the negative consequences of the development of bottlenecks based on receiving messages about critical points.

Keywords: efficiency improvement methods, use of computing resources, bottlenecks, high-performance computing.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Дослідження bottlenecks у високопродуктивних обчисленнях має велику наукову важливість і актуальність. Так, у високопродуктивних обчисленнях, де обробка великих обсягів даних та складні обчислення є стандартом, то покращення продуктивності є критичним завданням [1]. При цьому, дослідження вузьких місць може допомогти ідентифікувати проблемні точки і розробляти оптимізації, які підвищують ефективність обчислень.

Також, забезпечення оптимального використання обчислювальних ресурсів, включаючи суперкомп'ютери та обчислювальні кластери, є важливим завданням. Дослідження вузьких місць може допомогти виявити, як ресурси розподіляються та як можна покращити їхнє використання.

З ростом технологічного прогресу і виникненням нових архітектур обчислювальних систем з'являються нові вузькі місця. Дослідження їх може допомогти адаптувати програмні рішення до нових технологій та визначити оптимальні стратегії для їх використання.

В багатьох наукових областях, включаючи фізику, біологію, кліматологію та інші галузі наук, високопродуктивні обчислення є ключовим інструментом. Збільшення продуктивності дозволяє вченим аналізувати дані та виконувати моделювання в більших масштабах і з великою точністю. Досить вагомі вдосконалення в галузі продуктивності обчислень також надаються для підприємств і промисловості. За допомогою високопродуктивних обчислень, бізнеси можуть оптимізувати процеси, проводити аналіз даних та розробляти нові продукти та послуги. Тому, у великих промислових системах, де надійність і продуктивність важливі, виявлення та усунення вузьких місць може допомогти попередити аварії та збої.

Таким чином, дослідження bottlenecks є ключовим елементом для підвищення ефективності обчислень та забезпечення стійкої роботи систем у сферах науки, індустрії та досліджень [1,2].

Мета та завдання дослідження:

Метою роботи є підвищення ефективності використання програмних засобів з метою зменшення наслідків bottlenecks у високопродуктивних обчисленнях.

У відповідності до поставленої мети потрібно виконати такі **завдання**:

- виконати обгрунтування методу аналізу дослідження bottlenecks у високопродуктивних обчисленнях та розглянути необхідність використання сучасних методів та інструментів розробки програмних засобів;

- розробити задачі розробки та виконано аналіз сучасних методів та програмних засобів дослідження bottlenecks;

- розглянути загальну характеристику та дослідити особливість використання методів дослідження bottlenecks;

- запропонований метод діагностики bottlenecks у програмному засобі;

- визначити вимоги щодо функціонування програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях;

- розробити структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks;

- виконати реалізацію розробленого методу у програмному засобі;

- розробити вхідні та вихідні параметри програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях;

- виконати обчислення вхідних параметрів для обробки даних у програмному засобі.

Об’єктом дослідження процеси появи та розвитку bottlenecks у високопродуктивних обчисленнях.

Предметом дослідження є методи та засоби підвищення ефективності використання програмних засобів з метою зменшення наслідків bottlenecks у високопродуктивних обчисленнях.

Методи дослідження. У процесі дослідження використовувались: теорія дерева рішень, системний аналіз процесу обробки даних, моделі надійності систем, методи прийняття рішень.

Наукова новизна одержаних результатів:

- вперше запропоновано метод використання системи сповіщень про негативні наслідки розвитку bottlenecks, на основі отримання повідомлень про критичні точки, що дозволяє приймати ефективні рішення в галузі горизонтального масштабування обчислювальних ресурсів (на вузлах розподіленої системи);

- здобуло подальший розвиток моніторинг використання інформаційних ресурсів, де збираються дані в реальному режимі часу про виконання програмних модулів в робочому середовищі високопродуктивних обчислень на основі відправлення попереджувальних повідомлень до системи сповіщень, яка допомагає виявляти небезпечний розвиток навантажувальних впливів.

Практична цінність отриманих результатів. Практичне значення отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритм попередження розвитку bottlenecks у програмному забезпеченні в інформаційних системах та розроблено програмні засоби.

Особистий внесок здобувача. У магістерській кваліфікаційній роботі усі результати дослідження здобуті автором даної роботи самостійно. У роботі [1], опублікованій самостійно, автору належить формування постановки проблеми, мети роботи та основної частини.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів "Комютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023", (Одеса, 2023).

Публікації. Основні результати досліджень опубліковано в науковій праці у матеріалах конференції.

РОЗДІЛ 1
ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ АНАЛІЗА

1.1. Необхідність використання сучасних методів та розробки програмних засобів

Існує необхідність використання сучасних методів та розробки програмних засобів досліджень bottlenecks у високопродуктивних обчисленнях [1,2]. Дослідження bottlenecks у високопродуктивних обчисленнях має велику наукову важливість і актуальність з кількох ключових причин (рис. 1.1).

Рисунок 1.1 - Актуальінсть проведення досліджень bottlenecks у високопродуктивних обчисленнях

У високопродуктивних обчисленнях, де обробка великих обсягів даних та складні обчислення є стандартом, покращення продуктивності є критичним завданням. Дослідження вузьких місць може допомогти ідентифікувати проблемні точки і розробляти оптимізації, які підвищують ефективність обчислень [3].

Покращення продуктивності обчислень полягає в оптимізації та підвищенні швидкості та ефективності виконання обчислень на комп'ютерних системах. Це важливий аспект у багатьох галузях, де використовуються великі обчислювальні завдання, включаючи науку, інженерію, бізнес та інші.

Таблиця 1.1

Важливість та актуальність проведення досліджень bottlenecks у високопродуктивних обчисленнях

Чинники	Опис
Покращення продуктивності обчислень	Дослідження bottlenecks дозволяє розробити оптимізовані алгоритми і програми, що сприяють збільшенню швидкодії та ефективності обчислень. Це важливо для різних галузей, де обчислювальні завдання стають все більш об'ємними та складними.
Ефективне використання обчислювальних ресурсів	Забезпечення оптимального використання обчислювальних ресурсів, таких як CPU, GPU, мережі та пам'ять, допомагає економити час і кошти. Дослідження bottlenecks важливо для оптимізації обчислювальних задач та зменшення навантаження на інфраструктуру.
Підтримка нових технологій і архітектур	З розвитком обчислювальних технологій і архітектур з'являються нові можливості та виклики. Дослідження bottlenecks дозволяє адаптувати програми до нових архітектур і забезпечити сумісність з новими технологіями, такими як квантові обчислення або великі дані.
Вдосконалення наукових досліджень	Високопродуктивні обчислення є важливою частиною наукових досліджень у багатьох галузях, включаючи фізику, біологію, хімію, геологію і багато інших. Оптимізація продуктивності дозволяє дослідникам обробляти та аналізувати великі обсяги даних та

	виконувати складні моделювання швидше і ефективніше.
Застосування в індустрії та бізнесі	У сучасному бізнесі та індустрії високопродуктивні обчислення використовуються для різних завдань, включаючи аналіз даних, машинне навчання, фінансовий аналіз та інше. Ефективні обчислення можуть призвести до економії коштів і збільшення конкурентоспроможності.
Забезпечення стабільної роботи систем	Дослідження bottlenecks також важливе для забезпечення стабільної роботи інформаційних систем, особливо в критичних сферах, таких як медицина та авіація. Відсутність вузьких місць допомагає уникнути відмов та незручностей для користувачів.

Основні методи та стратегії покращення продуктивності обчислень включають оптимізацію алгоритмів, де вибір або розробка окремих алгоритмів виконують задачу більш ефективно. Це може включати в себе використання алгоритмів з меншою обчислювальною складністю або оптимізацію існуючих алгоритмів.

Також, до методів та стратегій покращення продуктивності обчислень відносяться паралельне обчислення, використання високопродуктивних обчислювальних архітектур, кешування даних, оптимізація пам'яті, прпрофілювання та аналіз продуктивності апаратне прискорення, керування ресурсами, використання нових технологій та інше.

Покращення продуктивності обчислень допомагає зменшити час виконання завдань, зекономити ресурси та забезпечити більшу ефективність в різних областях, включаючи науку, інженерію, бізнес та багато інших галузей.

Ефективне використання обчислювальних ресурсів полягає у забезпеченні оптимального використання обчислювальних ресурсів, включаючи суперкомп'ютери та обчислювальні кластери, є важливим завданням. Дослідження вузьких місць може допомогти виявити, як ресурси розподіляються та як можна покращити їхнє використання.

Ефективне використання обчислювальних ресурсів означає максимально раціональне та продуктивне використання доступних обчислювальних потужностей, таких як процесори, пам'ять, сховище даних і мережеві з'єднання. Це важливо для забезпечення оптимальної продуктивності системи та ефективного вирішення завдань.

Основні аспекти ефективного використання обчислювальних ресурсів включають [4]: використання паралельності, розподіл завдань, оптимізація алгоритмів, керування пам'яттю, резервування ресурсів, моніторинг та аналіз продуктивності, автоматизація та оптимізація процесів, використання спеціалізованих пристроїв таких як спеціалізовані апаратні пристрої, такі як графічні процесори (GPU), FPGA або прискорювачі тензорних операцій (TPU), для виконання обчислень, які вони можуть виконувати набагато швидше за центральний процесор.

Ефективне використання обчислювальних ресурсів допомагає забезпечити максимальну продуктивність та ефективність обчислень, що є важливим у наукових дослідженнях, інженерних проектах, бізнес-завданнях та багатьох інших областях.

З ростом технологічного прогресу і виникненням нових архітектур обчислювальних систем з'являються нові вузькі місця. Дослідження їх може допомогти адаптувати програмні рішення до нових технологій та визначити оптимальні стратегії для їх використання.

Підтримка нових технологій і архітектур в обчисленнях означає здатність програмного та апаратного забезпечення пристосовуватися до інноваційних технологій та обчислювальних архітектур, які з'являються на ринку.

Ця підтримка є важливою для забезпечення конкурентоспроможності та ефективності обчислювальних систем і має такі ключові аспекти як адаптація до нових архітектур та оптимізація паралельних систем.

Споживання енергії стає важливою проблемою. Підтримка нових технологій включає в себе оптимізацію програмного забезпечення для максимально ефективного використання енергії і збереження енергоресурсів.

Сучасні процесори мають різноманітні набори інструкцій та функціональні можливості. Тому, підтримка нових технологій включає в себе використання спеціалізованих інструкцій і бібліотек для покращення продуктивності, апаратне прискорення, розробка для обробки великих обсягів даних, безпека і захист даних.

Поява нових мов програмування та фреймворків робить актуальною підтримку цих інструментів для створення програмного забезпечення. Підтримка нових технологій і архітектур дозволяє обчислювальним системам залишатися актуальними, конкурентоспроможними та здатними вирішувати нові завдання і виклики [3-5]. Це особливо важливо в швидкоплинних та інноваційних галузях, таких як інформаційні технології, наука та дослідження, медицина, фінанси та інші.

В багатьох наукових областях, включаючи фізику, біологію, кліматологію та інші, високопродуктивні обчислення є ключовим інструментом. Збільшення продуктивності дозволяє вченим аналізувати дані та виконувати моделювання в більших масштабах і з більшою точністю. Тому, вдосконалення наукових досліджень включає в себе низку заходів та стратегій, спрямованих на покращення якості та результативності наукових досліджень у різних галузях науки і технологій [4].

Розробка та застосування нових методів та підходів до дослідження, дозволяють отримувати більш точні, об'єктивні та надійні результати. Це може включати в себе використання нових інструментів аналізу даних, статистичних методів, математичних моделей тощо.

Впровадження сучасних технологій, дозволяють покращити обробку та аналіз даних, зробити дослідження більш швидким та ефективним. Це може включати в себе використання обчислювальних потужностей, сучасних лабораторних пристроїв, програмного забезпечення для моделювання тощо.

Співпраця та обмін знаннями між різними науковими галузями та дослідниками є досить важливими. Це дозволяє вирішувати більш складні проблеми та відкривати нові можливості для дослідження.

Забезпечення доступності даних, наукових публікацій і результатів досліджень для інших дослідників, сприяє перевірці та реплікації досліджень, а також забезпечує більшу прозорість та довіру в наукових гуртках.

Розвиток ефективних методів управління дослідженнями, включає планування, ресурси, графіки виконання та контроль якості. Оптимізований процес дозволяє використовувати ресурси більш раціонально та зменшує час на проведення досліджень.

Сприяння спілкуванню та співпраці між дослідниками, вченими та науковими групами з різних країн і галузей дозволяє обмінюватися ідеями, ресурсами і результатами досліджень для розширення знань і вирішення глобальних проблем. При цьому, існує всебічне фінансування та підтримка для наукових досліджень та інноваційних проектів. Фінансування дозволяє вченим та дослідникам зосередитися на важливих завданнях та експериментах.

Таким чином, вдосконалення наукових досліджень є ключовим аспектом наукового прогресу і сприяє вирішенню різних соціальних, технологічних та екологічних викликів [4-6]. Такий підхід допомагає створювати нові знання, вдосконалювати технології, вирішувати проблеми та сприяє розвитку суспільства в цілому.

Значні вдосконалення в продуктивності обчислень також мають велике значення для підприємств і промисловості. За допомогою високопродуктивних обчислень, бізнеси можуть оптимізувати процеси, проводити аналіз даних та розробляти нові продукти та послуги.

Застосування в індустрії та бізнесі включає в себе використання результатів наукових досліджень та новітніх технологій для поліпшення виробничих процесів, оптимізації управління, підвищення продуктивності та конкурентоспроможності підприємств і підтримки прийняття стратегічних рішень [5].

Використання автоматизації, роботизації та IoT-технологій (Інтернет речей) допомагає оптимізувати та автоматизувати виробничі процеси, зменшувати витрати ресурсів і підвищувати якість продукції.

Управління ланцюгом постачання та аналітикою даних, штучного інтелекту і блокчейн-технологій дозволяє оптимізувати управління ланцюгом постачання, забезпечуючи більшу прозорість, ефективність та безпеку в постачальному ланцюгу.

Використання аналітики даних та машинного навчання допомагає бізнесам аналізувати великі обсяги даних для прийняття більш обґрунтованих стратегічних рішень і передбачення тенденцій ринку. Оптимізація споживання енергії, води, матеріалів та інших ресурсів дозволяє підприємствам заощаджувати кошти і сприяє сталому розвитку.

Застосування сучасних технологій контролю та моніторингу допомагає забезпечити високу якість продукції та безпеку виробничих процесів.

Багато підприємств переходять до цифрових моделей бізнесу, що дозволяє їм підтримувати конкурентоспроможність у цифровій економіці та розвивати нові цифрові послуги. Застосування в індустрії та бізнесі сприяє підвищенню ефективності, зниженню витрат, розвитку нових ринків і підтримці сталого розвитку. Технологічні інновації і науковий прогрес відіграють важливу роль у розвитку сучасного бізнесу та індустрії. Тому, у великих системах, де надійність і продуктивність важливі, виявлення та усунення вузьких місць може допомогти попередити аварії та збої.

Забезпечення стабільної роботи систем (або системний стабільність) означає здатність комп'ютерних, програмних або інших інформаційних систем функціонувати безперебійно та надійно впродовж тривалого періоду часу. Це важливий аспект для багатьох областей, включаючи інформаційні технології, інженерію, виробництво, медицину та інші галузі.

Система повинна бути стійкою до випадкових або непередбачуваних відмов. Це може бути досягнуто за допомогою резервного копіювання, резервних апаратних засобів та інших заходів. При виникненні відмови система повинна бути здатною відновлювати свою роботу. Це включає в себе процес відновлення даних та конфігурацій, а також відновлення послуг та функцій. Тому, система повинна мати засоби виявлення і корекції помилок. Це може бути

досягнуто за допомогою механізмів самодіагностики і автоматичного виправлення [4-6].

Системи моніторингу та аналізу дозволяють вчасно виявляти проблеми і незвичайну активність, що допомагає уникнути відмов та простоїв. Використання резервних апаратних засобів та резервних джерел живлення може забезпечити стабільність у разі виходу з ладу основних ресурсів.

Ефективне використання обчислювальних, мережевих та інших ресурсів допомагає уникнути перевантаження та забезпечити стабільність. Використання технологій і методів, що ізолюють помилки від інших частин системи, може запобігти розповсюдженню помилок та впливу їх на інші компоненти.

Розробка планів дій та процедур для реагування на можливі аварійні ситуації і відновлення роботи системи.

Підготовка працівників до правильного реагування на аварійні ситуації та виконання процедур відновлення. Тому, забезпечення стабільної роботи систем є критично важливим завданням для будь-якого підприємства чи організації, оскільки від цього залежить безперебійність бізнес-процесів і надійність послуг, що надаються.

Таким чином, дослідження bottlenecks є ключовим елементом для підвищення ефективності обчислень та забезпечення стійкої роботи систем у сферах науки, індустрії та досліджень.

1.2. Задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях

Розробка методів та програмних засобів для дослідження bottlenecks у високопродуктивних обчисленнях включає в себе різноманітні завдання та задачі. Основні задачі цього процесу включають наступні завдання (рис. 1.2).

Рисунок 1.2 - Основні задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях

Збір даних про продуктивність включає розробку засобів для моніторингу та збору даних про продуктивність обчислювальних систем. Це включає в себе вимірювання часу виконання обчислювальних завдань, обсяг використовуваної пам'яті, навантаження на процесор, швидкість передачі даних через мережу і багато іншого [6-8].

Збір даних про продуктивність високопродуктивних обчислень - це процес збору і реєстрації різноманітних даних і показників, які відображають продуктивність обчислювальної системи або програми. Цей процес важливий для виявлення bottlenecks і потребує збору різноманітної інформації, щоб зрозуміти, як система працює і які можливості її оптимізації і покращення продуктивності. Основні аспекти збору даних про продуктивність включають наступні чинники.

Час виконання задач, що вимірює час, який потрібно системі для виконання конкретної обчислювальної задачі або операції. Це може включати в

себе час виконання окремих функцій, операцій читання/запису на диск або мережеві операції.

Споживана пам'ять за допомогою якої фіксується обсяг оперативної пам'яті, що використовує програма або процес під час виконання. Це важливо для визначення ефективності використання пам'яті та ідентифікації можливих проблем зі справжньою або витіканням пам'яті (табл. 1.2).

Таблиця 1.2

Задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях

Задача	Опис
Збір даних про продуктивність	Розробка інструментів для систематичного збору даних про продуктивність обчислювальних задач, включаючи час виконання, споживану пам'ять, завантаження процесорів тощо.
Аналіз даних про продуктивність	Розробка методів та алгоритмів для аналізу зібраних даних з метою виявлення вузьких місць та аномалій у продуктивності системи.
Визначення метрик продуктивності	Розробка об'єктивних метрик та показників для вимірювання продуктивності, таких як швидкість, пропускна здатність, час відгуку, кількість оброблених завдань тощо.
Візуалізація даних	Розробка інтерфейсів та засобів візуалізації результатів досліджень для кращого сприйняття даних користувачами і адміністраторами систем.
Специфікації вузьких місць	Документування та класифікація виявлених bottlenecks з метою подальшої оптимізації та розробки рекомендацій для виправлення проблем.
Підтримка різних обчислювальних платформ	Розробка програмних рішень, які можуть працювати на різних архітектурах та платформах, включаючи CPU, GPU, FPGA, хмарні обчислення тощо.
Налаштування та управління	Розробка інструментів для налаштування параметрів роботи системи з метою досягнення оптимальної продуктивності та її подальшого контролю та управління.

Забезпечення стабільної роботи системи	Розробка методів та засобів для виявлення та усунення проблем, які можуть впливати на стабільність та надійність роботи обчислювальної системи.
Журнали подій та відладка	Розробка системи ведення журналів подій та інструментів для відладки та виявлення помилок, що можуть виникати під час роботи системи та впливати на продуктивність.
Дослідження та впровадження інноваційних підходів	Розробка нових методів та технологій для виявлення та оптимізації вузьких місць, а також впровадження інноваційних рішень для покращення продуктивності обчислень.
Підтримка розширених функціональних можливостей системи	Розробка засобів для підтримки розширених функціональних можливостей системи, таких як паралельні обчислення, великі обсяги даних, машинне навчання тощо.

Використання процесора, що виконує моніторинг навантаження на центральний процесор (CPU) дозволяє виявити, скільки обчислювальних ресурсів використовується для виконання задачі і чи існують обмеження процесора.

Мережева активність, що використовується у випадку, якщо система взаємодіє з іншими системами через мережу, то важливо моніторити обсяг передачі даних, затримки і швидкість передачі для визначення ефективності мережевих операцій.

Навантаження на диск, що вимірює обсяг операцій читання/запису на диск та може виявити можливі бутланеки на рівні зберігання даних.

Інші параметри - в залежності від конкретного завдання можуть вимірюватися інші параметри, такі як кількість запитів до бази даних, обсяги обміну даними між процесами, обсяги даних у кешах тощо.

Збір даних про продуктивність допомагає ідентифікувати області [5-8], в яких можливе покращення продуктивності, і надає об'єктивну основу для аналізу та оптимізації системи чи програми.

Розробка алгоритмів і методів аналізу зібраних даних для виявлення bottlenecks. Це може включати в себе визначення оптимальних параметрів роботи системи, виявлення аномалій та невідповідностей у продуктивності.

Аналіз даних про продуктивність - це процес обробки та інтерпретації зібраних даних про продуктивність обчислювальної системи або програми з метою виявлення проблем, вузьких місць та можливостей для оптимізації. Основна мета аналізу полягає в розумінні того, як ефективно працює система і як її продуктивність може бути покращена. Основні етапи аналізу даних про продуктивність включають наступне.

Першим кроком є обробка зібраних даних, включаючи їх очищення від помилок, фільтрацію та перетворення у зручний для аналізу формат. Це може включати в себе обчислення додаткових параметрів і показників, які будуть корисними для подальшого аналізу. Важливо визначити ключові метрики та показники продуктивності, які слід аналізувати. Це можуть бути такі показники, як час виконання, пропускна здатність, завантаження процесора, споживана пам'ять, швидкість мережі тощо.

Використання графіків і діаграм для візуалізації даних. Графіки можуть допомогти виявити залежності та тренди у продуктивності, які можуть бути невидимими на рівні окремих чисел.

Використання статистичних методів для виявлення вузьких місць, аномалій або статистично значущих відмінностей у продуктивності. Це може включати аналіз розподілу даних, кореляційний аналіз, регресійний аналіз тощо.

Ідентифікація проблем та вузьких місць, що основна на меті аналізу - це виявити проблеми, які впливають на продуктивність системи, і визначити їх джерела. Це може бути підвищений обсяг запитів до бази даних, перевантаження мережі, неефективне використання ресурсів процесора тощо.

На основі результатів аналізу розробляються рекомендації для покращення продуктивності. Це може включати в себе зміни в програмному коді, налаштування системи, впровадження кешування, оптимізацію запитів до бази даних тощо.

Перевірка та впровадження змін основана на оптимізаційних заходах, що впроваджуються в систему, і продуктивність перевіряється після впровадження, щоб переконатися, що зміни призвели до покращення.

Аналіз даних про продуктивність є ключовим етапом у виявленні, вирішенні та усуненні проблем, які можуть виникати під час високопродуктивних обчислень, і у покращенні ефективності системи або програми.

Розробка метрик та показників, які дозволяють об'єктивно вимірювати продуктивність системи. Це може включати в себе такі метрики, як час відгуку системи, пропускна здатність, кількість оброблених завдань на одиницю часу тощо.

Визначення метрик продуктивності - це процес обрання та визначення конкретних показників і параметрів, які будуть використовуватися для вимірювання продуктивності обчислювальної системи, програми чи процесу. Метрики продуктивності допомагають отримати кількісну інформацію про ефективність роботи системи та стануть основою для подальшого моніторингу, аналізу та оптимізації. Основні критерії визначення метрик продуктивності включають наступні чинники. Метрики повинні бути конкретними і відображати певні аспекти продуктивності, які важливі для конкретного завдання. Наприклад, для веб-сайту це може бути час завантаження сторінки або кількість запитів за секунду.

Метрики повинні бути об'єктивними і вимірюватися за допомогою об'єктивних інструментів чи методів. Наприклад, час відповіді сервера вимірюється точною системою моніторингу, а не оцінюється за відчуттями користувача.

Метрики повинні бути чутливими до змін в продуктивності. Це означає, що вони повинні виявляти вплив змін в програмному коді, обладнанні або навантаженні на систему.

Метрики повинні бути відносно легкими для збору та аналізу. Складні метрики можуть вимагати більше ресурсів і часу для вимірювання та обробки.

Метрики повинні бути повторюваними, щоб можна було виміряти продуктивність у різний час і порівнювати результати.

Прикладами метрик продуктивності можуть бути наступні.

Середній час відповіді (Average Response Time) - це час, який система витрачає на відповідь на запити користувачів.

Пропускна здатність (Throughput) - це кількість операцій чи запитів, які система може обробити за одиницю часу.

Споживана пам'ять (Memory Consumption) - це кількість оперативної пам'яті, яку використовує програма.

Час завантаження сторінки (Page Load Time) - це час, який потрібно для завантаження веб-сторінки.

Використання процесора (CPU Usage) - це відсоток обчислювальних ресурсів, які використовуються процесором.

Визначення метрик продуктивності допомагає зрозуміти, наскільки добре працює система і які аспекти можуть бути покращені або оптимізовані.

Графічне відображення результатів досліджень допомагає користувачам краще розуміти продуктивність системи і приймати рішення щодо її оптимізації.

Візуалізація даних - це процес відображення інформації у вигляді графічних елементів, таких як графіки, діаграми, схеми, карти, теплові карти та інші візуальні об'єкти. Головна мета візуалізації - робити дані більш зрозумілими, зручними для сприйняття та аналізу, а також виділяти шаблони, тренди та відмінності в даних. Основні аспекти візуалізації даних включають наступні чинники.

Візуалізація може використовувати різні графічні елементи, такі як лінії, бари, точки, пізні, зони тощо, для представлення даних в різних формах.

Використання кольорів дозволяє виділити різні категорії даних, позначити важливість або інші характеристики даних.

Для пояснення графіків та діаграм, важливо включати легенди та підписи, які роз'яснюють значення та властивості даних.

Візуалізація може бути інтерактивною, дозволяючи користувачам взаємодіяти з даними, збільшувати, переміщати або фільтрувати візуальні елементи [12,14]. Деякі візуальнізації можуть представляти дані в тривимірному просторі, що додає глибину та розуміння. Для динамічних даних анімація може використовуватися для відслідковування змін у часі.

Вибір правильного типу візуалізації залежить від характеру даних і мети аналізу. До типів візуалізації входять лінійні графіки, кругові діаграми, стовпчикові діаграми, графіки розсіювання, теплові карти, графи, дерева тощо.

Візуалізація даних допомагає спростити складні дані, робити їх більш доступними для розуміння та приймати на їхній основі узагальнені висновки. Вона є потужним інструментом для аналітиків даних, науковців та приймачів рішень у багатьох галузях, включаючи бізнес, медицину, наукові дослідження та інші [14,15].

Документування та класифікація виявлених bottlenecks важливо для подальшої оптимізації. Це може включати в себе опис типових сценаріїв, які призводять до виникнення вузьких місць.

Специфікації вузьких місць (Bottleneck Specifications) - це опис або характеристика конкретних обмежень чи обставин, які обмежують продуктивність чи ефективність системи, програми або процесу. Вузькі місця можуть виникати у будь-якій обчислювальній системі і призводити до зниження продуктивності або погіршення ефективності. Специфікації вузьких місць допомагають ідентифікувати ці обмеження і визначити, де та чому вони виникають. Основні аспекти специфікацій вузьких місць включають наступні чинники.

Місце виникнення, що вказує, де саме в системі, програмі або процесі виникає вузьке місце. Наприклад, це може бути підсистема, модуль коду, мережеве з'єднання, обладнання тощо.

Причина виникнення, що пояснює, чому вузьке місце виникає. Причина може бути обмеженням обладнання (наприклад, обмеження ресурсів процесора

чи пам'яті), недоліком оптимізації коду, зростанням навантаження, неправильною конфігурацією системи та іншими факторами.

Вплив на продуктивність, що описує, як вузьке місце впливає на продуктивність системи чи процесу. Наприклад, це може призводити до зниження швидкості обробки запитів, підвищення часу відповіді або інших негативних наслідків.

Критичність, що вказує на те, наскільки критичним є вузьке місце для загальної продуктивності чи ефективності системи. Деякі вузькі місця можуть бути критичними та потребувати негайного виправлення, тоді як інші можуть мати менший вплив.

Рекомендації щодо виправлення занадто великих значень у метриках включає в себе рекомендації та стратегії для виправлення вузького місця. Це може включати оптимізацію коду, збільшення ресурсів, перерозподіл завдань, встановлення обладнання чи інші заходи.

Специфікації вузьких місць є важливим інструментом для розробників та аналітиків, які працюють над оптимізацією та покращенням продуктивності систем. Вони допомагають ідентифікувати проблемні місця та розробляти стратегії для їх вирішення, що в свою чергу сприяє покращенню роботи системи та забезпечує задоволення потреб користувачів.

Налаштування та управління в програмному засобі для дослідження bottlenecks у високопродуктивних обчисленнях відіграють важливу роль у забезпеченні правильної роботи та ефективного використання інструменту. Ця функціональність передбачає можливість налаштовувати різні параметри та параметризувати процеси дослідження, а також забезпечує зручний інтерфейс для керування та моніторингу роботи програми. Основні аспекти налаштування та управління включають наступні елементи.

Конфігурація параметрів дослідження, де користувач повинен мати можливість визначати параметри дослідження, такі як об'єм даних, тривалість експериментів, вибір метрик для моніторингу та інші параметри.

Вибір платформи та ресурсів, де програмний засіб повинен дозволяти вибирати обчислювальну платформу, на якій він буде працювати, а також розподіляти ресурси, наприклад, кількість ядер процесора або обсяг доступної пам'яті.

Керування експериментами, де існує необхідність забезпечення можливості запуску, призупинення та завершення експериментів [14,16]. Користувач повинен мати зручний інтерфейс для керування цими операціями.

Налаштування способів збору даних визначає те, які дані будуть збиратися під час дослідження, та налаштування інтервалів збору даних.

Управління ресурсами здійснюється контролем над використанням ресурсів системи, щоб уникнути перевантаження та забезпечити стабільну роботу. Журналювання та збереження налаштувань записує та зберігає налаштування, щоб користувач міг повторно використовувати їх у майбутніх дослідженнях. Моніторинг та звіти забезпечує можливості візуального моніторингу ходу дослідження та генерація звітів про продуктивність та результати.

Автоматизація процесів існує для надання можливості автоматизувати деякі операції та процеси, щоб зменшити вручну втручання та покращити ефективність.

Налаштування та управління важливі для забезпечення зручного та ефективного використання програмного засобу для дослідження bottlenecks, дозволяючи користувачеві гнучко налаштовувати і керувати процесом дослідження відповідно до своїх потреб і завдань.

Розробка методів і інструментів існує для виявлення та усунення проблем, які можуть призводити до відмов або зниження продуктивності системи.

Забезпечення стабільної роботи системи в контексті дослідження bottlenecks у високопродуктивних обчисленнях включає в себе низку заходів та практик, спрямованих на збереження функціональності та продуктивності програмного засобу на протязі тривалого часу та в різних умовах. Це особливо важливо для досліджень, оскільки непередбачувані витоки ресурсів або падіння

продуктивності можуть спотворити результати. Основні аспекти забезпечення стабільної роботи системи включають наступне.

Постійний моніторинг роботи системи та аналіз даних про продуктивність допомагають виявляти аномалії та вузькі місця у реальному часі. Це дозволяє оперативно реагувати на проблеми та уникати надзвичайних ситуацій.

Забезпечення ефективного використання ресурсів системи шляхом оптимізації алгоритмів та структур даних. Використання резервування ресурсів дозволяє уникнути конфліктів та перевантажень.

Розробку механізмів автоматичного виявлення помилок та їхнього відновлення допомагає уникнути зупинок системи через некоректну роботу.

Проведення ретельного тестування програмного засобу на різних платформах та у різних умовах дозволяє виявити та усунути проблеми перед випуском виробу.

Забезпечення можливості регулярного створення резервних копій даних та системи, щоб у випадку витоку ресурсів або виходу з ладу можна було швидко відновити роботу [18,19]. Забезпечення безпеки системи шляхом виявлення та захисту від потенційних загроз, таких як злами, віруси або атаки з боку зовнішніх зловмисників.

Регулярне оновлення та підтримка операційних систем, бібліотек, та інфраструктури, на якій працює програмний засіб.

Забезпечення стабільної роботи системи важливо для збереження надійності та достовірності результатів досліджень, а також для забезпечення високої якості роботи програмного засобу під час використання в реальних умовах.

Розробка системи ведення журналів подій та інструментів є досить важливими для відладки та виявлення помилок, які можуть впливати на продуктивність.

Журнали подій та відладка (Event and Debug Logs) - це важливий аспект розробки і використання програмного засобу для дослідження bottlenecks в високопродуктивних обчисленнях. Ці інструменти допомагають відслідковувати

події та помилки під час роботи програми, виправляти проблеми та аналізувати роботу системи. Вони надають змогу розробникам і дослідникам збирати інформацію про виконання програми та виявляти проблемні моменти. Основні аспекти журналів подій та відладки включають наступне.

Запис подій, де програма веде журнал подій, реєструючи важливі події, такі як запуск, завершення експерименту, вибір параметрів тощо. Це дозволяє відслідковувати порядок виконання операцій та аналізувати послідовність подій. Запис помилок, де розроблена спеціальна відладка дозволяє виявляти та реєструвати помилки в програмі. Це важливо для ідентифікації та виправлення недоліків в програмному коді.

Логування даних про продуктивність при якому журнали можуть містити інформацію про продуктивність, таку як час виконання операцій, завантаження процесора, використання пам'яті тощо. Це допомагає аналізувати ефективність програми та виявляти вузькі місця у продуктивності.

Різні рівні журналування за допомогою яких журнали подій мають різні рівні, включаючи інформаційний, попередження та помилки. Це дозволяє користувачам налаштовувати, яку інформацію записувати в журнал, щоб не перенасичувати його непотрібною інформацією.

Відладка - це процес виявлення, аналізу та виправлення помилок у програмному коді. Використовуючи журнали подій, розробник може відслідковувати послідовність операцій та визначати, де саме виникають проблеми. За допомогою журналів можна аналізувати події, що передували помилкам або падінням продуктивності, і визначити імовірні причини цих проблем.

Деякі програмні засоби дозволяють віддалено аналізувати журнали подій, що дозволяє виправляти проблеми без фізичного доступу до обчислювального обладнання.

Всі ці аспекти допомагають розробникам і дослідникам забезпечити стабільну та ефективну роботу програмного засобу для дослідження bottlenecks

та виявити та вирішити проблеми, які можуть вплинути на результати дослідження.

Таким чином, задачі є важливими для забезпечення ефективності та стабільності високопродуктивних обчислень і вимагають інноваційних підходів і рішень у галузі інженерії програмного забезпечення та обчислювальної науки.

1.3. Аналіз сучасних методів розробки програмних засобів для дослідження bottlenecks

Аналіз сучасних методів та розробки програмних засобів для дослідження bottlenecks у високопродуктивних обчисленнях - це важливий етап для покращення продуктивності та ефективності обчислювальних систем (рис. 1.3). Наведемо деякі з основних аспектів цього аналізу.

Сучасні програмні засоби надають можливість збирати різноманітні дані про продуктивність, включаючи час виконання операцій, завантаження процесора, використання пам'яті, швидкість передачі даних тощо. Важливо аналізувати, які методи збору даних використовуються і які нові підходи можуть бути введені для збору більш повної та релевантної інформації (табл. 1.3).

Методи збору даних про продуктивність включають в себе моніторинг і запис даних про час виконання операцій, використання ресурсів (CPU, пам'ять, диск), а також збір структурованих журналів подій та метрик продуктивності.

Сучасні методи аналізу включають в себе використання статистичних і математичних моделей для ідентифікації bottlenecks. Методи машинного навчання та інші аналітичні техніки допомагають виявляти закономірності та тренди у продуктивності системи.

Методи аналізу продуктивності включають в себе використання статистичних та математичних моделей для виявлення та ідентифікації вузьких місць у високопродуктивних обчисленнях.

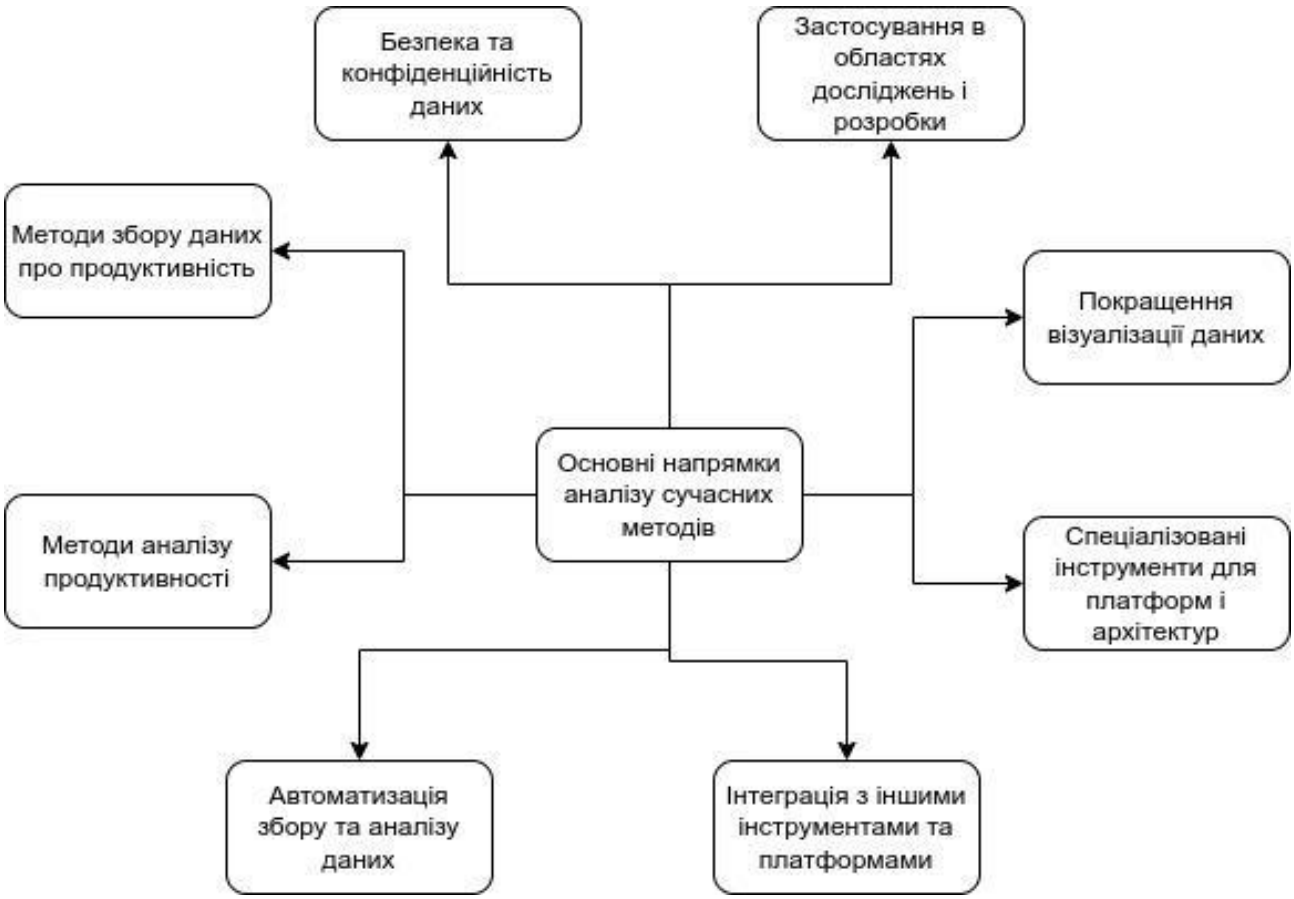


Рисунок 1.3 - Аналіз сучасних методів та розробки програмних засобів для дослідження bottlenecks у високопродуктивних обчисленнях

Сучасні засоби все більше надають можливості автоматизувати процеси збору та аналізу даних. Це дозволяє зекономити час і зусилля дослідників та розробників. Автоматизація збору та аналізу даних полягає в використанні програмних інструментів для автоматизованого збору, обробки і аналізу інформації про продуктивність без значного втручання користувача.

Зручність інтеграції програмного засобу для дослідження bottlenecks з іншими інструментами та платформами стає все важливішою. Це дозволяє використовувати дані про продуктивність в різних аспектах розробки та управління обчислювальною інфраструктурою.

Таблиця 1.3

Опис аналізу сучасних методів та розробки програмних засобів для дослідження bottlenecks у високопродуктивних обчисленнях

Чинник	Опис
Методи збору даних про продуктивність	Збір різноманітних даних про продуктивність систем, таких як час виконання операцій та використання ресурсів.
Методи аналізу продуктивності	Використання статистичних і математичних моделей для ідентифікації bottlenecks.
Автоматизація збору та аналізу даних	Автоматизація процесів збору та аналізу даних для зекономлення часу та ресурсів.
Інтеграція з іншими інструментами та платформами	Забезпечення зручної інтеграції з іншими інструментами розробки та управління обчислювальною інфраструктурою.
Спеціалізовані інструменти для платформ і архітектур	Розробка інструментів, спеціалізованих для конкретних видів апаратного забезпечення.
Покращення візуалізації даних	Розробка засобів для побудови графіків та діаграм для кращого розуміння даних про продуктивність.
Застосування в областях досліджень і розробки	Адаптація методів та інструментів до конкретних потреб різних галузей.
Безпека та конфіденційність даних	Забезпечення безпеки та конфіденційності даних, особливо у чутливих областях.

Інтеграція з іншими інструментами та платформами - це забезпечення можливості обміну даними із сторонніми програмами та системами для покращення функціональності і розширення можливостей дослідження продуктивності.

З розвитком різних обчислювальних платформ і архітектур з'являються спеціалізовані інструменти для їхнього аналізу та вимірювання продуктивності. Для кожного типу апаратного забезпечення можуть бути розроблені інструменти, що допомагають виявити вузькі місця продуктивності.

Спеціалізовані інструменти для платформ і архітектур - це програмні засоби, розроблені для оптимізованого аналізу та дослідження продуктивності певних типів обчислювальних систем та архітектур.

Сучасні інструменти надають можливість побудови графіків, діаграм і інших візуальних елементів для кращого розуміння даних про продуктивність.

Покращення візуалізації даних - це використання додаткових методів та інструментів для створення більш інформативних та зрозумілих графіків та діаграм для представлення даних про продуктивність.

Важливо також враховувати, які конкретні завдання та завдання стоять перед дослідниками і розробниками у високопродуктивних обчисленнях та адаптувати методи та інструменти до конкретних потреб даних областей [19,20].

Застосування в областях досліджень і розробки - це можливість використовувати засоби для дослідження bottlenecks в різних наукових та технічних дисциплінах та для розробки оптимізованих програм та систем.

Збір і аналіз даних про продуктивність також повинен враховувати вимоги щодо безпеки та конфіденційності даних, особливо у випадках, коли обробка виконується в чутливих областях, таких як медицина або фінанси.

Безпека та конфіденційність даних - це заходи, спрямовані на забезпечення захисту інформації про продуктивність від несанкціонованого доступу та збереження конфіденційності даних користувачів.

Таким чином, аналіз і розробка методів та програмних засобів для дослідження bottlenecks постійно розвиваються, враховуючи зростаючу потребу у високопродуктивних обчисленнях в різних галузях, і цей процес є важливим для покращення ефективності та ресурсозбереження у сучасних обчислювальних системах.

1.4. Задачі дослідження

1. Виконати обгрунтування методу аналізу дослідження bottlenecks у високопродуктивних обчисленнях.

2. Розглянути необхідність використання сучасних методів та інструментів розробки програмних засобів.

3. Розробити задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях.

4. Виконано аналіз сучасних методів розробки програмних засобів для дослідження bottlenecks.

5. Розглянути загальна характеристика методів дослідження bottlenecks у високопродуктивних обчисленнях.

6. Дослідити особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях.

7. Запропонований метод діагностики bottlenecks у програмному засобі.

8. Визначити вимоги щодо функціонування програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.

9. Розробити структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.

10. Виконати реалізацію розробленого методу у програмному засобі.

11. Розробити вхідні та вихідні параметри програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.

12. Виконати обчислення вхідних параметрів для обробки даних у програмному засобі.

1.5. Висновки

В першому розділі виконувалось обгрунтування методу аналізу дослідження bottlenecks у високопродуктивних обчисленнях, де була розглянута необхідність використання сучасних методів та інструментів розробки програмних засобів. Розроблені задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях. Виконано аналіз сучасних методів розробки програмних засобів для дослідження bottlenecks.

РОЗДІЛ 2
ЗАГАЛЬНА ХАРАКТЕРИСТИКА МЕТОДІВ ДОСЛІДЖЕННЯ BOTTLENECKS

2.1. Особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях

Основна особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях полягає в тому, що ці методи дозволяють ідентифікувати та вирішувати проблеми, які обмежують ефективність обчислень у великих обчислювальних системах (рис. 2.1). Основна ідея полягає в тому, щоб знайти та виправити "вузькі місця" або обмеження, які гальмують продуктивність, тим самим забезпечити кращу ефективність використання обчислювальних ресурсів.

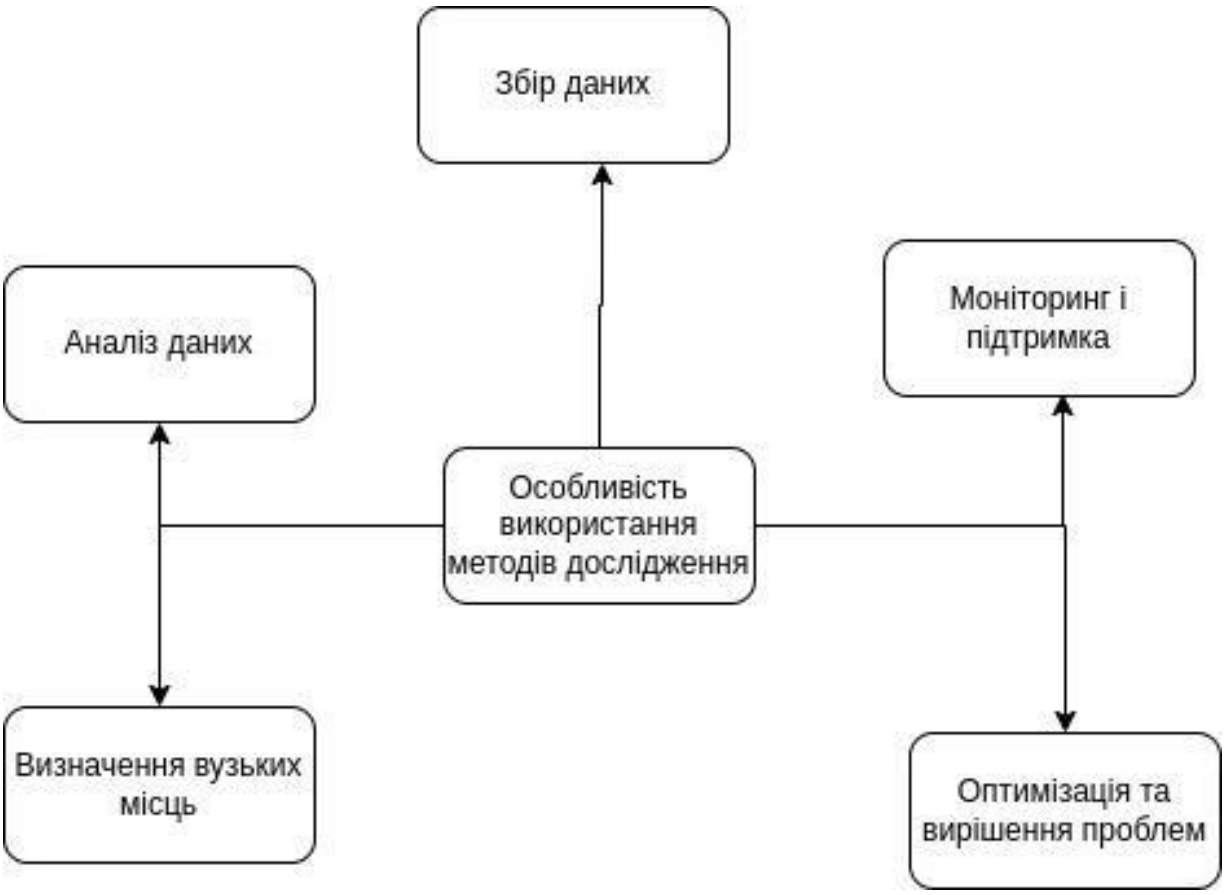


Рисунок 2.1 - Особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях

Дослідження bottlenecks включає в себе збір даних про продуктивність, їх аналіз та ідентифікацію проблемних ділянок. Після цього можна розробити стратегії для оптимізації роботи системи, виявити потреби в апаратному чи програмному оновленні, а також покращити загальну ефективність обчислень.

Основні переваги використання таких методів полягають у тому, що вони допомагають підвищити продуктивність системи, зменшити витрати на обчислювальні ресурси та енергію, а також поліпшити якість обчислень у високопродуктивних обчисленнях.

Особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях складається з кількох компонентів (табл. 2.1).

Таблиця 2.1

Особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях

Компоненти	Опис
Збір даних	Збір інформації про продуктивність системи.
Аналіз даних	Аналіз даних для виявлення аномалій та проблем.
Визначення вузьких місць	Визначення складових системи, що обмежують продуктивність.
Оптимізація та вирішення проблем	Розробка стратегій для вирішення виявлених проблем.
Моніторинг і підтримка	Постійний моніторинг та забезпечення стабільної роботи системи.

Збір даних включає в себе зібрання різноманітних даних про продуктивність системи [18,20]. Це може бути інформація про завантаження процесорів, використання пам'яті, швидкість введення-виведення даних, роботу мережі, інші параметри продуктивності.

Після збору даних проводиться їх аналіз. Це включає в себе виявлення аномалій, пошук кореляцій між різними параметрами та ідентифікацію проблемних ділянок.

В результаті аналізу визначаються ті компоненти системи або операції, які обмежують її продуктивність. Це можуть бути процесори, пам'ять, дисковий доступ, мережеві операції тощо.

Після ідентифікації вузьких місць розробляються стратегії для їх усунення або поліпшення. Це може включати в себе зміни в програмному коді, налаштування системних параметрів, оновлення обладнання або використання оптимізованих алгоритмів.

Розглянемо переваги та недоліки кожної з основних особливостей в дослідженні bottlenecks в високопродуктивних обчисленнях.

Перевага у зборі даних полягає в тому, що цей етам дозволяє отримувати об'єктивну інформацію про продуктивність системи. Також, безперервний моніторинг може допомогти вчасно виявити аномалії.

До недоліків слід віднести необхідність додаткового обладнання та програмних засобів для збору даних. Також, цей етап може вимагати значних ресурсів для зберігання та обробки великої кількості даних. Аналіз даних допомагає виявити проблеми та аномалії у продуктивності. Він дозволяє проводити глибокий аналіз даних для зрозуміння кореневих причин проблем.

До недоліків слід віднести необхідність великої кількості обчислень та обробки даних. Також, може бути занадто складні завданням при обробці великих обсягів інформації.

Визначення вузьких місць допомагає ідентифікувати конкретні компоненти системи, що обмежують продуктивність в системі. Також, орієнтований на визначену точку фокусу він дозволяє зосередитися на конкретних проблемах.

До недоліків слід віднести необхідність глибокого розуміння архітектури системи та специфічних алгоритмів. Також, не завжди він може виявити складні міжкомпонентні проблеми.

Етап оптимізації та вирішення проблем дозволяє покращити продуктивність і виправити існуючі проблеми. Також, може спрямовуватися на конкретні аспекти системи для досягнення кращої продуктивності.

До недоліків слід віднести необхідність витрат значного часу та ресурсів на розробку та впровадження оптимізаційних рішень. Також, цей етап може вимагати змін у коді програми або архітектурі самої системи.

Моніторинг і підтримка забезпечує стабільну роботу системи та попереджує аварії. Цей етап дозволяє реагувати на зміни у середовищі та витрати ресурсів.

До недоліків слід віднести необхідність постійного нагляду та затратних ресурсів для підтримки самої системи. Також, цей етап може стати занадто витратним завданням при масштабуванні системи.

Загалом, використання цих особливостей у високопродуктивних обчисленнях має за мету покращити продуктивність та стабільність системи, але може вимагати значних зусиль та ресурсів для їх впровадження та підтримки.

Таким чином, після вирішення проблем важливо постійно відслідковувати продуктивність і забезпечувати стабільну роботу системи. Це включає в себе постійний моніторинг та можливість реагувати на зміни у навантаженні або роботі системи. Такий комплексний підхід дозволяє виявляти та вирішувати проблеми продуктивності у високопродуктивних обчисленнях, щоб забезпечити ефективне використання обчислювальних ресурсів та покращити загальну продуктивність системи.

2.2. Запропонований метод попередження розвитку bottlenecks у програмному засобі

Метод попередження розвитку bottlenecks під час виконання програмного забезпечення в інформаційних системах, базується на експертному та експериментальному визначенні критичних точок та встановленні різних обробників у системах високопродуктивних обчислень (рис. 2.2).

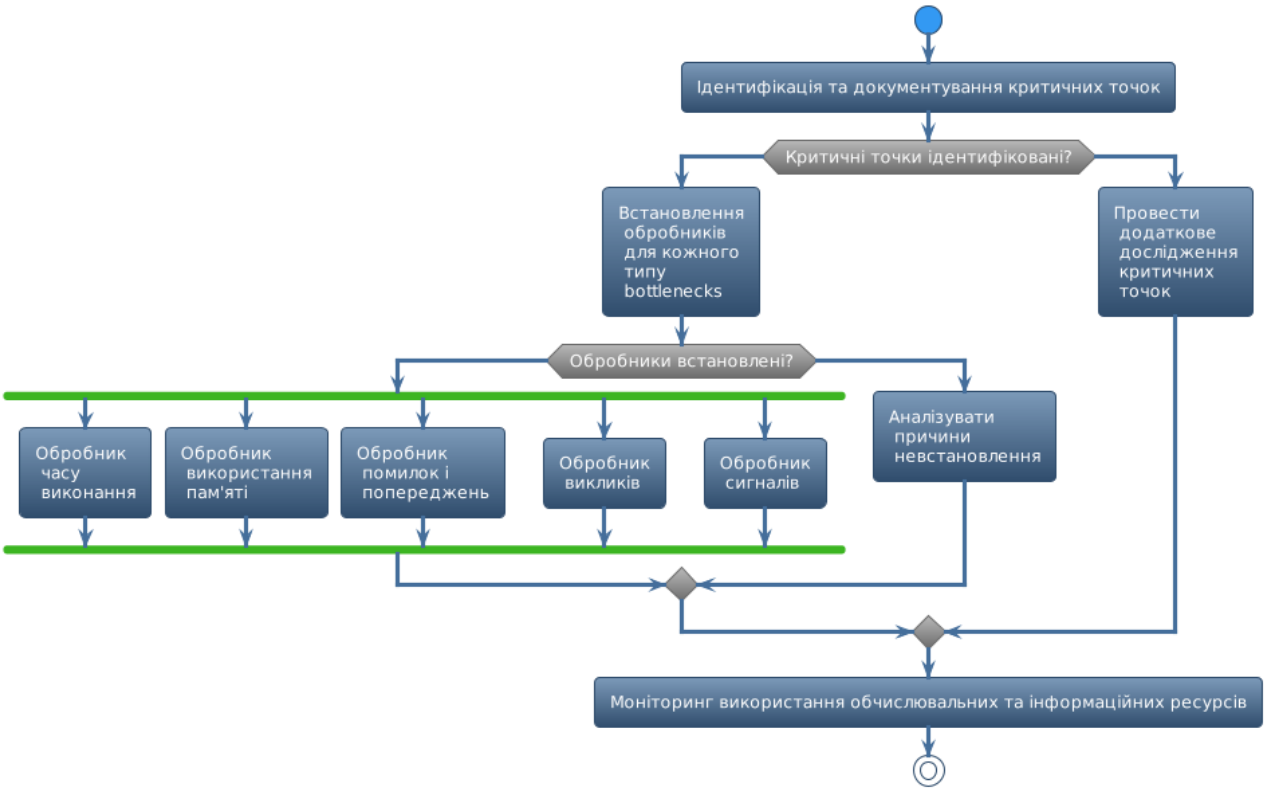


Рисунок 2.2 - UML діаграма визначенні критичних точок та встановленні різних обробників у системах високопродуктивних обчислень

При цьому, ідентифікація критичних точок являє собою досить серьйозну технічну проблему, яка вимагає сучасних рішень та полягає в настуному. Визначення критичних точок, починається з ідентифікації та документування критичних точок у інформаційній системі. Критичні точки - це ті місця в програмному коді, де можливі найбільші ризики, що позначаються на зниженні продуктивності, витіку пам'яті, зниженні надійності та інших негативних аспектів. Це можуть бути різні функціональні петлі, зайві функції, не досить ефективні алгоритми, модулі або інші важливі частини програми.

Після ідентифікації критичних точок, виконується встановлення обробників для кожного типу bottlenecks (табл. 2.2), які необхідно діагностувати.

Ідентифікація критичних точок - це перший крок у методі діагностики bottlenecks, спрямований на знаходження місць у програмному коді, де можливі найбільші проблеми з продуктивністю, пам'яттю, надійністю та іншими параметрами інформаційної системи.

Таблиця 2.2

Кроки проведення дослідження bottlenecks у високопродуктивних обчисленнях

Крок (Дія)	Опис
Визначення критичних точок	Починається з ідентифікації та документування критичних точок у вашій інформаційній системі. Критичні точки - це ті місця в програмному коді, де можливі найбільші ризики для продуктивності, пам'яті, надійності та інших аспектів. Це можуть бути петлі, функції, алгоритми, модулі або інші важливі частини програми.
Встановлення обробників	Після ідентифікації критичних точок, розробляється встановлення обробників для кожного типу bottlenecks
Обробник часу виконання	Вимірюється час виконання кожної критичної точки або функції. Використовується профілювання коду для знаходження найвитратніших за часом частин програми.
Обробник використання пам'яті	Вимірюється використання пам'яті кожною критичною точкою. Використовуються інструменти для аналізу пам'яті, щоб знайти утечки пам'яті або надмірне використання ресурсів.
Обробник помилок і попереджень	Моніторяться та журналюються помилки та попередження, які виникають в критичних точках. Важливо збирати інформацію про помилки для подальшого аналізу.
Обробник викликів	Вивчається, які функції або методи викликаються в кожній критичній точці. Це допомагає зрозуміти ланцюжок викликів та їх вплив на продуктивність.
Обробник сигналів	Моніториться обробка сигналів та виключень, які виникають у критичних точках. З'ясовується, як ці сигнали обробляються та впливають на роботу програми.

Аналіз функціональності критичних точок функціонального обстеження програмних засобів інформаційної системи. Визначаються ключові функції,

операції та процеси, які є критичними для роботи системи. Це можуть бути, наприклад, функції, які викликаються найчастіше, або операції, які вимагають найбільшого обсягу обчислень.

Також, використовуються інструменти для профілювання коду, що допомагають визначити, які фрагменти програми займають найбільше часу виконання, використовують найбільше пам'яті або видають найбільше помилок. Це може бути важливим для підказки на потенційні bottlenecks.

Під час моніторингу використання обчислювальних та інформаційних ресурсів збираються дані про реальне виконання програмних модулів в робочому середовищі. Це допомагає виявити ті частини програми, які найбільше впливають на її продуктивність та надійність під час роботи з реальними даними та навантаженням.

Аналіз взаємодій інших компонентів розглядає взаємодії з іншими компонентами системи, такими як бази даних, мережеві служби або зовнішні API. Це допомогає проаналізувати, як взаємодія з цими компонентами впливає на продуктивність і роботу системи в цілому.

Для врахування вимог користувачів береться до уваги різні чинникі, як користувачі взаємодіють з системою і які функції вони вважають найбільш важливими. Це може вказати на ті частини системи, які потребують особливої уваги під час ідентифікації критичних точок.

Таким чином, ідентифікація критичних точок - це процес аналізу, дослідження та визначення тих елементів програми, які найбільше впливають на її продуктивність та надійність, і визначення їх як потенційних місць для подальшого дослідження та виправлення bottlenecks.

Не менш важливими є використання результатів таких рішень. Тому, правильність у прийнятті рішень внаслідок знаходження критичних точок bottlenecks дозволяє попереджати небезпечні наслідки. Здійснення адекватних заходів після виявлення проблем дозволяє запобігти небезпечним наслідкам та забезпечити більш ефективну роботу програмного забезпечення та інформаційних систем, яке полягає в наступному:

- управлінню ресурсами на вирішення саме тих проблем, які найбільше впливають на продуктивність або надійність програмних модулів інформаційної системи;

- створенню стратегій оптимізації, які вирішують виявлені проблеми та спрямовують зусилля на пріоритетні аспекти у прийнятті подальших рішень;

- попередження подальших проблем в наслідок на ідентифіковані критичні точки, що в значній мірі може запобігти подальшим проблемам чи збоїв у майбутньому;

- покращення продуктивності та надійності, що полягає у правильності в прийнятті рішень, що засновані на аналізі критичних точок та дозволяє підвищити ефективність та стабільність системи.

Завдяки тому, що існують різні стадії розвитку bottlenecks, виділяють різні шляхи щодо прийняття рішень стосовно їх виявлення. Такі різні стадії розвитку bottlenecks можуть вимагати різних підходів до прийняття рішень для їх виявлення та вирішення. Залежно від того, на якій стадії знаходиться bottleneck, можуть застосовуватися різні стратегії, які полягають в наступному.

У випадку, якщо ще на стадії розробки програмного забезпечення відомо, що певні компоненти можуть потенційно стати bottlenecks, можна застосувати превентивні заходи. Це може включати в себе вибір більш ефективних алгоритмів, оптимізацію коду або використання більш потужного обладнання.

Моніторинг у реальному часі також є досить важливим чинником, який полягає в тому, що під час роботи системи важливо постійно моніторити її продуктивність. Тому, щоб вчасно виявляти bottlenecks, які можуть виникнути під впливом різних умов або навантажень необхідно виконувати спостереження за процесами обробки інформації. Це в значній мірі дозволить оперативно реагувати на проблеми.

Аналіз ідентифікованих bottlenecks дозволяє після виявлення bottlenecks провести глибокий аналіз, щоб зрозуміти причини та вплив на систему появи таких вузьких місць продуктивності. Це в повній мірі допомогає визначити оптимальні стратегії вирішення поточних проблем.

Оптимізація та реакція на зміни у продуктивності системи виконується після виправлення або оптимізації системи. Це є важливим заходом так як дозволяє слідкувати за функціонуванням програмних модулів інформаційної системи. Іноді bottlenecks можуть змінюватися під впливом нових умов, тому реагувати на зміни вчасно є критично важливим.

Таким чином, різні етапи розвитку bottlenecks потребують різних стратегій та підходів. Відповідний підхід на кожній стадії дозволяє ефективно виявляти, аналізувати та вирішувати проблеми, що забезпечує стабільність та ефективність системи.

Перші ознаки bottlenecks, до яких відносяться поява багатьох потоків або процесів в межах виконання однієї задачі повинні носити інформаційний та одноразовий характер у прийнятті рішень системою сповіщень.

У таких випадках важливо вчасно виявляти та реагувати на ці сигнали, щоб уникнути подальших проблем. Тому, у таких сигналів виділяють різних характер:

- інформаційний характер, який сповіщає про те, що в системі вже існує багато потоків чи процесів, що можуть слугувати попередженням про можливе виникнення bottlenecks. Це може бути основним індикатором того, що певна частина системи працює з надмірним навантаженням, де можливе настання майбутніх проблем;

- одноразовий характер, який полягає в тому, що у деяких випадках поява багатьох потоків чи процесів може бути тимчасовим явищем, яке виникає в результаті пікового навантаження чи тимчасової нестабільності системи. Однак це також може бути початковим сигналом про можливі майбутні проблеми, якщо такі умови будуть постійними;

- характер системних сповіщень, яке полягає в тому, що виявлення багатьох ознак настання bottlenecks може бути частиною системи сповіщень, яка автоматично виявляє незвичайні події або показники у роботі системи. Це дозволяє оперативно реагувати та вживати заходів для попередження потенційних проблем.

Таким чином, загальна стратегія полягає в тому, щоб система могла ідентифікувати та аналізувати ці ознаки, сповіщати про них та вживати відповідних заходів. Це дозволить забезпечити більшу стабільність та ефективність системи у майбутньому.

Подальший розвиток bottlenecks, який характеризується збільшенням навантаження на ресурси обчислювальної системи повинен забезпечуватись рішучими діями в галузі прийняття рішення. Тому, збільшення навантаження на ресурси обчислювальної системи, що є наступною стадією розвитку bottlenecks, потребує рішучих дій та прийняття відповідних рішень. У таких ситуаціях часто застосовують засоби горизонтального масштабування, що дозволяють розширювати можливості системи шляхом додавання нових ресурсів або збільшення кількості серверів чи обчислювальних вузлів.

Горизонтальне масштабування - це стратегія, коли система розширюється шляхом додавання нових ресурсів на рівні обладнання, серверів або рабочих станцій. Наприклад, при цьому може використовуватися розподілена архітектура, де завдання розподіляються між кількома серверами для підвищення продуктивності та здатності системи обробляти навантаження. Також, проблему збільшеного навантаження на ресурси обчислювальної системи може вирішуватись за допомогою кластеризації.

Кластерізація - це використання кластерів серверів для підтримки більшої кількості користувачів або обробки великих обсягів даних. Кластеризація дозволяє розділити завдання між декількома вузлами для швидшої обробки та збільшення продуктивності.

До ще одних ефективних шляхів вирішення проблем збільшених навантажень відносять технології балансування навантажень та автоматизації масштабування. Балансування навантаження виконується для застосування методів балансування навантаження для рівномірного розподілу завдань між різними серверами або вузлами. Це допомагає уникнути перевантаження окремих компонентів та забезпечує оптимальне використання ресурсів.

Автоматизація масштабування - це використання технологій, інструментів та систем автоматизованого масштабування, які можуть автоматично реагувати на збільшення навантаження, розширюючи ресурси системи при потребі.

Таким чином, підтримка постійного моніторингу для виявлення росту навантаження та реагування на нього шляхом використання даних аналізу продуктивності та обсягу навантаження на систему та горизонтальне масштабування надає можливість системі реагувати на зростаюче навантаження, забезпечуючи більшу потужність та продуктивність для подальшого розвитку і підтримки високої ефективності. При цьому, досить дієвим механізмом для вирішення цієї проблеми також є розробка ефективних алгоритмів попередження розвитку bottlenecks у програмному забезпеченні в інформаційних системах включає наступні кроки (рис. 2.3).

Крок 1. Початковий моніторинг та аналіз продуктивності, який включає в себе систематичну перевірку роботи окремних програмних модулів згідно поставленої меті та задачам для виявлення проблемних ділянок коду та аналізу їх продуктивності, що може допомогти ідентифікувати потенційні bottlenecks.

Крок 2. Якщо перевірка зібраних даних на поставлену мету та задачі досліджень ідентифікації потенційних bottlenecks не відповідає заданим значенням, то виконується перехід на крок 1. В іншому випадку, переходять на крок 3.

Крок 3. Виконання профілювання коду, що включає в себе використання інструментів для аналізу роботи програми під час її виконання для виявлення місць зайвої складності, непотрібних операцій або неефективних алгоритмів.

Крок 3. Стрес-тестування, яке спрямоване на створення сценаріїв, які демонструють максимальне навантаження на систему для виявлення слабких місць у програмному забезпеченні.

Крок 4. Прогнозування проведення подальших обсягів роботи, що включає в себе аналіз тенденцій та попереднє передбачення потреб у ресурсах для забезпечення якісного функціонування системи.

Крок 5. Оптимізація алгоритмів та архітектури, що спрямоване на покращення ефективності шляхом оптимізації коду, використання більш швидких алгоритмів та удосконалення архітектури системи.

Крок 6. Резервне планування, яке включає в себе розробку плану дій для вирішення проблем, які можуть виникнути у випадку розвитку bottlenecks.

В цьому випадку, система сповіщень про негативні наслідки в розвитку bottlenecks повинна мати структурні особливості, що дозволяють ефективно виявляти, аналізувати та реагувати на проблеми у системі. Наведемо деякі структурні особливості такої системи сповіщень.

Використання різних сенсорів та механізмів моніторингу для постійного відстеження показників продуктивності та ресурсів загальної структури є необхідною умовою для впровадження системи оповіщення. Це може включати моніторинг завантаження процесора, використання пам'яті, часу відгуку та інших метрик продуктивності.

Також, система відстеження показників продуктивності повинна включати інструменти для аналізу та виявлення відхилень в показниках, які можуть вказувати на потенційні bottlenecks або збільшення навантаження на систему. Такі автоматизовані аналізатори дозволяють оперативно реагувати на проблеми.

В системі відстеження показників продуктивності також можуть бути конфігураційні параметри небезпечних показників апаратних та програмних метрик для сповіщень, коли значення показників перевищують встановлені межі або коли спостерігається раптовий зріст навантаження на систему.

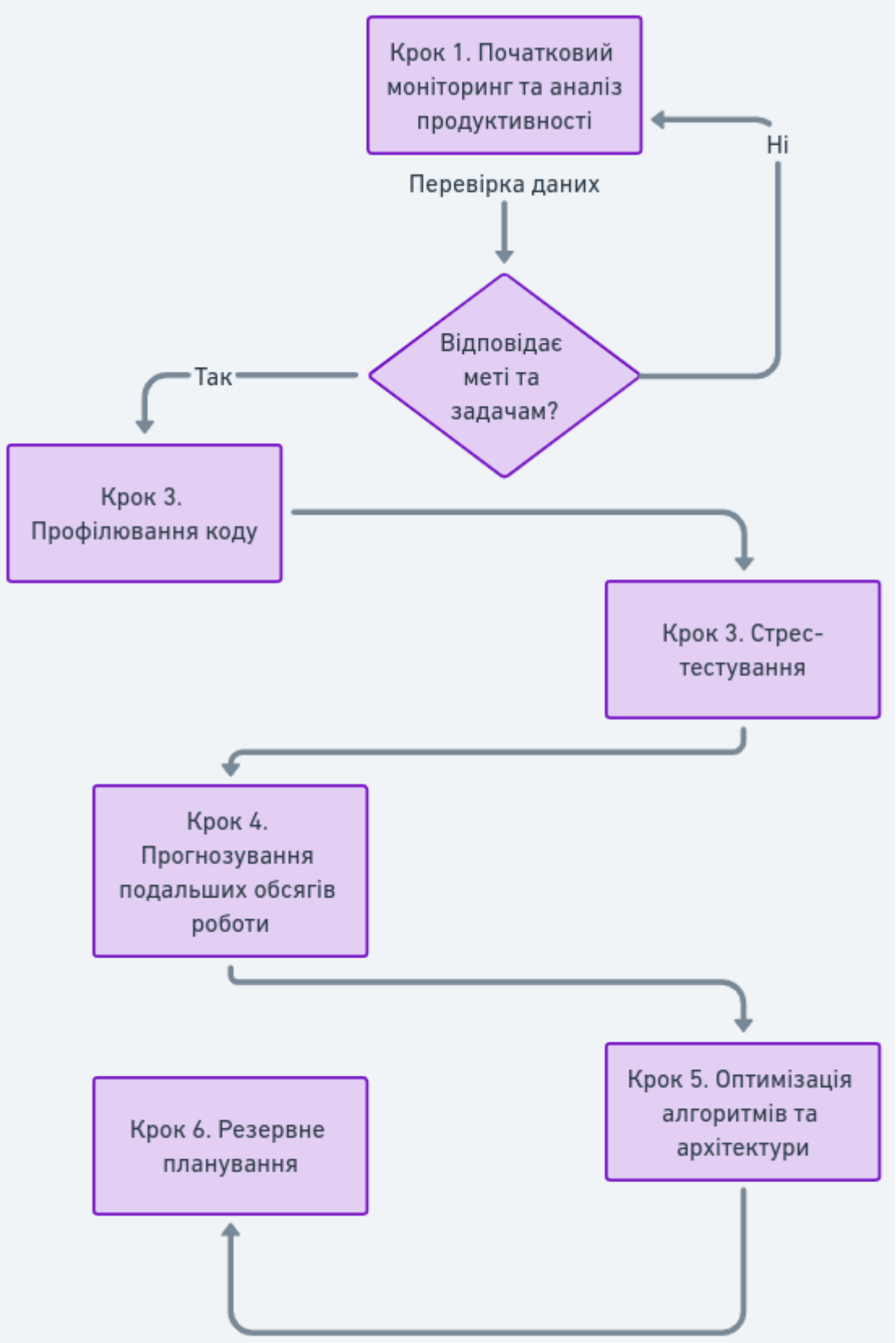


Рисунок 2.2 - Діаграма прикладу дослідження розвитку bottlenecks у програмному забезпеченні

Механізми для забезпечення відповідних реакцій щодо небезпечних дій інших програмних засобів повинні інформувати відповідних адміністраторів або відповідальних осіб про виявлені проблеми, а також активувати автоматичні дії для вирішення чи подальшого аналізу причин спрацювання сповіщень.

Також, система сповіщень про негативні наслідки в розвитку bottlenecks повинна мати такі функціональні особливості, що дозволяють ефективно виявляти, аналізувати та реагувати на проблеми в системі.

До таких функціональних особливостей належать:

- здатність системи оперативно реагувати на негативні наслідки та надсилати сповіщення в реальному часі, щоб відразу ж звернутися до проблеми;

- можливість автоматичного генерування сповіщень і повідомлень у випадку виявлення аномальних показників або негативних наслідків в роботі системи;

- можливість налаштування параметрів сповіщень, таких як тип повідомлення, рівень важливості, отримувачі, час доставки тощо, для кращого узгодження з потребами користувача;

- здатність програмного забезпечення об'єднувати сповіщення та повідомлення з різних джерел чи систем для уникнення спаму та покращення сприйняття інформації;

- можливість програмного засобу відслідковувати стан оброблених та необроблених сповіщень, а також зберігання історії сповіщень для подальшого аналізу та вирішення проблем;

- можливість встановлення пріоритетів для сповіщень, щоб відповідно реагувати на найважливіші проблеми та забезпечувати швидку реакцію;

- інтеграцію з іншими автоматичними системами або скриптами, що дозволяє виконувати автоматизовані дії для локалізації або вирішення виявлених проблем.

Ці функціональні особливості дозволяють системі сповіщень ефективно виявляти, передавати та реагувати на негативні наслідки в розвитку bottlenecks, що сприяє швидкому вирішенню проблем та підтримує стабільність системи.

Реалізація вищезгаданих структурних та фунціональних особливостей відбувається наступним чином. Спочатку визначається мета експерименту. Це може бути виявлення та виправлення конкретних bottlenecks у програмі або оцінка впливу оптимізаційних заходів на продуктивність системи.

При цьому, створюється тестове середовище, яке відповідає умовам реальної роботи системи. Це може включати в себе підготовку тестових даних, налаштування параметрів середовища та встановлення інструментів для моніторингу та профілювання. Тому, здійснюються тести та експерименти в тестовому середовищі. Це може включати в себе виконання стандартних операцій, симуляцію навантаження або інші дії, які характерні для реальної роботи системи.

Під час експериментів збираються дані про продуктивність, використання ресурсів (час виконання, використання пам'яті тощо), а також дані про помилки та інші події, які виникають під час експерименту. Отримані дані аналізуються для виявлення проблем, bottlenecks, та ідентифікації їх джерела. Це допомагає визначити, які частини програми або системи є критичними та потребують оптимізації.

На основі результатів аналізу розробляються стратегії вирішення виявлених проблем. Це може включати в себе оптимізацію програмного коду, зміну архітектури, використання більш швидких алгоритмів тощо.
Внесені зміни перевіряються в тестовому середовищі, і проводяться додаткові експерименти для підтвердження виправлення проблем і покращення продуктивності системи. Усі результати експериментів і аналізу документуються. Це допомагає стежити за здійсненими змінами та їх впливом на програму.

Експерименти та аналіз є невід'ємною частиною методу діагностики bottlenecks і дозволяють ідентифікувати, аналізувати та виправляти проблеми у програмному забезпеченні інформаційних систем з метою покращення їх продуктивності та надійності.

Після виявлення bottlenecks, розробляються та впроваджуються оптимізаційні заходи. Це може включати в себе оптимізацію алгоритмів, використання кешування, виправлення помилок та інші заходи для покращення продуктивності та надійності програми.

Оптимізація і виправлення - це фінальний крок у методі діагностики bottlenecks у програмному забезпеченні інформаційних систем, після виявлення та аналізу проблем. Цей процес включає в себе заходи для вирішення ідентифікованих проблем і покращення продуктивності та надійності програми. На основі результатів діагностики і аналізу проблем визначаються пріоритети для оптимізації. Ідентифікуються найкритичніші bottlenecks, які негативно впливають на продуктивність або надійність системи.

При цьому, розробляються стратегії для вирішення кожної із виявлених проблем. Це може включати в себе оптимізацію програмного коду, зміну архітектури, виправлення помилок або використання більш ефективних алгоритмів.

Після внесення змін проводиться тестування для переконанняся в тому, що внесені зміни вирішили проблеми та не спричинили нових помилок. Потім, виконується верифікується, чи продуктивність та надійність системи покращилися. Далі, після внесення змін і тестування проводиться повторний аналіз для підтвердження вирішення проблем та оцінки досягнутих покращень. Усі внесені зміни та результати тестування документуються для подальшого використання та відстеження. Після впровадження оптимізацій слідкується за продуктивністю системи в реальному середовищі, щоб переконатися, що вона функціонує без проблем та надійно.

Процес оптимізації та виправлення не завершується одноразово. Він повинен бути частиною постійного циклу підтримки і покращення програми, оскільки нові проблеми можуть виникати під час розвитку системи. Оптимізація і виправлення допомагають вирішити виявлені проблеми і покращити продуктивність та надійність програмного забезпечення інформаційної системи.

Моніторинг та підтримка є важливою частиною методу діагностики bottlenecks у програмному забезпеченні інформаційних систем, оскільки дозволяють постійно слідкувати за продуктивністю та надійністю системи та реагувати на виникаючі проблеми.

Спочатку встановлюються інструменти для моніторингу, які дозволяють збирати дані про роботу системи в режимі реального часу. Ці інструменти можуть включати в себе системи журналювання, моніторингу ресурсів, аналізу логів та інші. Система постійно моніториться на предмет виникнення проблем та буттленеків. Це може включати в себе слідкування за завантаженням CPU, використанням пам'яті, часом відгуку та іншими показниками продуктивності.

Моніторингові інструменти допомагають виявити будь-які проблеми або буттленеки, що можуть виникнути в системі. Наприклад, система може сповістити про надмірне використання пам'яті або зростання часу відгуку.

В системі можуть бути налаштовані автоматичні реакції на виникнення проблем. Наприклад, система може автоматично перезавантажити компонент, який викликає bottlenecks, або відправити повідомлення адміністраторам.

Моніторинг і підтримка - це постійний процес. Інструменти моніторингу оновлюються та адаптуються до змін в системі, і вони постійно підтримуються для ефективного виявлення проблем. Інформація, що зібрана під час моніторингу, використовується для аналізу продуктивності та надійності системи. Якщо виявляються проблеми, проводиться оптимізація та виправлення.

Моніторинг та підтримка грають важливу роль у методі діагностики bottlenecks, дозволяючи підтримувати оптимальну продуктивність та надійність програмного забезпечення інформаційної системи в реальному часі та реагувати на виникаючі проблеми.

Таким чином, у роботі використовувалися методи дослідження bottlenecks у програмному забезпеченні інформаційних систем, були засновані на експертному визначенні критичних точок та встановлення обробників.

Даний метод заснований на встановлення та знищення обробників у програмному коді автоматизованої системи, що керується за допомогою

конфігураційних файлів на етапі компіляції та виконання програм. Такі обробників у програмному коді автоматизованої системи надають можливість визначати час виконання певних ділянок програмного коду та визначати неприпустимі дії виконання певних функцій (наприклад, ділення на нуль). Це дозволяє підвищити продуктивність та надійність у роботі програмного забезпечення в інформаційних системах як виробничої експлуатації.

Таким чином, розглянутий метод використання системи сповіщень про негативні наслідки розвитку bottlenecks, заснований на отриманні повідомлень про критичні точки та може дозволяти ефективно приймати рішення у сфері горизонтального масштабування обчислювальних ресурсів, зокрема на вузлах розподіленої системи. У порівнянні з відомими аналогами це є важливим кроком у покращенні продуктивності та стабільності системи.

В той же час, отримання повідомлень про критичні точки системи, що можуть вказувати на виникнення або потенційні ризики bottlenecks, дозволяє вчасно реагувати на ці проблеми. Використання системи сповіщень дозволяє автоматизувати процес виявлення, сповіщення та реагування на виникнення таких критичних точок.

При цьому, з'являється можливість адміністраторам системи або відповідальним особам оперативно приймати рішення щодо масштабування обчислювальних ресурсів, наприклад, шляхом додавання нових вузлів у розподілену систему або реорганізації поточних вузлів для оптимізації продуктивності та запобігання затримкам у роботі системи.

Такий підхід надає можливість підтримувати ефективність системи на високому рівні, шляхом оперативного реагування на потенційні проблеми, що забезпечує стабільність та надійність її роботи.

Поряд з цим, в роботі запропоновано подальший розвиток у механізмах моніторингу використання обчислювальних та інформаційних ресурсів включає збір даних про реальне виконання програмних модулів у робочому середовищі. Ці дані використовуються для відправлення попереджувальних повідомлень до

системи сповіщень з метою виявлення небезпечного розвитку навантажувальних впливів.

Це означає, що система моніторингу здатна в реальному часі відслідковувати роботу програмних модулів, аналізувати використання ресурсів та спрогнозувати можливі проблеми у виконанні. Коли виявляються певні аномалії або надмірне навантаження, ця система надсилає попереджувальні повідомлення до системи сповіщень.

Використання такого підходу дозволяє не лише моніторити ресурси, але й передбачати можливі проблеми з використанням цих ресурсів у майбутньому. Вчасне виявлення навантажувальних впливів дозволяє системі реагувати на них, запобігаючи затримкам у роботі та забезпечуючи стабільність та ефективність системи.

2.3. Вимоги щодо програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Для розробки програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях, необхідно враховувати наступні вимоги (рис. 2.3). Програмний засіб повинен бути спроможним працювати на різних обчислювальних платформах, таких як CPU, GPU, FPGA та інші спеціалізовані апаратні пристрої, які використовуються у високопродуктивних обчисленнях (табл. 2.2).

Вимога щодо "Підтримки різних обчислювальних платформ" передбачає, що ваш програмний засіб для дослідження bottlenecks повинен бути спроможним працювати на різних видів обчислювальних платформ і апаратних пристроях.

Ця вимога є дуже важливою в контексті високопродуктивних обчислень, оскільки такі обчислення можуть використовувати різні типи апаратних пристроїв для виконання обчислень.

Основні аспекти цієї вимоги включають наступні компоненти. Підтримка різних архітектур, що передбачає те , що програмний засіб повинен бути здатним

оптимізовано працювати на різних архітектурах обчислювальних пристроїв, таких як x86, ARM, CUDA (для GPU), FPGA і так далі.

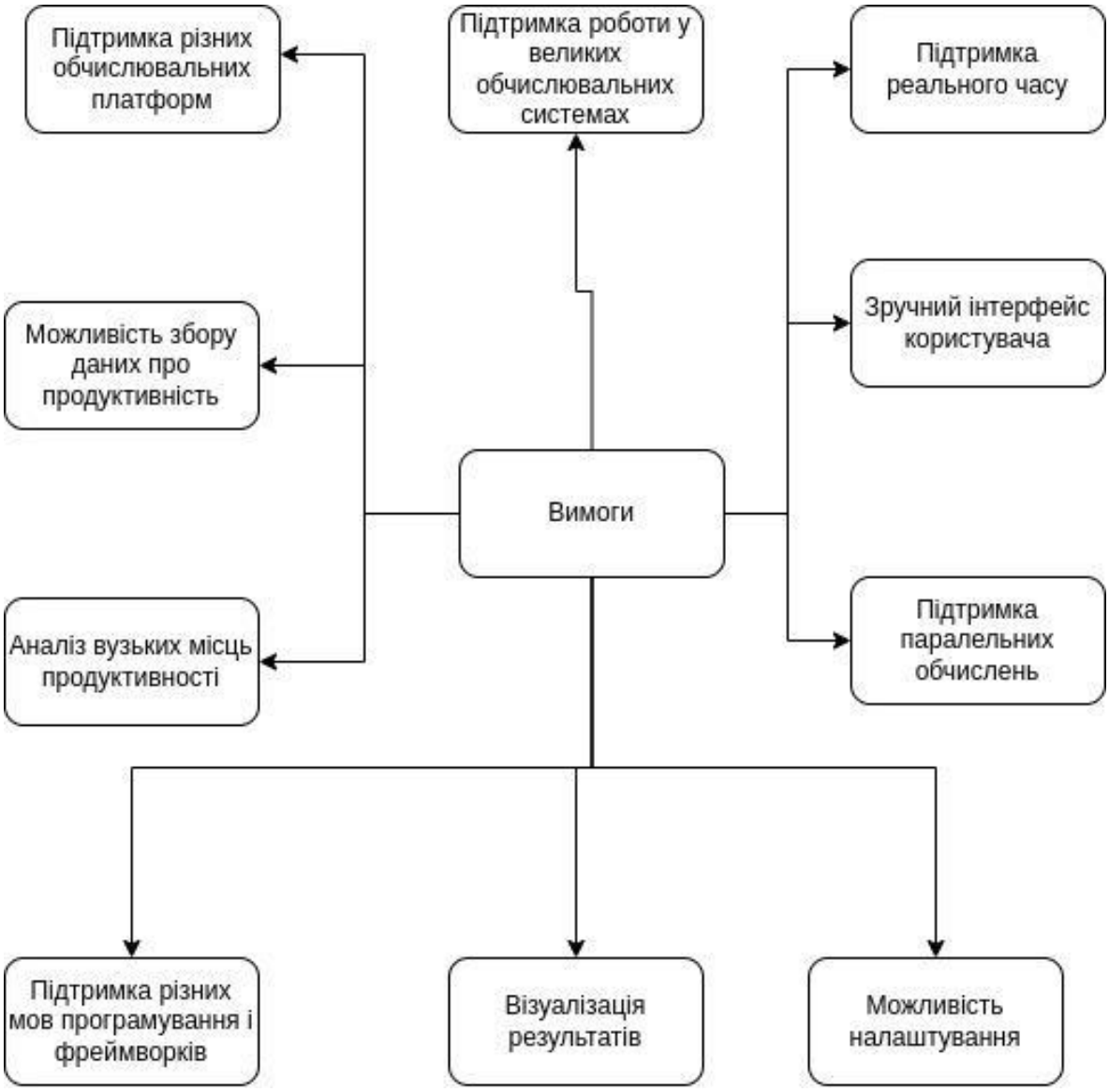


Рисунок 2.3 - Вимоги щодо розробки програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Підтримка різних виробників має бути сумісним з обчислювальними пристроями від різних виробників, таких як Intel, AMD, NVIDIA, Xilinx і інші.

Таблиця 2.2

Вимоги щодо програмного засобу для дослідження bottlenecks у високопродуктивних обчисленнях

Вимога	Пояснення
Підтримка різних обчислювальних платформ	Програмний засіб повинен бути здатним працювати на різних типах обчислювальних платформ, таких як CPU, GPU, FPGA тощо.
Можливість збору даних про продуктивність	Потрібно мати можливість збирати дані про продуктивність, такі як час виконання, використання ресурсів і обсяги пам'яті.
Аналіз bottlenecks	Програмний засіб має надавати можливість аналізувати зібрані дані і виявляти вузькі місця продуктивності в коді чи обчислювальній системі.
Підтримка різних мов програмування і фреймворків	Програмний засіб повинен бути сумісним з різними мовами програмування та фреймворками, що використовуються в обчисленнях.
Візуалізація результатів	Має бути можливість візуалізувати зібрані дані про продуктивність у зрозумілому інтерфейсі, наприклад, у вигляді графіків чи діаграм.
Можливість налаштування	Користувач повинен мати можливість налаштовувати параметри збору даних та аналізу для відповідності конкретним потребам дослідження.
Підтримка паралельних обчислень	Засіб повинен бути здатним виявляти проблеми, пов'язані з паралельним виконанням коду, що використовується у високопродуктивних системах.
Зручний інтерфейс користувача	Програмний інтерфейс повинен бути зрозумілим і зручним для користувача для налаштування, візуалізації та аналізу даних.
Підтримка реального часу	Якщо потрібно аналізувати продуктивність в реальному часі, програмний засіб повинен бути готовий до моніторингу та аналізу в реальному часі.
Підтримка роботи у великих обчислювальних системах	Засіб повинен бути здатним працювати у великих обчислювальних кластерах або хмарних середовищах, якщо це необхідно.

Для досягнення максимальної продуктивності, програмний засіб може вимагати оптимізацій, які специфічні для конкретного типу пристрою. Наприклад, оптимізації для виконання на CPU можуть відрізнятися від оптимізацій для GPU.

Автоматичне визначення пристрою визначає те, що програмний засіб може бути здатним автоматично визначати доступні обчислювальні ресурси на системі і використовувати їх ефективно. Наприклад, він може перевіряти доступні GPU та вибирати найпотужніший для обчислень.

Кросплатформеність забезпечує широку доступність інструменту, програмний засіб може бути розроблений так, щоб працювати на різних операційних системах, таких як Windows, Linux, а також на хмарних обчислювальних платформах.

Сумісність з бібліотеками та фреймворками необхідний для враховування сумісності з популярними бібліотеками та фреймворками для високопродуктивних обчислень, такими як TensorFlow, PyTorch, OpenCL, і багатьма іншими.

Забезпечення підтримки різних обчислювальних платформ дозволить користувачам вашого програмного засобу максимально використовувати їх наявні ресурси і обирати оптимальні апаратні рішення для своїх завдань у високопродуктивних обчисленнях.

Програмний засіб повинен здатний збирати важливі дані про продуктивність системи, такі як час виконання, використання ресурсів, обсяги пам'яті тощо. Він повинен надавати можливість аналізувати зібрані дані і виявляти вузькі місця продуктивності, тобто ті частини програми або обчислювальної системи, де витрати часу або ресурсів є надмірними.

Підтримка різних мов програмування і фреймворків визначає те, що засіб має бути універсальним і здатним працювати з різними мовами програмування та фреймворками, що використовуються в обчисленнях.

Для зручності аналізу результатів, програмний засіб повинен надавати можливість візуалізації даних, наприклад, у вигляді графіків чи діаграм.

Користувач повинен мати можливість налаштовувати параметри збору даних та аналізу для відповідності конкретним потребам дослідження.

Оскільки високопродуктивні обчислення часто використовують паралельне програмування, засіб повинен бути здатним виявляти проблеми, пов'язані з паралельним виконанням коду.

При цьому, програмний засіб повинен мати зручний інтерфейс користувача для налаштування, візуалізації результатів та аналізу даних.

Підтримка програмного часу у режимі реального часу необхідна, якщо потрібно аналізувати продуктивність в реальному часі, засіб повинен бути здатним до моніторингу та аналізу в реальному часі. Якщо плануєтьтся використовувати засіб у великих обчислювальних кластерах або в хмарних середовищах, він повинен бути готовий до такого роду роботи.

Таким чином, розробка програмного засобу для дослідження bottlenecks у високопродуктивних обчисленнях вимагає спеціалізованих знань у сфері обчислювальних технологій і високопродуктивних обчислень, а також досвіду у розробці програмного забезпечення для моніторингу та аналізу продуктивності.

2.4. Структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Основний програмний засіб для дослідження bottlenecks в високопродуктивних обчисленнях може бути розділений на різні модулі та компоненти для виконання різних завдань. Наведемо деякі зі структурно-функціональними особливостями та властивостями основних модулів такого програмного засобу (рис. 2.4).

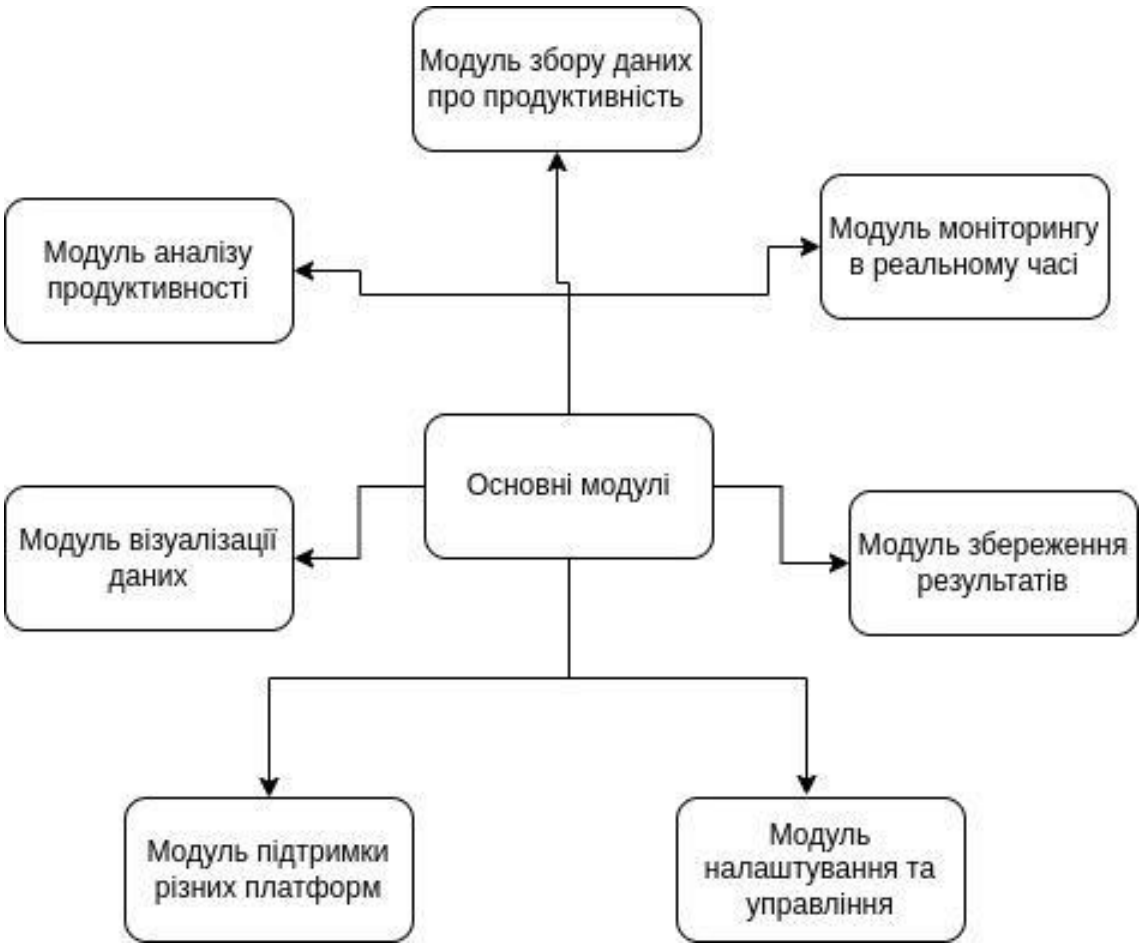


Рисунок 2.4 - Структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Модуль збору даних про продуктивність є одним з ключових компонентів програмного засобу для дослідження bottlenecks у високопродуктивних обчисленнях (табл. 2.3). Його головна функціональність полягає в зборі різних видів даних про продуктивність, які необхідні для подальшого аналізу та ідентифікації проблем з продуктивністю. Функціональність модуля збору даних про продуктивність може бути наступною.

Модуль може вимірювати час, який займає виконання конкретних операцій, функцій чи завдань. Це включає в себе час виконання всієї програми або окремих її частин. Моніторинг використання ресурсів полягає у здатності вимірювати використання ресурсів, таких як процесорний час, обсяги пам'яті,

мережевий обсяг і т.д. Це дозволяє виявити, які ресурси витрачаються під час виконання програми.

Збір даних про робоче навантаження може бути забезпечено за допомогою функціональності записувати дані про навантаження на систему, такі як кількість одночасно активних потоків, обсяги даних, що обробляються тощо. Збір статистики в реальному часі необхідно проводити, якщо модуль може підтримувати режим збору даних у реальному часі, дозволяючи користувачам моніторити продуктивність системи в реальному часі. Модуль може надавати можливість докладніше розділяти зібрані дані, включаючи інформацію про окремі функції, методи, модулі чи компоненти програми.

До властивостей модуля збору даних про продуктивність відноситься наступне (табл. 2.4). Модуль повинен дозволяти користувачам налаштовувати параметри збору даних, такі як інтервали вимірювань, обсяг даних, які збираються тощо.

Сам по собі модуль повинен мінімізувати вплив на продуктивність системи, оскільки надмірний збір даних може спотворювати результати аналізу.

Зібрані дані повинні бути збережені в безпечному місці для подальшого аналізу та звітування. Можливість зберігання в різних форматах може бути корисною.

Модуль повинен бути здатним працювати на різних обчислювальних платформах та операційних системах.

Автоматичний старт та зупинка збору даних надає можливість автоматичного запуску та зупинки збору даних може бути корисною для підтримки специфічних сценаріїв використання.

Для збереження конфіденційності даних, зібраних модулем, може бути важливим включити заходи забезпечення.

Таким чином, модуль збору даних про продуктивність є важливим компонентом для забезпечення точності та повноти аналізу продуктивності високопродуктивних обчислень. Його налаштування та робота повинні враховувати конкретні потреби та обмеження користувачів програмного засобу.

Таблиця 2.3

Основна функціональність модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Модуль	Функціональність
Модуль збору даних про продуктивність	Збирає дані про продуктивність, включаючи час виконання, використання ресурсів та обсяги пам'яті.
Модуль аналізу продуктивності	Обробляє та аналізує зібрані дані для виявлення bottlenecks, надає результати аналізу та виділяє проблемні області.
Модуль візуалізації даних	Надає можливість візуалізувати результати аналізу у зрозумілому графічному вигляді, такому як графіки, діаграми, графи тощо.
Модуль підтримки різних платформ	Забезпечує підтримку різних обчислювальних платформ, автоматично визначає доступні ресурси та вибирає оптимальну платформу.
Модуль налаштування та управління	Дозволяє користувачам налаштовувати параметри збору даних, аналізу, візуалізації та інші налаштування програмного засобу.
Модуль збереження результатів	Забезпечує збереження результатів аналізу та даних про продуктивність для подальшого використання чи обміну з іншими користувачами.
Модуль моніторингу в реальному часі	Надає можливість моніторити продуктивність в реальному часі під час виконання програми чи обчислювального завдання.

Модуль аналізу продуктивності є ключовим компонентом програмного засобу для дослідження bottlenecks в високопродуктивних обчисленнях. Цей модуль відповідає за обробку та аналіз зібраних даних про продуктивність для виявлення проблемних областей та оптимізації програм або систем. Функціональність модуля аналізу продуктивності має наступні параметри.

Аналіз часу виконання модуля полягає в тому, що він здатен аналізувати дані про час виконання операцій, функцій або завдань. Він може ідентифікувати, які частини програми вимагають найбільше часу для виконання.

Таблиця 2.4

Основні властивості модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Модуль	Властивості
Модуль збору даних про продуктивність	Можливість збору даних в реальному часі або за запитом користувача. Точність та надійність вимірювань.
Модуль аналізу продуктивності	Використання алгоритмів та методів аналізу, таких як профілювання і трасування викликів.
Модуль візуалізації даних	Параметризоване налаштування візуалізації та інтерактивні можливості для дослідження даних.
Модуль підтримки різних платформ	Сумісність з різними архітектурами та виробниками обчислювальних пристроїв.
Модуль налаштування та управління	Інтерфейс користувача для зручного управління програмним засобом та налаштування параметрів.
Модуль збереження результатів	Підтримка різних форматів збереження даних та можливість експорту результатів.
Модуль моніторингу в реальному часі	Інтерфейс моніторингу для негайного виявлення проблем та можливість коригування параметрів в реальному часі.

Виявлення bottlenecks у модулі визначає фрагменти програми, які викликають затримки чи витрати ресурсів і можуть бути потенційними вузькими місцями.

Аналіз використання ресурсів у модулі може аналізувати використання ресурсів, таких як CPU, пам'ять, диск та інші, для визначення, де саме відбуваються інтенсивні операції. В разі недостатньої кількості ресурсів для задачі модуль може виявити такі ситуації та рекомендувати оптимізаційні заходи.

Генерація звітів та рекомендацій полягає у створені за допомогою модуля звіти та рекомендації для користувачів, вказуючи на місця, які потребують оптимізації та пропонуючи конкретні заходи.

Підтримка різних методів аналізу може використовувати різні методи аналізу, такі як профілювання коду, трасування викликів, аналіз графів залежностей, щоб отримати докладну інформацію про продуктивність.

Властивості модуля аналізу продуктивності полягають в наступному. Модуль повинен дозволяти користувачам налаштовувати параметри аналізу, такі як рівень деталізації та методи, що використовуються. Інтерактивний інтерфейс модуля може надавати інтерактивний інтерфейс для дослідження результатів аналізу та виправлення помилок.

Підтримка багаторівневого аналізу полягає в тому, що модуль може визначити вузькі місця на різних рівнях абстракції, від окремих функцій до загальної архітектури системи. При цьому, під час візуалізації результатів мається можливість візуалізації даних та результатів аналізу може полегшити сприйняття інформації та виявлення проблем.

Модуль повинен бути сумісним з різними мовами програмування та платформами розробки. Для збереження конфіденційності дані, що аналізуються модулем, можуть бути захищені.

Таким чином, модуль аналізу продуктивності є важливим інструментом для виявлення та усунення проблем з продуктивністю в програмах і системах для високопродуктивних обчислень. Він дозволяє розробникам та інженерам отримувати інформацію, необхідну для оптимізації роботи програм та забезпечення ефективної роботи систем.

Модуль візуалізації даних є також важливим компонентом програмного засобу для дослідження bottlenecks в високопродуктивних обчисленнях. Він відповідає за перетворення зібраних даних про продуктивність у зрозумілий і графічний вигляд, що дозволяє користувачам легше сприймати і аналізувати інформацію. Функціональність модуля візуалізації даних полягає в наступному.

Графічна репрезентація даних дозволяє перетворити зібрані дані про продуктивність у графічну форму, таку як графіки, діаграми, графи, графіки обмежень тощо.

Підтримка різних типів даних полягає в тому, що модуль може візуалізувати різні види даних, включаючи числові дані, часові ряди, діаграми розподілу, графи залежностей тощо.

Модуль дозволяє користувачам налаштовувати параметри візуалізації, такі як вибір типу графіка, шкали, колірну палітру тощо. Він також може візуалізувати одночасно дані з різних джерел чи експериментів для порівняння результатів. Візуалізація може бути інтерактивною, що дозволяє користувачам навігувати, масштабувати, вибирати та взаємодіяти з графічними елементами.

Властивості модуля візуалізації даних є наступні. Розширені можливості візуалізації, де модуль може підтримувати різні типи візуалізації, включаючи лінійні графіки, кругові діаграми, гістограми, теплові карти, графи тощо.

Сумісність з іншими інструментами полягає в інтеграції з іншими інструментами для аналізу даних або навіть для автоматизованої оптимізації на основі результатів візуалізації.

Модуль дозволяє зберігати графіки та візуалізації у різних форматах (наприклад, зображення, PDF) та експортувати їх для подальшого використання чи обміну з іншими користувачами. Також, модуль може бути адаптованим до різних розмірів екрану та пристроїв, щоб забезпечити оптимальний вигляд візуалізацій на різних пристроях.

Модуль може надавати інтерфейс користувача для налаштування та інтерактивного взаємодії з візуалізацією.

Таким чином, модуль візуалізації даних важливий для того, щоб робити дані про продуктивність більш зрозумілими та доступними для аналізу. Це допомагає інженерам та розробникам отримувати важливий інсайт в продуктивність своїх програм та систем і приймати на цій основі рішення для їх оптимізації.

Модуль підтримки різних платформ є важливим компонентом програмного засобу для дослідження bottlenecks в високопродуктивних обчисленнях. Його основна функція - забезпечити сумісність та ефективну роботу програмного засобу на різних обчислювальних платформах та

архітектурах. Функціональність модуля підтримки різних платформ полягає в наступному. Модуль може визначати, які обчислювальні ресурси доступні на конкретній платформі, включаючи кількість процесорів, обсяг пам'яті, наявність графічних прискорювачів тощо.

Автоматичний вибір оптимальної платформи полягає в тому, що модуль може автоматично вибирати найбільш підходящу обчислювальну платформу для виконання задачі, враховуючи обсяг обчислювальних ресурсів та інші параметри.

Управління конфігураціями модуля дозволяє налаштовувати параметри програми для оптимальної роботи на конкретній платформі. Це може включати в себе налаштування паралельного виконання, оптимізацію коду та інші параметри. Підтримка різних операційних систем необхідно для збереження сумісності з різними операційними системами, такими як Windows, Linux, macOS та іншими.

Робота на хмарних платформах дозволяє модулю підтримувати роботу на хмарних обчислювальних платформах, що дозволяє використовувати ресурси хмарних серверів.

Сумісність з різними мовами програмування необхідна для здатності модуля підтримувати різні мови програмування та технології розробки.

Відлагодження та профілювання на різних платформах дозволяє модулю надавати засоби для відлагодження та профілювання програми на різних платформах для виявлення проблем з продуктивністю.

До властивостей модуля підтримки різних платформ відносяться наступні. Модуль може автоматично визначати, на якій платформі виконується програма, і відповідно налаштовувати її параметри. Модуль може складатись з окремих частин, щоб дозволити легке додавання підтримки для нових платформ у майбутньому.

Сумісність з різними архітектурами полягає в тому, що модуль повинен бути здатним працювати на різних архітектурах процесорів, включаючи x86, ARM, PowerPC тощо.

Для збереження конфіденційності даних, зокрема, якщо програма працює на хмарних платформах, можуть бути важливі заходи забезпечення. Легка інтеграція з іншими модулями дозволяє легко інтегруватися з іншими компонентами програмного засобу для дослідження bottlenecks.

Таким чином, модуль підтримки різних платформ допомагає забезпечити гнучкість та розширюваність програмного засобу, дозволяючи йому працювати на різних обчислювальних платформах та забезпечувати оптимальну продуктивність на кожній з них.

Модуль налаштування та управління є важливим компонентом програмного засобу для дослідження bottlenecks в високопродуктивних обчисленнях. Цей модуль відповідає за конфігурацію, налаштування та керування різними аспектами роботи програми або системи для оптимізації продуктивності та виявлення вузьких місць. Функціональність модуля налаштування та управління є наступною.

Настройка параметрів продуктивності модуля дозволяє користувачам налаштовувати параметри, які впливають на продуктивність програми, такі як рівень паралельності, обсяг пам'яті, режими оптимізації тощо.

Планування завдань і ресурсів допомогає планувати роботу програми, включаючи розподіл завдань між різними обчислювальними ресурсами та контроль за їх виконанням. Динамічне налаштування модуля може підтримувати різне налаштування, що дозволяє змінювати параметри під час роботи програми в залежності від обчислювальних умов та завдань.

Модуль може здійснювати контроль за використанням обчислювальних ресурсів, включаючи виділення, звільнення та реалокацію ресурсів. При цьому, модуль може вести журнал подій та стану програми, що допомагає відстежувати роботу та виявляти проблеми.

До властивостей модуля налаштування та управління відносяться наступні. Гнучкість налаштувань, де модуль повинен надавати широкі можливості для налаштування параметрів, щоб користувачі могли адаптувати програму до різних обчислювальних умов.

Автоматичне управління модуля може включати автоматичні механізми управління ресурсами та параметрами продуктивності, які реагують на зміни у середовищі виконання. Для збереження конфіденційності та безпеки даних, що обробляються програмою, може бути важливим включити заходи забезпечення.

Модуль може дозволяти зберігати налаштування та конфігураційні файли для подальшого використання чи резервного копіювання.

Інтерфейс користувача модуля надає графічний інтерфейс для зручного налаштування та управління програмою.

Таким чином, модуль налаштування та управління є важливим інструментом для забезпечення оптимальної продуктивності та ефективного використання обчислювальних ресурсів у високопродуктивних обчисленнях. Він дозволяє користувачам впливати на роботу програми під час її виконання і вирішувати проблеми, які можуть виникнути в залежності від обчислювального середовища.

Модуль збереження результатів є також важливим компонентом програмного засобу для дослідження bottlenecks в високопродуктивних обчисленнях. Основна функція цього модуля - забезпечити можливість збереження та обробки результатів аналізу та експериментів для подальшого аналізу, спільного використання та референсування. До особливостей та властивостей цього модуля відносяться наступні. Збереження результатів аналізу, де модуль дозволяє зберігати результати аналізу продуктивності, включаючи дані про вимірювання часу, структури даних, звіти про продуктивність і т. д.

Управління результатами модуля надає можливість створювати, видаляти, перейменовувати та організовувати файли з результатами, щоб зручно вести облік експериментів та досліджень. Експорт та імпорт результатів роботи модуля дозволяє експортувати та імпортувати результати аналізу для обміну з іншими користувачами або іншими програмами. Є також можливість візуалізації результатів, наприклад, за допомогою графіків, діаграм або таблиць, є важливою частиною цього модуля.

Модуль може включати заходи для захисту конфіденційності даних, що зберігаються в результаті аналізу.

До властивості модуля збереження результатів відносяться наступні. Модуль може організовувати результати аналізу у вигляді структури каталогів та файлів для зручного пошуку та управління. При цьому, модуль може зберігати історію результатів аналізу, щоб користувачі могли повертатися до попередніх експериментів та порівнювати результати.

Автоматичне збереження полягає в тому, що модуль може автоматично зберігати результати аналізу після кожного експерименту або вимірювання. Для збереження безпеки та обмеження доступу до результатів можуть бути введені механізми авторизації та аутентифікації.

Модуль може зберігати метадані, такі як параметри експерименту, дати та часи проведення аналізу, для додаткового контексту та пояснень результатів.

Сумісність з різними форматами даних дозволяє модулю бути сумісним з різними форматами файлів, що використовуються для збереження результатів (наприклад, CSV, JSON, XML, SQLite тощо).

Таким чином, модуль збереження результатів грає важливу роль у забезпеченні коректного та систематичного збереження результатів аналізу та експериментів. Це дозволяє дослідникам та інженерам ефективно працювати з отриманими даними, а також забезпечує можливість архівування та обміну результатами з іншими користувачами чи дослідницькими групами.

Модуль моніторингу в реальному часі є важливим компонентом програмного засобу для дослідження bottlenecks в високопродуктивних обчисленнях. Його основна функція - надавати інформацію про роботу програми або системи в режимі реального часу, щоб дозволити користувачам спостерігати за продуктивністю та ефективністю в реальному часі.

Наведемо функціональність модуля моніторингу в реальному часі. Збір та відображення даних про продуктивність полягає в тому, що модуль може збирати дані про роботу програми в режимі реального часу, такі як використання процесора, пам'яті, дисків, мережі тощо, і відображати цю інформацію на

графіках або в текстовому вигляді. Моніторинг системних параметрів дозволяє модулю слідкувати за системними параметрами, такими як температура процесора, стан жорстких дисків, використання ресурсів інших програм, що працюють на комп'ютері тощо.

Модуль може виявляти аномалії в роботі програми або системи, такі як перевищення певних ресурсів, і надсилати повідомлення або сповіщення користувачам. Модуль є інтерактивним, що дозволяє користувачам здійснювати дії, такі як зупинка або перезавантаження програми, зміна параметрів в реальному часі тощо.

Збереження історії моніторингу допомогає модулю зберігати історію даних про продуктивність, щоб користувачі могли аналізувати динаміку роботи програми та виявляти тенденції в часі.

Властивості модуля моніторингу в реальному часі є наступною. Низький вплив на продуктивність полягає в тому, що модуль повинен мати низький вплив на продуктивність самої системи, щоб не спричиняти перевантаження обчислювальних ресурсів. При цьому, модуль дозволяє налаштовувати параметри моніторингу, включаючи обсяг збираної інформації та частоту збору даних. Модуль має графічний інтерфейс для відображення даних в зручній формі, такій як графіки, діаграми, графи тощо, де модуль може надавати можливість налаштовувати сповіщення або аварійні сигнали при виявленні критичних аномалій. Модуль має механізми захисту доступу до даних моніторингу для забезпечення конфіденційності.

Модуль моніторингу в реальному часі допомагає користувачам відслідковувати та аналізувати роботу програми або системи в реальному часі, що дозволяє вчасно реагувати на проблеми та оптимізувати продуктивність. Цей модуль особливо корисний для моніторингу та управління високопродуктивними обчисленнями, де ефективність є критичною.

Таким чином, розглянуті модулі мають спільно працювати для створення комплексного програмного засобу для дослідження bottlenecks у

високопродуктивних обчисленнях, що дозволяє користувачам ефективно виявляти та вирішувати проблеми з продуктивністю своїх програм і систем.

2.5. Висновки

В другому розділі розглянуто загальна характеристика методів дослідження bottlenecks у високопродуктивних обчисленнях. Досліджено особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях, яка полягала в тому, щоб знайти та виправити "вузькі місця" або обмеження, які гальмують продуктивність, тим самим забезпечити кращу ефективність використання обчислювальних ресурсів. Запропонований метод діагностики bottlenecks у програмному засобі, який полягає в встановлені та знищені обробників у програмному коді автоматизованої системи, що керується за допомогою конфігураційних файлів на етапі компіляції та виконання програм. Такі обробників у програмному коді автоматизованої системи надають можливість визначати час виконання певних ділянок програмного коду та визначати неприпустимі дії виконання певних функцій (наприклад, ділення на нуль). Це дозволяє підвищити продуктивність та надійність у роботі програмного забезпечення в інформаційних системах як виробничої експлуатації.

Також, визначені вимоги щодо функціонування програмного засобу та розроблені структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.

РОЗДІЛ 3.
РЕАЛІЗАЦІЯ МЕТОДУ ТА ПРОГРАМНОГО ЗАСОБУ

3.1. Розробка вхідних параметрів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Вхідні параметри для обробки даних у програмному засобі для дослідження bottlenecks у високопродуктивних обчисленнях можуть бути дуже різноманітними і залежать від конкретних завдань дослідження. Розглянемо основні вхідні параметри більш ретельно (табл. 3.1):

Таблиця 3.1

Вхідні дані програмного засобу дослідження bottlenecks

Вхідні дані	Опис
Код програми або обчислювальний граф	Вихідний код програми або структура обчислювального графа, що досліджується.
Вхідні дані	Дані, які використовуються програмою під час виконання, наприклад, вхідні файли.
Параметри запуску	Параметри, які вказуються при запуску програми або обчислення (командний рядок).
Дані про обчислювальну інфраструктуру	Інформація про обчислювальну інфраструктуру, таку як кількість процесорів, пам'ять.
Метрики продуктивності	Метрики для вимірювання продуктивності, наприклад, час виконання, завантаження CPU.
Алгоритми аналізу та параметри	Алгоритми та параметри для аналізу даних і виявлення bottlenecks.
Параметри експерименту	Параметри, що визначають налаштування експериментів, такі як обсяги даних, змінні.
Дані відладки і журнали подій	Дані, які допомагають відстежувати та аналізувати виконання програми чи системи.
Параметри збору даних про продуктивність	Параметри модуля збору даних про продуктивність, такі як інтервали і типи метрик.

Код програми або обчислювальний граф описує вихідний код програми або обчислювальний граф, який досліджуєтьтся. Цей код може містити обчислювальні завдання, алгоритми та конфігураційні налаштування.

Код програми або обчислювальний граф є одним із найважливіших вхідних параметрів для програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях. Ця частина вхідних даних визначає структуру та логіку обчислень, які досліджуються. Цей параметр представляє собою вихідний код обчислювальної програми або структуру обчислювального графа, яка виконується. Важливо враховувати його наступні аспекти.

Мова програмування, яка може бути у вигляді кода, що написаний на певній мові програмування, такій як Python, C++, Java, або інших. Мова програмування визначає синтаксис, можливості та інші характеристики коду.

Структура програми або графа може бути організована у вигляді окремих функцій, класів або інших структур. Обчислювальний граф визначає послідовність та зв'язки між обчислювальними вузлами.

Алгоритми та обчислювальні завдання, що містить алгоритми та обчислювальні завдання, які виконуються. Це може бути математичні обчислення, обробка даних, обчислення графів, інтегрування тощо.

Конфігураційні параметри може містити параметри, які визначають конфігурацію обчислення, такі як розмір блоків для паралельних обчислень, кількість потоків, параметри апаратного забезпечення тощо.

Залежності та будь-яка інтеграція, що означає якщо програмний засіб включає в себе кілька обчислювальних модулів або взаємодіє з іншими системами, код також містить залежності та інтеграційні точки.

Реалізація алгоритмів оптимізації використовується якщо досліджується оптимізація продуктивності, то код може містити реалізацію алгоритмів оптимізації, таких як розпаралелювання, підготовка даних, оптимізація пам'яті тощо. В деяких випадках код може вимагати додаткових ресурсів або вхідних даних, які також включаються в вхідні параметри.

Враховуючи всі ці аспекти, програмний засіб для дослідження bottlenecks може аналізувати виконання коду, вимірювати час виконання, моніторити ресурси та ідентифікувати можливі вузькі місця та оптимізаційні можливості.

Вхідні дані можуть бути представлені як дані, які використовуються програмою або системою під час виконання. Це може бути великий обсяг даних, параметри налаштування, вхідні файли, масиви даних тощо.

Параметри запуску - це параметри, які вказуються при запуску програми або обчислення. Наприклад, це можуть бути параметри командного рядка, конфігураційні файли або налаштування обчислювального середовища.

Параметри запуску є важливою частиною вхідних даних для програмного засобу дослідження bottlenecks в високопродуктивних обчисленнях. Ці параметри визначають конфігурацію та спосіб запуску програми або обчислення. Розглянемо їх докладніше.

Параметри запуску можуть бути у вигляді командного рядку та являти собою набір аргументів, які передаються програмі через командний рядок під час запуску.

Наприклад, це може бути команда вигляду:

python script.py --param1 value1 --param2 value2,

де --param1 та --param2 - це параметри запуску, а value1 та value2 - їхні значення.

Програмний засіб може підтримувати конфігураційні файли, в яких зберігаються параметри запуску. Це зручний спосіб налаштування програми без зміни командного рядка.

Деякі програми можуть читати параметри змінних середовища операційної системи. Наприклад, ви можете встановити змінні середовища, які впливають на поведінку програми.

Параметри ресурсів можуть використовуватись у вигляді параметрів, які визначають доступ до обчислювальних ресурсів, таких як кількість процесорів, обсяг оперативної пам'яті, кількість графічних карток і т. д.

Параметри також можуть вказувати режим роботи програми та вхідні дані, які програма повинна обробляти. Наприклад, режим розробки або виробництва.

Завдання для паралельних обчислень можуть виконуватися якщо програма підтримує паралельні обчислення, параметри запуску можуть включати налаштування для розподілу завдань між потоками чи процесами.

Деякі програми можуть вимагати параметрів безпеки, таких як ключі для шифрування чи автентифікації. Параметри можуть визначати, як програма має здійснювати логування подій і створювати журнали для аналізу. Залежно від програми та її функцій, можуть існувати специфічні параметри запуску.

Параметри запуску визначають, як саме програма або обчислення будуть виконуватися, і вони можуть бути важливими для дослідження продуктивності, оцінки впливу на різні конфігурації та виявлення можливих проблем в ході виконання.

Інформація про обчислювальну інфраструктуру, на якій виконуються обчислення, така як кількість процесорів, обсяг оперативної пам'яті, типи графічних карток, мережева конфігурація може здійснюватися наступним чином. Дані про обчислювальну інфраструктуру - це інформація про обладнання та ресурси, на яких виконуються обчислення в програмному засобі для дослідження bottlenecks у високопродуктивних обчисленнях. Ця інформація є важливою для оцінки та аналізу продуктивності, а також для прийняття рішень щодо оптимізації та налаштування обчислювального середовища. Розглянемо ці дані більш докладно.

Кількість процесорів (CPU) визначає кількість фізичних та/або логічних ядер процесорів на обчислювальній машині.

Типи процесорів позначає інформацію про моделі та характеристики процесорів, такі як виробник, архітектура, тактова частота, кеш-пам'ять.

Оперативна пам'ять (RAM) показує обсяг доступної оперативної пам'яті на обчислювальній машині.

Система зберігання даних (наприклад, SSD або HDD) вказує на тип та обсяг пристроїв для зберігання даних.

Графічні процесори (GPU) позначають інформацію про наявність та характеристики графічних процесорів, якщо вони використовуються для обчислень.

Мережева конфігурація вказує на мережеву архітектуру та характеристики мережевого обладнання, включаючи швидкість передачі даних.

Операційна система та версія показує інформацію про операційну систему, яка використовується на обчислювальній машині, та її версію.

Специфікації мережі вказує на швидкість мережі, наявність фаєрволів, конфігурація мережі, стан з'єднань.

Інші ресурси можуть показувати ресурси, які можуть впливати на продуктивність, такі як доступ до спеціалізованих обчислювальних пристроїв (наприклад, FPGA або TPU).

Ці дані про обчислювальну інфраструктуру дозволяють програмному засобу дослідження bottlenecks налагоджувати та адаптуватися до конкретного обчислювального середовища. Вони також допомагають ідентифікувати можливі джерела обмежень продуктивності та шукати способи їх вирішення.

Метрики продуктивності є ключовою складовою для вимірювання та оцінки ефективності обчислювальних процесів у програмному засобі для дослідження bottlenecks у високопродуктивних обчисленнях. Вони дозволяють кількісно оцінювати різні аспекти продуктивності програми або системи. Розглянемо деякі основні метрики продуктивності та їх значення.

Час виконання (Execution Time) описує час, який потрібний для виконання обчислення або програми від початку до завершення. Низький час виконання вказує на високу продуктивність. Збільшення часу виконання може свідчити про можливі проблеми в продуктивності.

Час відгуку (Response Time) описує час, який витрачається на обробку та відповідь на запит користувача або системи. Низький час відгуку важливий для реактивних систем і покращення користувацького досвіду.

Завантаження процесора (CPU Utilization) описує відсоток часу, протягом якого процесор використовується для обчислень. Високе завантаження

процесора може вказувати на інтенсивні обчислення, але також може бути вузьким місцем, якщо досягає 100%.

Використання пам'яті (Memory Utilization) описує відсоток доступної оперативної пам'яті, яка використовується обчисленнями. Високе використання пам'яті може призвести до переповнення і впливати на продуктивність.

Трафік мережі (Network Traffic) описує обсяг даних, які передаються через мережу під час обчислень або обміну даними. Великий трафік мережі може призвести до затримок та впливати на продуктивність мережевих додатків.

Частота кадрів (Frame Rate) описує кількість кадрів або об'єктів, які відображаються на екрані за одну секунду. Висока частота кадрів важлива для графічних додатків і відеоігор.

Кількість операцій (Operations per Second) описує кількість обчислювальних операцій, які виконуються за одну секунду. Висока кількість операцій на секунду може свідчити про продуктивність програми.

Метрики пам'яті (Memory Metrics) описують детальні метрики використання пам'яті, такі як обсяг використаної пам'яті, кеш-пам'ять, витрати на алокування пам'яті тощо.

Метрики пам'яті допомагають ідентифікувати проблеми з витратою пам'яті та оптимізувати її використання.

Це показані лише деякі приклади метрик продуктивності. Вибір конкретних метрик залежить від типу дослідження та цілей вимірювань. Метрики продуктивності допомагають визначити, чи відповідає програма або система очікуваному рівню продуктивності і чи потребує вона оптимізації.

Алгоритми аналізу та параметри аналізу є важливими складовими програмного засобу для дослідження bottlenecks у високопродуктивних обчисленнях. Вони дозволяють системі визначити та оцінити чинники, які впливають на продуктивність обчислення або програми. Розглянемо їх більш докладно.

Профілювання коду (Code Profiling) описує це метод аналізу, який вимірює, скільки часу програма витрачає на виконання кожної функції або рядка

коду. Включає в себе вимірювання часу виконання, кількості викликів функцій, використання пам'яті тощо.

Аналіз графа залежностей (Dependency Graph Analysis) описує алгоритм, що створює граф залежностей між функціями та ресурсами в програмі. Він показує, як дані та обчислення взаємодіють між собою. Такий аналыз відображає структуру програми та можливі місця, де можна виявити вузькі місця.

Аналіз витрат пам'яті (Memory Usage Analysis) описує аналіз використання оперативної пам'яті під час виконання програми. При цьому, вимірюються розмір та типи об'єктів, які використовують пам'ять. Це допомагає виявити проблеми з витратою пам'яті та зменшити їх.

Режим аналізу (Analysis Mode) описує який тип аналізу буде використовуватися, наприклад, аналіз продуктивності, аналіз пам'яті, аналіз мережі і т. д. Вибір правильного режиму аналізу відповідає цілям дослідження.

Період аналізу (Analysis Period) описує на якому етапі виконання програми або обчислення буде проводитися аналіз. Важливо обрати момент чи період, коли аналіз найбільш репрезентативний.

Метрики аналізу (Analysis Metrics) описує які конкретні метрики будуть вимірюватися під час аналізу, такі як час виконання, використання пам'яті, завантаження процесора, тощо. Це дозволяє зосередитися на конкретних аспектах продуктивності.

Спосіб збору даних (Data Collection Method) описує яким чином будуть зібрані дані для аналізу, наприклад, інструменти профілювання, інструменти моніторингу системи тощо. Тому, важливо обрати інструменти, які надають необхідну інформацію.

Параметри фільтрації (Filtering Parameters) описує які дані або об'єкти слід включити чи виключити з аналізу, може бути застосований для скорочення обсягу даних. Це дозволяє сконцентруватися на конкретних аспектах аналізу. Ці алгоритми аналізу та параметри аналізу допомагають збирати та обробляти дані для оцінки продуктивності, виявлення вузьких місць та вдосконалення

програмного засобу або обчислювальної системи. Вони можуть варіюватися в залежності від конкретних цілей дослідження та типу аналізу, який виконується.

Параметри експерименту необхідні якщо проводиться серія експериментів, то параметри цих експериментів, такі як змінні налаштування, обсяги даних, різні конфігурації тощо.

Параметри експерименту - це конкретні налаштування та умови, які використовуються під час проведення дослідження для вимірювання продуктивності або оцінки певних характеристик системи або програми. Вони грають важливу роль у забезпеченні об'єктивності та достовірності експерименту. Давайте розглянемо докладніше параметри експерименту:

Вхідні дані (Input Data) описують які дані будуть використовуватися під час експерименту, і як ці дані будуть згенеровані або вибрані. Вони можуть бути синтетичними або реальними даними, залежно від цілей експерименту. Важливо обрати вхідні дані, які відповідають реальним сценаріям використання системи або програми.

Обчислювальні ресурси (Computational Resources) описують конфігурацію обчислювального середовища, таку як кількість процесорів, обсяг оперативної пам'яті, типи графічних процесорів тощо. Це дозволяє встановити, наскільки ресурси можуть вплинути на продуктивність та дозволяє проводити аналіз в різних обчислювальних умовах.

Параметри виконання (Execution Parameters) описують налаштування, які визначають, як саме програма або обчислення будуть запущені, такі як команди запуску, аргументи командного рядка, режими виконання тощо. При цьому, важливо встановити параметри виконання, які відповідають реальному використанню системи або програми.

Метрики продуктивності (Performance Metrics) описують які конкретні метрики продуктивності будуть вимірюватися під час експерименту, такі як час виконання, використання пам'яті, завантаження процесора, швидкість мережі тощо. Це дозволяє визначити, як саме ефективності та продуктивність будуть вимірюватися.

Розмір вибірки (Sample Size) описує кількість ітерацій або виконань експерименту для одного конкретного налаштування. Більший розмір вибірки зазвичай дозволяє отримати більш точні результати. Це визначається відповідно до статистичних вимог експерименту.

Повторюваність (Reproducibility) описує наскільки експеримент може бути відтворений у майбутньому, включаючи можливість ідентичного відтворення у різних умовах. Це важливо для підтвердження результатів та виявлення можливих варіацій.

Параметри аналізу (Analysis Parameters) включає в себе конкретні налаштування для алгоритмів аналізу, які використовуються для оцінки продуктивності. Це впливає на спосіб вимірювання та аналізу продуктивності.

Умови середовища (Environment Conditions) вказуєть будь-яку додаткову інформація про середовище, в якому відбувається експеримент, таку як температура, вологість, відстань до сервера тощо. Це допомагає враховувати зовнішні умови, які можуть вплинути на результати експерименту.

Параметри експерименту допомагають забезпечити консистентність та об'єктивність у проведенні дослідження продуктивності. Вони визначають, як саме експеримент буде виконаний і які результати можна отримати. Важливо добре документувати всі параметри експерименту для можливості повторення і порівняння результатів.

Дані відладки і журнали подій вказують про події, журнали відладки та інші дані, які допомагають відстежувати та аналізувати виконання програми чи системи. Дані відладки і журнали подій є важливими джерелами інформації під час дослідження bottlenecks у високопродуктивних обчисленнях. Вони допомагають відстежувати виконання програми або системи, виявляти помилки та проблеми, а також аналізувати їх вплив на продуктивність. Розглянемо їх більш докладно.

Журнали стеку викликів (Call Stack Logs) вказують послідовність функцій та методів, які були викликані в програмі, а також контекст виклику кожної

функції. Це допомагає виявити, яким чином виконується програма та визначити місця, де можуть виникати затримки або проблеми.

Відомості про змінні (Variable Information) містять дані про значення змінних та об'єктів під час виконання програми. Це дозволяє виявити, які дані обробляються та чи можливі помилки у їхньому обробленні.

Помилки та винятки (Errors and Exceptions) описують записи про помилки, які виникають під час виконання програми, разом зі стеком викликів, допоміжними повідомленнями та інформацією про помилку. Це допомагає виявити та усунути помилки, які можуть впливати на продуктивність.

Журнали подій (Event Logs) фіксують події та дії, які відбуваються в програмі, включаючи запуск функцій, завершення операцій та інші важливі події. Це допомагає відстежувати послідовність подій та реакцію програми на вхідні дані.

Час події (Event Timestamp) показує час, коли подія сталася. Це важливо для визначення часових затримок та послідовності подій.

Тип події (Event Type) вказує на природу події, наприклад, запуск функції, завершення операції, помилка тощо. Це дозволяє фільтрувати та групувати події за їхнім типом.

Ідентифікатор події (Event ID) описує унікальний ідентифікатор для кожної події. Це допомагає відстежувати та виокремлювати окремі події для аналізу.

Додаткові дані (Additional Data) включає додаткові параметри, аргументи, стан системи тощо, пов'язані з подією. Це надає деталізовану інформацію про події та дозволяє аналізувати їх контекст.

Повідомлення про помилку (Error Messages) включає повідомлення про помилки та стек викликів, пов'язаний з помилкою. Це допомагає ідентифікувати та усувати помилки в системі.

Запити та відповіді мережевих запитів (Network Requests and Responses) фіксує деталі мережевих запитів, такі як URL, час відправлення та отримання

запиту, статус відповіді тощо. Це дозволяє виявити проблеми з мережевими запитами та їх вплив на продуктивність.

Дані відладки і журнали подій є важливим інструментом для аналізу продуктивності та виявлення вузьких місць у високопродуктивних обчисленнях. Вони дозволяють розуміти, як програма або система веде себе під час виконання та виявляти можливі проблеми.

Параметри збору даних про продуктивність виконуються якщо програмний засіб має модуль збору даних про продуктивність, то параметри цього модуля, такі як інтервали збору даних, типи метрик, обсяги даних тощо. Параметри збору даних про продуктивність визначають, яку інформацію та які метрики слід збирати під час дослідження bottlenecks у високопродуктивних обчисленнях. Ці параметри грають ключову роль у вимірюванні продуктивності та аналізі системи чи програми. Розглянемо їх більш докладно.

Метрики продуктивності (Performance Metrics) визначають, які конкретні параметри продуктивності слід вимірювати під час експерименту. Це може включати час виконання, використання CPU, використання пам'яті, швидкість введення-виведення (I/O), пропускну здатність мережі і багато інших. Це дозволяє визначити конкретні характеристики продуктивності, які слід аналізувати.

Інтервали вимірювання (Measurement Intervals) вказують, як часто будуть вимірюватися метрики продуктивності, наприклад, кожну секунду, кожну мілісекунду, після кожної операції тощо. Це впливає на точність та обсяг зібраних даних.

Додаткові параметри вимірювання (Additional Measurement Parameters) вожуть включати додаткову інформацію для вимірювання, таку як вимірювачі температури, вологості, рівня шуму тощо. Це важливі для контекстуалізації даних продуктивності та аналізу їхнього впливу.

Спосіб збору даних (Data Collection Method) визначає, як саме будуть зібрані дані про продуктивність, включаючи інструменти профілювання, інструменти моніторингу системи, інструменти аналізу тощо. Це важливо

вибрати інструменти, які надають необхідну інформацію та не впливають негативно на продуктивність самого експерименту.

Обсяг даних (Data Volume) визначає, скільки даних буде збиратися під час експерименту, наприклад, кількість вимірювань, кількість записів тощо. Це впливає на обсяг та обробку даних, які потрібно зберігати та аналізувати.

Агрегація даних (Data Aggregation) визначає, яким чином зібрані дані будуть агреговані або оброблятися, наприклад, середнє значення, медіана, максимальне та мінімальне значення тощо. Це дозволяє отримати загальну картину продуктивності на основі великої кількості даних.

Збереження даних (Data Storage) вказує, де та як будуть зберігатися зібрані дані, включаючи формати файлів, бази даних або спеціалізовані системи для зберігання даних про продуктивність. Це важливо для зберігання та аналізу зібраних даних.

Запуск та завершення збору даних (Start and Stop Data Collection) вказує момент початку та завершення збору даних під час експерименту. Це визначає, яким чином дані будуть пов'язані з виконанням програми або системи.

Управління вибіркою (Sampling Control) визначає, чи будуть дані збиратися для всього обсягу виконання програми або для певного підмножини операцій. Це дозволяє сконцентруватися на основних областях дослідження продуктивності.

Ці параметри збору даних про продуктивність допомагають визначити, яким чином і яку інформацію збирати для аналізу продуктивності системи або програми. Вони грають важливу роль у вимірюванні та розумінні продуктивності та вузьких місць.

Розглянуті вхідні параметри можуть бути використані для аналізу продуктивності, виявлення вузьких місць та оптимізації обчислень у високопродуктивних обчисленнях.

3.2. Розробка вихідних параметрів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях

Вихідні параметри для обробки даних у програмному засобі дослідження bottlenecks у високопродуктивних обчисленнях зазвичай включають в себе результати аналізу продуктивності та іншу інформацію, яка може бути корисною для подальшого вдосконалення системи чи програми (табл. 3.2). Опишемо деякі з можливих вихідних параметрів.

Звіти про продуктивність (Performance Reports) є одним із основних вихідних параметрів для обробки даних у програмному засобі дослідження bottlenecks у високопродуктивних обчисленнях. Ці звіти надають детальну інформацію про продуктивність програми чи системи під час експерименту та аналізують ключові метрики продуктивності. Більш докладний опис параметрів, які можуть бути включені до звітів про продуктивність є наступними.

Метрики продуктивності (Performance Metrics) є перелік ключових метрик, які були виміряні під час експерименту. Це може включати час виконання, завантаження CPU, використання пам'яті, швидкість введення-виведення (I/O), пропускну здатність мережі та інші важливі характеристики продуктивності. Це надає інформацію про те, які аспекти продуктивності були виміряні та аналізовані.

Графіки і діаграми (Charts and Graphs) описує візуальне представлення результатів вимірювань у вигляді графіків, діаграм або інших візуальних засобів. Це допомагає наглядно відображати зміни продуктивності з часом або порівнювати різні експерименти.

Таблиця 3.2

Вихідні параметри для обробки даних у програмному засобі дослідження bottlenecks у високопродуктивних обчисленнях

Параметр	Опис
Звіти про продуктивність	Результати вимірювань метрик продуктивності, графіки та діаграми.
Специфікації вузьких	Інформація про місця, де виявлені вузькі місця

місць	продуктивності та їх фактори.
Рекомендації щодо оптимізації	Список рекомендацій для вдосконалення продуктивності та ефективності.
Деталі аналізу	Додаткові дані та висновки про фактори, що впливають на продуктивність.
Звіти про споживану ресурсами	Інформація про використання ресурсів, таких як CPU, пам'ять, диск і мережу.
Журнали подій та відладки	Записи подій, виявлені помилки та діагностичні дані для подальшого аналізу та відлагодження.
Дані про продуктивність на різних платформах	Інформація про продуктивність на різних обчислювальних платформах.
Інша статистика та метрики	Додаткові числові значення, що вказують на різні аспекти продуктивності, такі як середнє значення, стандартне відхилення тощо.

Таблиці результатів (Result Tables) описує таблиці, що містять числові значення метрик продуктивності для різних вимірювань чи умов. Це дозволяє докладно аналізувати дані та проводити порівняння між різними експериментами.

Середні значення та статистика (Averages and Statistics) описує розраховані середні значення, стандартні відхилення та інші статистичні показники для метрик продуктивності. Це надає загальну картину продуктивності та розподілу значень метрик.

Аналіз відхилень (Deviation Analysis) описує визначення відхилень або аномалій у продуктивності, які можуть свідчити про проблеми або несправності. Це допомагає виявити незвичайні ситуації, що впливають на продуктивність.

Порівняння з базовою лінією (Baseline Comparison) описує порівняння продуктивності з еталонною або базовою лінією, яка служить відповідною точкою для оцінки змін.

Застосування: Виявлення змін у продуктивності в порівнянні з попередніми вимірюваннями.

Схеми розподілу (Distribution Plots) описує графічне представлення розподілу значень метрик продуктивності, таких як гістограми чи ящики з вусами. Це допомагає з'ясувати, як розподілені дані та чи є вони нормальними.

Таким чином, звіти про продуктивність є важливою складовою результатів дослідження та аналізу продуктивності. Вони дозволяють користувачам та інженерам отримати чітку картину профілю продуктивності програми чи системи, виявити проблеми та приймати рішення щодо подальших дій для оптимізації.

Специфікації вузьких місць (Bottleneck Specifications) - це інформація про точні місця, де виявлені вузькі місця продуктивності. Специфікації вузьких місць - це документація або інформація, яка надає докладний опис місць в системі чи програмі, де виявлені вузькі місця продуктивності. Вони вказують на конкретні частини коду, операції, алгоритми, або ресурси, які стали обмежувачем для ефективності та вказують на фактори, що впливають на ці вузькі місця. Наведемо докладний опис параметрів, які можуть бути включені до специфікацій вузьких місць є наступними.

Локація вузького місця (Bottleneck Location) вказує точне місце, де було виявлено вузьке місце продуктивності. Це може бути конкретний файл, функція, кодовий блок чи алгоритм. Це допомагає розмірковувати про можливі зміни чи оптимізації в цій області.

Опис вузького місця (Bottleneck Description) - це подробний опис того, як саме виявлене вузьке місце впливає на продуктивність і які операції чи ресурси затримуються. Це допомагає розуміти природу проблеми та можливі шляхи вирішення.

Фактори вузького місця (Bottleneck Factors) є вказівка на ті чинники, які призвели до виникнення вузького місця. Це може бути об'єм даних, обсяг обчислень, доступ до зовнішніх ресурсів і т. д. Це допомагає розуміти, які фактори слід враховувати при вирішенні проблеми.

Посилання на вихідні коди (Source Code References) - це вказівка на рядки коду, які стосуються вузького місця. Це може бути посилання на конкретні рядки

коду, класи, методи тощо. Воно допомагає розробникам знаходити та аналізувати код, пов'язаний з вузьким місцем.

Вплив на продуктивність (Impact on Performance) - це вказівка на те, як вузьке місце впливає на продуктивність системи або програми, наприклад, збільшення часу виконання. Це допомагає визначити, наскільки критично вузьке місце для загальної продуктивності.

Потенційні варіанти вирішення (Possible Solutions) - це пропозиції або ідеї щодо того, як можна виправити вузьке місце або оптимізувати його. Це може включати в себе рекомендації щодо змін коду, алгоритмів, оптимізації ресурсів і т. д. Це допомагає визначити можливі шляхи вирішення проблеми.

Пріоритет (Priority) позначає те, наскільки важливо вирішити вузьке місце в порівнянні з іншими завданнями. Такий параметр може бути високий, середній, низький пріоритет тощо. Це допомагає приймати рішення про те, як спрямувати ресурси для оптимізації.

Специфікації вузьких місць є важливою інформацією для розробників та інженерів, оскільки вони надають чітку картину проблеми та вказують на шляхи для її вирішення. Ця документація є корисною під час розробки плану оптимізації та вдосконалення продуктивності системи чи програми.

Рекомендації щодо оптимізації (Optimization Recommendations) - це частина вихідних даних програмного засобу дослідження bottlenecks, яка містить рекомендації та поради щодо покращення продуктивності системи чи програми. Ці рекомендації надають користувачам інформацію про те, як можна оптимізувати виявлені вузькі місця та поліпшити продуктивність. Розглянемо докладніше про параметри, які можуть бути включені до рекомендацій щодо оптимізації.

Опис проблеми (Problem Description) описує пояснення природи та причин вузького місця, яке потребує оптимізації. Це надає контекст для розуміння проблеми, яку слід вирішити.

Рекомендації (Recommendations) описують конкретні поради щодо того, як виправити вузьке місце чи здійснити оптимізацію. Такий параметр може

включати в себе зміни в коді, алгоритмах, використанні ресурсів, налаштуваннях системи та інше. Це надає інструкції для розробників щодо кроків, які слід вжити для вирішення проблеми.

Приклади коду (Code Samples) описують фрагменти вихідного коду, які ілюструють рекомендовані зміни або оптимізації. Це допомагає розробникам розуміти, як конкретно реалізувати рекомендації в коді.

Підказки та кращі практики (Tips and Best Practices) описують загальні поради та підказки, які допоможуть покращити продуктивність системи або програми. Це надає загальні принципи, які можуть бути корисними при оптимізації.

Посилання на документацію (Documentation References) -це посилання на офіційну документацію або джерела інформації, де можна знайти докладну інформацію про рекомендовані зміни. Це допомагає розробникам знайти додаткові ресурси для детального вивчення.

Прогнозований виграш в продуктивності (Expected Performance Gain) - це оцінка того, наскільки покращиться продуктивність після внесення рекомендованих змін. Це надає інформацію про те, наскільки вигідно буде виконати оптимізацію.

Пріоритет рекомендації (Recommendation Priority) описує позначення пріоритету рекомендації в порівнянні з іншими завданнями. Це може бути високий, середній, низький пріоритет тощо. Це допомагає визначити, які оптимізації слід виконувати першими.

Рекомендації щодо оптимізації є важливими для розробників і інженерів, оскільки вони надають конкретні інструкції та поради для покращення продуктивності системи чи програми. Вони допомагають зробити оптимальні зміни та ефективно вирішити виявлені вузькі місця.

Ще існують додаткові дані та висновки щодо факторів, які впливають на продуктивність. Це може бути інформація про виявлені аномалії або особливості у роботі системи за допомогою різних деталей аналізу - розділу вихідних даних програмного засобу дослідження bottlenecks, який містить більш докладну

інформацію про результати аналізу вузьких місць. Ця інформація може включати в себе специфікації, статистику, графіки, аналіз впливу та інші деталі, що допомагають зрозуміти, яким чином вузькі місця впливають на продуктивність системи чи програми. Розглянемо докладніше про параметри, які можуть бути включені до такого аналізу.

Специфікації вузьких місць (Bottleneck Specifications) описує повторення специфікацій вузьких місць, але з більш докладним описом та аналізом кожного з них. Це допомагає зрозуміти, яким чином кожне вузьке місце впливає на продуктивність.

Статистика вузьких місць (Bottleneck Statistics) описує кількісні дані та статистику, що пов'язані з кожним виявленим вузьким місцем, такі як час виконання, обсяг пам'яті, використання ресурсів тощо. Це надає додаткову інформацію про розмір та обсяг проблеми.

Графіки продуктивності (Performance Charts) описує графіки та діаграми, які ілюструють зміни продуктивності під час аналізу вузьких місць. Це допомагає візуалізувати зміни продуктивності та їх динаміку.

Аналіз впливу (Impact Analysis) - це оцінка впливу кожного вузького місця на загальну продуктивність системи чи програми. Це допомагає визначити, які зміни можуть мати найбільший вплив на продуктивність.

Додаткові деталі аналізу (Additional Analysis Details) - це додаткова інформація, яка може бути важливою для розуміння проблеми та виявлення можливих рішень. Це може включати в себе результати досліджень, висновки та рекомендації. Це надає додатковий контекст та інформацію для аналізу.

Посилання на джерела (References) описує посилання на джерела або документацію, де можна знайти додаткову інформацію про аналіз вузьких місць та рекомендації щодо оптимізації. Це допомагає розробникам знайти докладну інформацію та джерела для подальшого дослідження.

Таким чином, розгляд таких деталей аналізу є важливими для того, щоб зрозуміти природу проблеми з продуктивністю та прийняти ефективні рішення для її вирішення. Вони надають більше інформації та контексту для розробників

та інженерів, щоб вони могли вжити відповідних заходів щодо оптимізації системи чи програми.

Звіти про споживану ресурсами (Resource Utilization Reports) - це частина вихідних даних програмного засобу дослідження bottlenecks, яка містить інформацію про споживання різних ресурсів, таких як центральний процесор (CPU), оперативна пам'ять (RAM), диск, мережеві ресурси та інші, під час виконання системи чи програми. Ці звіти надають користувачам інформацію про те, як ресурси використовуються та чи є вони вузьким місцем продуктивності. Розглянемо докладніше про параметри, які можуть бути включені до звітів про споживану ресурсами.

Споживання CPU (CPU Utilization) описує інформацію про використання процесорного часу (CPU) під час роботи системи чи програми. Це може включати в себе відсоток використання CPU, розподілення за ядрами CPU та іншу відомості. Це допомагає визначити, чи відбувається інтенсивне використання CPU, що може бути вузьким місцем.

Споживання RAM (Memory Utilization) описує інформацію про використання оперативної пам'яті (RAM) під час роботи системи чи програми. Включає обсяг використаної пам'яті, обсяг вільної пам'яті, обмін пам'яті та інші показники. Це допомагає визначити, чи виникає необхідність в оптимізації використання пам'яті.

Дискове споживання (Disk Utilization) описує інформацію про використання дискового простору під час роботи системи чи програми. Включає обсяг записів та читань на диск, швидкість доступу до диску та інші дані. Це допомагає визначити, чи є доступ до диску вузьким місцем для додатка.

Мережеве споживання (Network Utilization) описує інформацію про використання мережевих ресурсів під час роботи системи чи програми. Включає в себе обсяг передачі даних через мережу, швидкість передачі, пінги та інші мережеві параметри. Це допомагає визначити, чи мережа впливає на продуктивність додатка.

Завантаження файлів та ресурсів (File and Resource Loading) описує інформацію про завантаження файлів, бібліотек, зображень та інших ресурсів під час виконання програми. Включає в себе час завантаження та кількість завантажених ресурсів. Це допомагає визначити, чи є завантаження ресурсів вузьким місцем.

Графіки та діаграми (Charts and Graphs) описує графіки та діаграми, які ілюструють споживання ресурсів в динаміці, допомагають візуалізувати зміни в споживанні ресурсів з часом. Це допомагає аналізувати тренди та зміни в споживанні ресурсів.

Посилання на джерела (References) описує посилання на джерела або іншу інформацію, де можна знайти додаткові деталі щодо споживання ресурсів та оптимізації. Це допомагає розробникам знайти більше інформації для подальшого аналізу та вирішення проблем.

Звіти про споживану ресурсами є важливими для аналізу та оптимізації продуктивності системи чи програми, оскільки вони надають інформацію про те, як ресурси використовуються та чи є вони вузьким місцем продуктивності. Вони допомагають визначити, де слід звернути увагу для покращення продуктивності та ефективності роботи додатка.

Журнали подій та відладки (Event and Debug Logs) - це спеціальні файли чи записи, в яких фіксуються події та відомості про виконання програми чи системи в процесі їх роботи. Вони є незамінним інструментом для виявлення, відстеження та вирішення проблем у програмному забезпеченні. Розглянемо докладніше про параметри, які можуть бути включені до журналів подій та відладки.

Типи подій (Event Types), де кожна подія має свій тип, який вказує на природу події. Типи можуть включати в себе інформацію про помилки, важливі події, попередження та інші статуси. Це допомагає класифікувати події та розуміти, що саме сталося.

Час події (Event Timestamp) описує дату та час, коли подія відбулася. Зазвичай вказується точний момент часу або часовий проміжок. Це дозволяє відстежувати порядок подій та визначати, коли вони відбулися.

Ідентифікатор події (Event ID) описує унікальний ідентифікатор для кожної події. Цей ідентифікатор допомагає знаходити конкретні події в журналах. Це використовується для пошуку та посилання на певні події.

Повідомлення про подію (Event Message) описує текстове або числове повідомлення, яке містить інформацію про подію. Це може бути описом помилки, діагностичною інформацією, деталями процесу та інше. Це надає докладну інформацію про те, що сталося під час події.

Додаткові атрибути (Additional Attributes) описує додаткові дані, які можуть бути пов'язані з подією. Це можуть бути параметри, змінні, об'єкти чи будь-яка інша інформація, що допомагає розібратися в події. Це надає додаткову контекстну інформацію для аналізу події.

Рівень важливості (Severity Level) вказує на серйозність події. Це може бути інформаційний, попередження, помилка, критична помилка тощо. Це допомагає виділяти найбільш важливі та критичні події.

Стек викликів (Call Stack) описує інформацію про послідовність викликів функцій та методів, яка привела до виникнення події. Допомагає відстежувати шлях, яким була спричинена подія. Це використовується при відладці та аналізі помилок.

Корисна інформація (Useful Information) описує додаткову інформацію, яка може бути корисною для розуміння контексту події та її впливу на систему. Це надає додаткову інформацію для аналізу та вирішення проблеми.

Рівень відладки (Debug Level) вказує на рівень відладки, який активував цю подію. Зазвичай використовується для відстеження відладочних повідомлень. Це використовується при розробці та відладці програми.

Файл журналу (Log File) описує інформацію про сам файл журналу, включаючи назву, розмір, розташування та інші деталі. Це вказує, де можна знайти журнали подій та відладки.

Посилання на джерела (References) описує посилання на джерела або іншу інформацію, де можна знайти додаткові деталі щодо подій та відладки. Це допомагає розробникам знайти більше інформації для подальшого дослідження.

Журнали подій та відладки є незамінними для виявлення, відстеження та вирішення проблем у програмному забезпеченні. Вони дозволяють розробникам та інженерам відстежувати події та відладкову інформацію, що допомагає знайти та виправити помилки, а також зберігати історію подій для аналізу та вдосконалення програми чи системи.

Інформація про продуктивність на різних обчислювальних платформах або пристроях, яка може бути корисною для порівняння та оптимізації.

Дані про продуктивність на різних платформах (Performance Data Across Platforms) - це інформація, яка дозволяє визначити, як програма чи система працює на різних обчислювальних платформах або апаратних конфігураціях. Ці дані грають важливу роль у визначенні того, чи відповідає продукт вимогам різних платформ і як можна оптимізувати його продуктивність. Розглянемо докладніше про параметри, які можуть бути включені до даних про продуктивність на різних платформах.

Платформа (Platform) описує інформацію про конкретну обчислювальну платформу, на якій був виконаний тест чи вимірювання продуктивності. Включає в себе назву платформи, апаратну архітектуру, операційну систему та інші характеристики. Це дозволяє визначити, як продукт веде себе на різних платформах.

Параметри апаратної конфігурації (Hardware Configuration) описує деталі про апаратну конфігурацію платформи, включаючи кількість та тип процесорів, обсяг оперативної пам'яті (RAM), видеокарти, тип та швидкість дискового простору та інші характеристики. Це допомагає визначити, як апаратна конфігурація впливає на продуктивність.

Параметри середовища (Environment Variables) описують налаштування середовища, такі як шляхи до даних, змінні середовища, параметри конфігурації

інфраструктури та інші налаштування, які можуть впливати на роботу програми чи системи. Це допомагає визначити, як середовище впливає на продуктивність.

Результати вимірювань продуктивності (Performance Metrics) описують числові та графічні показники продуктивності на даній платформі, такі як час відгуку, швидкість обробки даних, обсяг використаної пам'яті, навантаження на CPU та інші метрики. Це надає об'єктивну інформацію про продуктивність на даній платформі.

Графіки та діаграми (Charts and Graphs) описують графіки та діаграми, які візуалізують дані продуктивності на різних платформах. Вони демонструють зміни та тренди продуктивності в залежності від платформи. Це допомагає аналізувати та порівнювати продуктивність на різних платформах.

Порівняння результатів (Benchmark Comparison) описують порівняння результатів продуктивності на різних платформах, щоб визначити, яка платформа є більш вигідною для виконання програми чи системи. Це допомагає вибрати оптимальну платформу для запуску продукту.

Рекомендації щодо оптимізації (Optimization Recommendations) описують рекомендації та поради щодо оптимізації продукту на даній платформі для покращення продуктивності. Це допомагає розробникам покращити продукт для конкретної платформи.

Посилання на джерела (References) описують посилання на джерела або документацію, де можна знайти додаткову інформацію щодо продуктивності на різних платформах. Це допомагає розробникам знайти більше інформації для подальшого дослідження.

Дані про продуктивність на різних платформах допомагають розробникам та інженерам зрозуміти, як їх програми чи системи працюють на різних обчислювальних середовищах і як можна оптимізувати їх роботу. Ця інформація важлива для забезпечення максимальної продуктивності та ефективності продукту на різних платформах.

Інша статистика та метрики (Additional Statistics and Metrics) показують додаткові числові значення, які вказують на різні аспекти продуктивності, такі як середнє значення, стандартне відхилення тощо.

Інша статистика та метрики (Additional Statistics and Metrics) в контексті дослідження bottlenecks у високопродуктивних обчисленнях можуть включати різноманітні показники та дані, які допомагають розробникам отримати більше інсайтів про функціонування програми або системи. Розглянемо кілька типових метрик і статистики.

Розподіл використання ресурсів (Resource Utilization Distribution) описує графіки або діаграми, які показують, як різні ресурси, такі як CPU, пам'ять, диск, мережа і інші, використовуються програмою на різних етапах її роботи. Це допомагає виявити, які ресурси є вузькими місцями продуктивності.

Час виконання функцій (Function Execution Times) вимірює час, який витрачається на виконання окремих функцій або операцій у програмі. Це допомагає виявити, які функції є найбільш часоємкими і можуть бути оптимізовані.

Кількість оброблених запитів (Processed Requests Count) описує кількість запитів, які програма успішно обробила за певний період часу.Це вказує на завантаженість системи та її здатність обслуговувати запити.

Обсяг оброблених даних (Processed Data Volume) описує кількість даних, які були оброблені програмою чи системою. Це вказує на обсяг роботи, яку виконує програма.

Ступінь паралельності (Degree of Parallelism) описує кількість паралельних обчислень або операцій, які виконуються одночасно. Це допомагає визначити, як програма використовує можливості паралельного обчислення.

Параметри робочого навантаження (Workload Parameters) описує додаткові параметри, які визначають робоче навантаження на програму чи систему, такі як кількість одночасних користувачів, обсяги даних, типи операцій і інші. Це допомагає визначити, які навантаження призводять до погіршення продуктивності.

Метрики ефективності алгоритмів (Algorithm Efficiency Metrics) описують вимірювання та оцінку ефективності використовуваних алгоритмів, такі як складність обчислень, кількість операцій і пам'яті, час виконання тощо. Це допомагає визначити, які алгоритми є оптимальними для конкретних завдань.

Метрики мережевої взаємодії (Network Interaction Metrics) описують інформацію про роботу мережевої взаємодії, така як час передачі даних, кількість переданих пакетів, пропускна здатність мережі тощо. Це допомагає виявити проблеми в мережевій взаємодії та шляхи їх вирішення.

Метрики збереження даних (Data Storage Metrics) описує інформацію про роботу зберігання даних, така як швидкість запису/читання, обсяг використаного простору, ступінь стиснення даних тощо. Це допомагає визначити, як ефективно програма взаємодіє зі сховищем даних.

Діагностика помилок (Error Diagnostics) описує інформацію про помилки та винятки, які виникають під час роботи програми. Це допомагає виявити та виправити проблеми та помилки в програмі.

Ці метрики та статистика допомагають розробникам більш глибоко розуміти роботу програми чи системи, виявляти проблеми продуктивності та шукати способи їх вирішення. Вони є важливою частиною процесу оптимізації та покращення продукту.

Таким чином, розглянуті вихідні параметри служать основою для подальшого аналізу, вдосконалення та оптимізації програмного засобу або системи для досягнення кращої продуктивності та ефективності.

3.3. Обчислення вхідних параметрів для обробки даних у програмному засобі

Обчислення вхідних параметрів для обробки даних у програмному засобі дослідження bottlenecks у високопродуктивних обчисленнях може бути виконано за допомогою різних методів та інструментів, залежно від конкретної задачі та програмного середовища (табл. 3.3). Розглянемо деякі загальні кроки та методи для обчислення вхідних параметрів.

Спершу потрібно зібрати необхідні дані для обробки (рис. 3.1). Це може включати в себе вимірювання продуктивності програми або системи на різних платформах або з різними параметрами. Дані також можуть бути зібрані з різних джерел, таких як журнали подій, моніторинг ресурсів тощо.

Збір даних - це перший та дуже важливий етап процесу дослідження bottlenecks у високопродуктивних обчисленнях. Під час цього етапу збираються різні види даних та інформації про роботу програми чи системи з метою подальшого аналізу та оцінки продуктивності. Розглянемо процес збору даних наступним чином.

Визначення даних для збору з'ясовує, які конкретні дані та метрики цікавлять. Це можуть бути, наприклад, час виконання, використання процесора, обсяги оброблених даних, кількість запитів, швидкість мережі, витрати пам'яті та інше. Інструменти збору даних визначають, які інструменти та методи можна використовувати для збору даних. Це можуть бути спеціалізовані програми для моніторингу, системи журналювання подій, інструменти для профілювання коду, або навіть власний код, який реєструє необхідну інформацію.

Таблиця 3.3

Обчислення вхідних параметрів для обробки даних у програмному засобі дослідження bottlenecks

Крок обчислення	Опис
Крок 1	Збір даних про продуктивність програми чи системи на різних платформах або з різними параметрами. Це може включати вимірювання часу виконання, використання ресурсів, обсяги оброблених даних та інше.
Крок 2	Аналіз зібраних даних, включаючи статистичний аналіз для визначення ключових параметрів, таких як середні значення, дисперсія, медіани тощо.
Крок 3	Визначення параметрів і метрик, які будуть використовуватися

	для оцінки продуктивності та вузьких місць. Це може включати в себе вимірювання часу виконання, розподіл ресурсів, кількість оброблених даних та інше.
Крок 4	Проведення експериментів і моделювання для отримання даних щодо продуктивності на різних параметрах або платформах. Це може включати в себе запуск програми з різними налаштуваннями та збір даних про її роботу.
Крок 5	Автоматизація процесу збору даних для постійного моніторингу продуктивності. Використання інструментів для моніторингу продуктивності та збору метрик.
Крок 6	Періодичний аналіз продуктивності для виявлення змін та bottlenecks з плином часу або при зміні умов.

Налаштування інструментів встановлє та налаштовує обрані інструменти для збору даних. Налаштовувати їх потрібно так, щоб вони збирали необхідну інформацію та метрики під час роботи програми або системи.

Початок досліджень запускає програму чи систему під контролем інструментів для збору даних. Доцільно провести дослідження на різних платформах або з різними параметрами, якщо це застосовно до вашого дослідження.

Збір та реєстрація даних означає, що під час роботи програми чи системи інструменти для збору даних автоматично фіксують та реєструють обрані метрики та інформацію. Це може включати в себе збереження часу виконання, запис подій, вимірювання використання ресурсів, тощо.

Рисунок 3.1 - UML діаграма аналізу вхідних параметрів для обробки даних у програмному засобі дослідження bottlenecks

Збереження та обробка даних показує зібрані дані, що зазвичай зберігаються у вигляді журналів або баз даних. Після збору вони піддаються обробці, включаючи агрегацію, фільтрацію та інші операції, які роблять дані зрозумілими та готовими для аналізу.

Візуалізація та аналіз є одним із важливих аспектів збору даних. При цьому можливо використовувати графіки, діаграми та інші інструменти для аналізу даних та виявлення закономірностей.

Збереження та архівування даних необхідно для збережені від втрати. Зазвичай дані архівуються для подальшого використання та порівняння з іншими даними.

Підготовка до аналізу продуктивності означає те, що дані, які були зібрані та оброблені, готуються для аналізу продуктивності та визначення вузьких місць.

Збір даних - це ітеративний процес, і залежно від результатів аналізу, можливо внести зміни в обрані метрики або методи збору даних. Головна мета

цього етапу - отримати об'єктивну та достовірну інформацію про продуктивність системи для подальшого аналізу та оптимізації.

Після збору даних потрібно їх аналізувати та обробляти. Це може включати в себе вимірювання часу виконання, розподіл ресурсів, обсяги оброблених даних і багато іншого.

Аналіз та обробка даних - це ключовий етап дослідження bottlenecks у високопродуктивних обчисленнях, під час якого зібрані дані перетворюються у корисну інформацію, що дозволяє розуміти, як система працює та виявляти проблеми продуктивності. Розглянемо детальніше процес аналізу та обробки даних, який полягає в наступному.

Перегляд та очищення даних означає те, що першим кроком є перегляд та очищення зібраних даних. Під час перегляду визначається, чи є в даних незрозумілі або відхилені значення, і виправляєте або вилучаєте їх. Також перевіряється на наявність пропущених даних.

Агрегація даних означає те, що для зручності аналізу може бути проведена агрегація даних. Наприклад, групування даних за певними параметрами, обчислення середніх значень, сум тощо. Це допомагає зменшити обсяг даних та виділити головні тренди.

Статистичний аналіз означає те, що важливо провести статистичний аналіз даних. Це включає в себе визначення основних характеристик розподілу даних, таких як середнє значення, медіана, дисперсія, стандартне відхилення тощо. Статистичний аналіз допомагає зрозуміти, як розподілені дані і чи є в них викиди (аномалії).

Візуалізація даних означає те, що використовуються графіки, діаграми та інші візуальні засоби для відображення даних. Візуалізація допомагає виявити залежності та закономірності, які можуть бути невидимими в сухих числах.

Порівняння даних означає те, що порівнюються дані з різних експериментів, платформ або налаштувань, щоб визначити, які параметри впливають на продуктивність. Це допомагає виявити вузькі місця.

Інтерпретація результатів означає те, що виконуються висновки з отриманих даних. Визначається, які параметри впливають на продуктивність, і які можливі шляхи оптимізації.

Формулювання висновків та рекомендацій означає те, що на основі аналізу даних формулюйте висновки та рекомендації щодо покращення продуктивності. Такі рекомендації можуть включати зміни у коді програми, оптимізацію алгоритмів, вибір певних платформ або налаштувань, тощо.

Документація результатів означає те, що важливо документувати всі результати аналізу та рекомендацій, щоб вони були доступні для подальшого використання та спільної роботи з іншими членами команди.

Аналіз та обробка даних допомагають отримати важливу інформацію для подальшої оптимізації продукту або системи та виявлення вузьких місць, що вимагають уваги.

Також, важливо використовувати методи статистичного аналізу для визначення ключових параметрів, таких як середні значення, дисперсія, медіани тощо. Це допомагає зрозуміти розподіл даних та виявити вузькі місця продуктивності.

Статистичний аналіз - це процес обробки та інтерпретації даних з використанням методів та технік статистики з метою отримання інсайтів та розуміння характеристик даних. Цей аналіз допомагає виявити патерни, закономірності та структуру даних, а також зрозуміти розподіл значень та ступінь варіації в них. Розглянемо докладніше етапи статистичного аналізу.

Описова статистика включає в себе обчислення основних статистичних параметрів даних, таких як середнє значення (середнє арифметичне), медіана (значення, що ділить набір даних на дві рівні частини), дисперсія (середній квадрат відхилення від середнього) і стандартне відхилення (квадратний корінь з дисперсії). Ці метрики надають загальне уявлення про розподіл даних.

Графіки та діаграми використовуються для візуалізації даних разом з гістограми, ящиками з вусами (box plots), діаграмами розсіювання (scatter plots)

та інші графічні представлення допомагають виявити залежності та закономірності в даних.

Інтервали довіри дозволяють оцінити, наскільки точними є обчислені статистичні параметри (наприклад, середнє значення) і які можливі діапазони можливих значень.

Тестування гіпотез застосовуються для статистичних тестів та перевірки гіпотез. Наприклад, тест Стьюдента для порівняння середніх значень двох груп або аналіз дисперсії для порівняння трьох або більше груп.

Кореляційний аналіз визначає ступінь взаємозв'язку між двома або більше змінними. Кореляція може бути позитивною (коли одна змінна зростає, інша також зростає), негативною (коли одна змінна зростає, інша спадає) або відсутньою.

Регресійний аналіз використовується для моделювання взаємозв'язку між залежною та незалежними змінними. Зазвичай використовується лінійна регресія для прогнозування значень залежної змінної на основі незалежних змінних.

Аналіз варіації (ANOVA) використовується для визначення статистичної важливості між більш ніж двома групами. Він допомагає визначити, чи є статистично значущі різниці між групами.

Часовий аналіз використовується якщо дані мають часовий аспект, то важливо аналізувати їх за допомогою часових рядів та моделей прогнозування.

Статистичний аналіз допомагає зрозуміти основні характеристики даних, виявити взаємозв'язки та закономірності, і зробити висновки на основі об'єктивних даних. Це важливий інструмент для вивчення продуктивності систем та виявлення вузьких місць для подальшої оптимізації.

Зазвичай, для отримання даних щодо продуктивності на різних параметрах або платформах, використовуються експерименти та моделювання. Експерименти полягають у запуску програми з різними налаштуваннями і зборі даних про її роботу. Моделювання може використовуватися для прогнозування продуктивності на основі зібраних даних.

Експерименти і моделювання - це методи дослідження, які використовуються для вивчення продуктивності і вузьких місць у високопродуктивних обчисленнях. Ці методи дозволяють вам провести контрольні та експериментальні випробування, використовуючи реальні або симульовані умови, щоб зрозуміти, як система працює та які фактори впливають на її продуктивність. Розглянемо ці методи більш ретельно.

Під час контрольного експерименту встановлюються базові умови і вимірюєте продуктивність системи. Наприклад, можна визначити, як система працює при певних налаштуваннях або на певних обчислювальних платформах.

Під час експериментів зі змінними факторами вносяться зміни в обрані параметри, такі як налаштування системи або характеристики обчислювальних платформ. Далі спостерігається, як ці зміни впливають на продуктивність. Це допомагає виявити оптимальні параметри.

Для проведення математичне моделювання використовується для створення математичних моделей системи, які описують її роботу та взаємозв'язки між параметрами. Можна проводити симуляції з різними вхідними даними, щоб прогнозувати продуктивність у різних сценаріях.

Симуляція використовується для створення віртуальних моделей системи, які дозволяють відтворити реальну роботу системи і вивчити її продуктивність без реального виконання експериментів. Симуляція допомагає зекономити час і ресурси.

Під час експериментів і моделювання важливо збирати дані про продуктивність. Це може включати вимірювання часу виконання, обсяг використаної пам'яті, завантажені ресурси обчислювальної платформи тощо.

Після проведення експериментів або моделювання аналізуються отримані дані та визначаються, які фактори впливають на продуктивність і які можливі шляхи оптимізації системи.

На основі результатів експериментів і моделювання формулюються рекомендації щодо покращення продуктивності. Це може включати зміни в коді

програми, вибір оптимальних налаштувань або використання більш потужних обчислювальних ресурсів.

Важливо документувати всі етапи експериментів та моделювання, щоб інші члени команди могли розуміти дії керівника і результати дослідження.

Експерименти і моделювання допомагають отримати глибоке розуміння продуктивності системи та виявити вузькі місця, що потребують уваги. Ці методи є важливою частиною процесу дослідження та оптимізації високопродуктивних обчислень.

На основі аналізу даних визначаються параметри та метрики, які слід використовувати для оцінки продуктивності та вузьких місць.

Визначення параметрів і метрик є важливою частиною дослідження bottlenecks в обчислювальних задачах. Цей процес включає в себе визначення ключових параметрів системи та вибір метрик, які допомагають вимірювати продуктивність цих параметрів. Розглянемо цей процес більш детально.

Для визначення параметрів системи спочатку потрібно ідентифікувати параметри, які впливають на продуктивність системи. Це можуть бути параметри обладнання (наприклад, частота процесора, обсяг пам'яті), параметри програмного забезпечення (наприклад, налаштування операційної системи або бази даних) та параметри задачі (наприклад, об'єм вхідних даних або розмір обчислювальних завдань).

Залежно від параметрів визначити, які змінні або параметри в системі залежать від основних параметрів. Наприклад, якщо вивчаєте вплив частоти процесора на продуктивність, то час виконання завдання може бути залежним параметром.

Визначаються основні метрики продуктивності, які відображають продуктивність системи. Це може бути час виконання, швидкість обробки даних, кількість операцій в секунду або інші вимірювані параметри.

Додаткові метрики можуть включати в себе споживання ресурсів (пам'ять, CPU, дисковий простір), кількість помилок або вартість обчислень. Вибір додаткових метрик залежить від конкретних цілей дослідження.

Після визначення параметрів і метрик, забезпечується можливість виміряти ці метрики в реальному часі під час виконання обчислювальних задач або експериментів. Вимірювання допомагають зібрати дані, необхідні для подальшого аналізу продуктивності.

Після збору даних виконується аналіз продуктивності параметрів системи. При цьому, оцінюється, які параметри впливають на основні метрики продуктивності та які мають найбільший вплив.

На основі аналізу результатів розробляються рекомендації щодо оптимізації системи. Наприклад, якщо по результатам досліджень визначили, що збільшення обсягу пам'яті покращує продуктивність, то рекомендацією може бути розширення пам'яті.

Після впровадження оптимізаційних заходів здійснюється моніторинг системи та вносити налаштування, якщо це необхідно, для забезпечення сталої продуктивності.

Таким чином, визначення параметрів і метрик є ключовим кроком у дослідженні та оптимізації продуктивності обчислювальних систем. Це допомагає зрозуміти, які фактори впливають на продуктивність і як їх оптимізувати для досягнення бажаних результатів.

Для постійного моніторингу продуктивності можна розглянути автоматизацію процесу збору даних. Для цього можуть бути використані інструменти для моніторингу продуктивності та збору метрик.

Автоматизація процесу збору даних - це важливий аспект дослідження bottlenecks у високопродуктивних обчисленнях. Цей процес передбачає використання програмних інструментів та скриптів для автоматичного збору інформації про продуктивність системи або програми без значного втручання користувача. Важливі аспекти автоматизації процесу збору даних включають:

Вибір інструментів для збору даних залежить від інструментів або програм, які дозволяють автоматизовано збирати дані про продуктивність. Це може включати в себе системи моніторингу, програми для збору журналів подій або інші спеціалізовані засоби.

Конфігурація і налаштування інструментів використовується для вимірювання та збору даних відповідно до потреб користувачів. Це може включати в себе встановлення параметрів моніторингу, вибір метрик, які потрібно виміряти, і налаштування інтервалів вимірювань.

З метою розробки скриптів і автоматизованих завдань створюються скрипти або завдання, які виконують збір даних на регулярній основі. Наприклад, можна написати скрипт, який регулярно запускається і вимірює час виконання певних операцій або моніторить використання ресурсів.

Збір та збереження даних автоматично збирає дані про продуктивність і зберігає їх у відповідних сховищах даних. Це може бути база даних, файли журналів або хмарні сховища.

Аналіз і візуалізація даних використовується під час автоматичного аналізу, де застосовуються інструменти візуалізації для перетворення зібраних даних у корисну інформацію та графіки. Це допомогає спостерігати за трендами і виявляти аномалії в онлайн режимі.

Для повідомлень та сповіщень встановлюються автоматичні сповіщення або повідомлення, якщо процеси показують вузькі місця або проблеми продуктивності. Це дозволяє оперативно реагувати на проблеми.

Моніторинг у реальному часі проводиться якщо це необхідно, тому він налаштовується у реальному часі, щоб користувачі могли отримувати актуальну інформацію про продуктивність в момент протікання різних процесів.

Таким чином, автоматизація процесу збору даних допомагає ефективно відстежувати продуктивність системи та реагувати на можливі проблеми без значного зусилля та втрати часу.

Також, важливо проведення періодичного аналізу, що використовується як процес обчислення вхідних параметрів та повинен бути періодичним, оскільки продуктивність може змінюватися з часом або при зміні умов.

Періодичний аналіз є важливою частиною дослідження bottlenecks у високопродуктивних обчисленнях. Цей підхід передбачає систематичний та регулярний аналіз продуктивності системи, програми або задачі через певні

проміжки часу. Основна мета періодичного аналізу - виявлення змін у продуктивності, ідентифікація проблем та вирішення їх до того, як вони стануть критичними. Основні аспекти періодичного аналізу включають:

Для визначення інтервалу аналізу обирається проміжок часу, з яким дослідник буде проводити аналіз. Це може бути щогодини, щодня, щотижня або інше підходяще для контексту.

Під час вибору метрик визначаються метрики, які будуть вимірюватись під час кожного періоду аналізу. Це може бути час виконання, кількість оброблених даних, споживання ресурсів, кількість помилок тощо.

Для автоматизованого збору даних автоматизована система налаштовується для збору даних визначених метрик протягом вибраного інтервалу. Це може включати в себе налаштування моніторингу, використання спеціалізованих інструментів для збору даних або розробку власних скриптів.

Для аналізу та порівняння даних порівнюйться дані, що зібрані під час різних періодів, щоб виявити зміни у продуктивності. При цьому, виконується пошук трендів, аномалій або вузьких місць, які можуть впливати на продуктивність системи.

Ідентифікація проблем і вирішення проводиться, якщо під час аналізу виявляються проблеми або зменшення продуктивності. Тоді, ідентифікуються їх причини і розробляють план вирішення цих проблем. Це може включати в себе налаштування системи, оптимізацію коду програми або розширення ресурсів.

Під час звітності та моніторингу генеруються звіти про результати аналізу та вживаються заходи для моніторингу виконання запланованих дій на основі виявлених проблем.

Таким чином, періодичний аналіз дозволяє вчасно виявляти проблеми з продуктивністю і покращувати ефективність системи або програми. Він також сприяє попередженню можливих аварій або збоїв, що можуть виникнути через вузькі місця у продуктивності.

Також, залежно від конкретної задачі та програмного середовища, може бути застосовано інші методи та інструменти для обчислення вхідних

параметрів. Важливо мати чіткий план та процес збору та аналізу даних для визначення bottlenecks та покращення програми чи системи.

3.4. Висновки

В третьому розділі виконано реалізацію розробленого методу у програмному засобі. Розроблені вхідні та вихідні параметри програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях. Виконані обчислення вхідних параметрів для обробки даних у програмному засобі, що виконані за допомогою різних методів та інструментів, залежно від конкретної задачі та програмного середовища. При цьому, ретельно розглянуті деякі загальні кроки та методи для обчислення таких вхідних параметрів.

РОЗДІЛ 4.
ЕКОНОМІЧНИЙ РОЗДІЛ

Виконання науково-дослідної роботи завжди передбачає отримання певних результатів і вимагає відповідних витрат. Результати виконаної роботи завжди дають нам нові знання, які в подальшому можуть бути використані для удосконалення та/або розробки (побудови) нових, більш продуктивних зразків техніки, процесів та програмного забезпечення.

Дослідження на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» може бути віднесено до фундаментальних і пошукових наукових досліджень і спрямоване на вирішення наукових проблем, пов’язаних з практичним застосуванням. Основою таких досліджень є науковий ефект, який виражається в отриманні наукових результатів, які збільшують обсяг знань про природу, техніку та суспільство, які розвивають теоретичну базу в тому чи іншому науковому напрямку, що дозволяє виявити нові закономірності, які можуть використовуватися на практиці.

Для цього випадку виконаємо такі етапи робіт:

1) здійснимо проведення наукового аудиту досліджень, тобто встановлення їх наукового рівня та значимості;

2) проведемо планування витрат на проведення наукових досліджень;

3) здійснимо розрахунок рівня важливості наукового дослідження та перспективності, визначимо ефективність наукових досліджень.

4.1 Оцінювання наукового ефекту

Основними ознаками наукового ефекту науково-дослідної роботи є новизна роботи, рівень її теоретичного опрацювання, перспективність, рівень розповсюдження результатів, можливість реалізації. Науковий ефект НДР на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» можна охарактеризувати двома показниками: ступенем наукової новизни та рівнем теоретичного опрацювання.

Значення показників ступеня новизни і рівня теоретичного опрацювання науково-дослідної роботи в балах наведені в табл. 4.1 та 4.2.

Таблиця 4.1 – Показники ступеня новизни науково-дослідної роботи виставлені експертами

Ступінь новизни	Характеристика ступеня новизни	Значення ступеня новизни, бали		
		Експерти (ПІБ, посада)		
		1	2	3
Принципово нова	Робота якісно нова за постановкою задачі і ґрунтується на застосуванні оригінальних методів дослідження. Результати дослідження відкривають новий напрям в даній галузі науки і техніки. Отримані принципово нові факти, закономірності; розроблена нова теорія. Створено принципово новий пристрій, спосіб, метод	0	0	0
Нова	Отримана нова інформація, яка суттєво зменшує невизначеність наявних значень (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту). Проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів	59	57	55
Відносно нова	Робота має елементи новизни в постановці задачі і методах дослідження. Результати дослідження систематизують і узагальнюють наявну інформацію, визначають шляхи подальших досліджень; вперше знайдено зв’язок (або знайдено новий зв’язок) між явищами. В принципі відомі положення розповсюджені на велику кількість об’єктів, в результаті чого знайдено ефективне рішення. Розроблені більш прості способи для досягнення відомих результатів. Проведена часткова раціональна модифікація (з ознаками новизни)	0	0	0

Традиційна	Робота виконана за традиційною методикою. Результати дослідження мають інформаційний характер. Підтверджені або поставлені під сумнів відомі факти та твердження, які потребують перевірки. Знайдено новий варіант рішення, який не дає суттєвих переваг в порівнянні з існуючим	0	0	0
Не нова	Отримано результат, який раніше зафіксований в інформаційному полі, та не був відомий авторам	0	0	0
Середнє значення балів експертів		57,0		

Згідно отриманого середнього значення балів експертів ступінь новизни характеризується як нова, тобто отримана нова інформація, яка суттєво зменшує невизначеність наявних знань (по-новому або вперше пояснені відомі факти, закономірності, впроваджені нові поняття, розкрита структура змісту) та проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів.

Таблиця 4.2 – Показники рівня теоретичного опрацювання науково-дослідної роботи виставлені експертами

Характеристика рівня теоретичного опрацювання	Значення показника рівня теоретичного опрацювання, бали		
	Експерт (ПІБ, посада)		
	1	2	3
Відкриття закону, розробка теорії	0	0	0
Глибоке опрацювання проблеми: багатоаспектний аналіз зв’язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу	65	64	66
Розробка способу (алгоритму, програми), пристрою, отримання нової речовини	0	0	0
Елементарний аналіз зв’язків між фактами та наявною гіпотезою, класифікація, практичні рекомендації для окремого випадку тощо	0	0	0
Опис окремих елементарних фактів, викладення досвіду, результатів спостережень, вимірювань тощо	0	0	0
Середнє значення балів експертів	65,0		

Згідно отриманого середнього значення балів експертів рівень теоретичного опрацювання науково-дослідної роботи характеризується як глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу.

Показник, який характеризує рівень наукового ефекту, визначаємо за формулою [Козловський, Лесько, Кавецький]:

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор}, \quad (4.1)$$

де $k_{нов}, k_{теор}$ - показники ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи, $k_{нов} = 57,0, k_{теор} = 65,0$ балів;

0,6 та 0,4 – питома вага (значимість) показників ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи.

$$E_{нау} = 0,6 \cdot k_{нов} + 0,4 \cdot k_{теор} = 0,6 \cdot 57,0 + 0,4 \cdot 65,00 = 60,20 \text{ балів.}$$

Визначення характеристики показника $E_{нау}$ проводиться на основі висновків експертів виходячи з граничних значень, які наведені в табл. 4.3.

Таблиця 4.3 – Граничні значення показника наукового ефекту

Досягнутий рівень показника	Кількість балів
Високий	70…100
Середній	50…69
Достатній	15…49
Низький (помилкові дослідження)	1…14

Відповідно до визначеного рівня наукового ефекту проведеної науково-дослідної роботи на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях», даний рівень становить 60,20 балів і відповідає статусу - середній рівень. Тобто у даному випадку можна вести мову про потенційну фактичну ефективність науково-дослідної роботи.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою **[Козловський, Лесько, Кавецький]**:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 20200{,}00 \cdot 21 / 21 = 20200{,}00 \ грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн

Керівник науково-дослідної роботи	20200,00	961,90	21	20200,00
Інженер-розробник програмного забезпечення	24000,00	1142,86	21	24000,00
Науковий співробітник	15800,00	752,38	7	5266,67
Технік	9000,00	428,57	7	3000,00
Всього				52466,67

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{4.3}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{4.4}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) **[Козловський, Лесько, Кавецький]**;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих

об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 21 дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 6700,00 · 1,10 · 1,35 / (21 · 8) = 59,22 *грн.*

$З_{р1}$ = 59,22 · 7,10 = 420,48 *грн.*

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка електронно-обчислювального обладнання	7,10	2	1,10	59,22	420,48
Підготовка робочого місця розробника програмного забезпечення	5,50	2	1,10	59,22	325,73
Інсталяція програмного забезпечення	8,20	5	1,70	91,53	750,52
Формування дослідної бази даних (2 бази)	24,00	2	1,10	59,22	1421,36
Налагодження програмних модулів	5,40	5	1,70	91,53	494,24
Підготовка програмного забезпечення серверного обладнання	11,00	5	1,70	91,53	1006,79
Тестування програмного забезпечення	11,00	2	1,10	59,22	651,46
Всього					5070,58

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (52466,67 + 5070,58) · 11 / 100% = 6329,10 грн.

4.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (52466,67 + 5070,58 + 6329,10) · 22 / 100% = 14050,60 грн.

4.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях».

Витрати на матеріали на даному етапі проведення досліджень в основному пов’язані з використанням моделей елементів та моделювання роботи і досліджень за допомогою комп’ютерної техніки та створення експериментальних математичних моделей або програмного забезпечення, тому дані витрати формуються на основі витратних матеріалів характерних для офісних робіт.

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\,, \qquad (4.7)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$M_1 = 4 \cdot 192{,}00 \cdot 1{,}05 - 0 \cdot 0 = 806{,}40$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір 500-80	192,00	4	-	-	806,40
Папір для записів А5	99,00	4	-	-	415,80
Органайзер офісний Office	210,00	3	-	-	661,50
Канцелярське приладдя (набір офісного працівника)	182,00	3	-	-	573,30
Картридж для принтера HP	1050,00	1	-	-	1102,50
Диск оптичний Optivisio CD-RW	24,00	3	-	-	75,60
USB Flesh-пам'ять DATA 16 GB	139,00	1	-	-	145,95
Тека для паперів	120,00	3	-	-	378,00
Всього					4159,05

4.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» відсутні.

4.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.8)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 31399,00 \cdot 1 \cdot 1,04 = 32654,96$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання бази даних (BigDATA)	1	31399,00	32654,96
Машрутизатор	1	12600,00	13104,00
Всього			45758,96

4.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.9)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *9325,00 · 1 · 1,01 = 9418,25 грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище розробки Intellij IDEA Ultimate	1	9325,00	9418,25
База даних MySQL Server	1	4672,00	4718,72
База даних DB Server	1	3975,00	4014,75
Всього			18151,72

4.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} , \qquad (4.10)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (52499{,}00 \cdot 1) / (3 \cdot 12) = 1458{,}31$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
ЕОМ для формування та дослідження систем високопродуктивного обчислення	52499,00	3	1	1458,31
ОС Windows	6750,00	3	1	187,50
Прикладне програмне забезпечення моделювання	81059,00	3	1	2251,64
Прикладне програмне забезпечення проектування систем	6780,00	3	1	188,33
Прикладний пакет Microsoft Office	6510,00	3	1	180,83
Приміщення лабораторії досліджень	39990,00	30	1	111,08

Принтер HP 1560 laserJet	7320,00	5	1	122,00
Робоче місце інженера-дослідника	8959,00	5	1	149,32
Всього				4649,01

4.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.11)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,25 \cdot 2,5 \cdot 7,50 \cdot 0,95 / 0,97 = 4,69$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Принтер HP 1560 laserJet	0,25	2,5	4,69
Робоче місце інженера-дослідника	0,10	160,0	120,00
Серверне обладнання бази даних (BigDATA)	0,56	120,0	504,00
Маршрутизатор	0,03	120,0	27,00
Інше обладнання	0,15	120,0	135,00
ЕОМ для формування та дослідження систем високопродуктивного обчислення	0,32	160,0	384,00

Всього	1174,69

4.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (4.12)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (52466,67 + 5070,58) · 20 / 100% = 11507,45 грн.

4.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (4.13)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (52466,67 + 5070,58) · 30 / 100% = 17261,18 грн.

4.2.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.14)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (52466,67 + 5070,58) · 50 / 100% = 28768,63 грн.

4.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.15)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 105%.

$В_{нзв}$ = (52466,67 + 5070,58) · 105 / 100% = 60414,11 грн.

Витрати на проведення науково-дослідної роботи на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.16)$$

$В_{заг}$= 52466,67 +5070,58 +6329,10 +14050,60 +4159,05 +0,00 +45758,96 + 18151,72 + 4649,01 +1174,69 +11507,45 +17261,18 +28768,63 +60414,11 = 269761,74 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.17)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 269761,74 / 0,95 = 283959,72 грн.

4.3 Оцінювання важливості та наукової значимості науково-дослідної роботи

Оцінювання та доведення ефективності виконання науково-дослідної роботи фундаментального чи пошукового характеру є достатньо складним процесом і часто базується на експертних оцінках, тому має вірогідний характер.

Для обґрунтування доцільності виконання науково-дослідної роботи на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» використовується спеціальний комплексний показник, що враховує важливість, результативність роботи, можливість впровадження її результатів у виробництво, величину витрат на роботу.

Комплексний показник K_P рівня науково-дослідної роботи може бути розрахований за формулою:

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t}, \quad (4.18)$$

де I – коефіцієнт важливості роботи. Приймемо $I = 4$;

n – коефіцієнт використання результатів роботи; $n = 0$, коли результати роботи не будуть використовуватись; $n = 1$, коли результати роботи будуть використовуватись частково; $n = 2$, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках; $n = 3$, коли

результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок. Приймемо n=2;

T_C – коефіцієнт складності роботи. Приймемо $T_C = 3$;

R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то $R = 4$; якщо результати роботи відповідають відомому рівню, то $R = 3$; якщо нижче відомих результатів, то $R = 1$. Приймемо $R = 3$;

B – вартість науково-дослідної роботи, тис. грн. Приймемо B =283959,72 грн;

t – час проведення дослідження. Приймемо t = 0,08 років, (1 міс.).

Визначення показників *I, n, T_C, R, B, t* здійснюється експертним шляхом або на основі нормативів [Козловський, Лесько, Кавецький].

$$K_P = \frac{I^n \cdot T_C \cdot R}{B \cdot t} = 4^2 \cdot 3 \cdot 3 \,/\, 284{,}0 \cdot 0{,}08 = 6{,}09.$$

Якщо $K_P > 1$, то науково-дослідну роботу на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» можна вважати ефективною з високим науковим, технічним і економічним рівнем.

4.4 Висновок до розділу 4

Витрати на проведення науково-дослідної роботи на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» складають 283959,72 грн. Відповідно до проведеного аналізу та розрахунків рівень наукового ефекту проведеної науково-дослідної роботи на тему «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях» є середній, а дослідження актуальними, рівень доцільності виконання науково-дослідної роботи $K_P > 1$, що свідчить про потенційну ефективність з високим науковим, технічним і економічним рівнем.

ВИСНОВКИ

В першому розділі виконувалось обгрунтування методу аналізу дослідження bottlenecks у високопродуктивних обчисленнях, де була розглянута необхідність використання сучасних методів та інструментів розробки програмних засобів. Розроблені задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях. Виконано аналіз сучасних методів розробки програмних засобів для дослідження bottlenecks.

В другому розділі розглянуто загальна характеристика методів дослідження bottlenecks у високопродуктивних обчисленнях. Досліджено особливість використання методів дослідження bottlenecks у високопродуктивних обчисленнях, яка полягала в тому, щоб знайти та виправити "вузькі місця" або обмеження, які гальмують продуктивність, тим самим забезпечити кращу ефективність використання обчислювальних ресурсів. Запропонований метод діагностики bottlenecks у програмному засобі, який полягає в встановлені та знищені обробників у програмному коді автоматизованої системи, що керується за допомогою конфігураційних файлів на етапі компіляції та виконання програм. Такі обробників у програмному коді автоматизованої системи надають можливість визначати час виконання певних ділянок програмного коду та визначати неприпустимі дії виконання певних функцій (наприклад, ділення на нуль). Це дозволяє підвищити продуктивність та надійність у роботі програмного забезпечення в інформаційних системах як виробничої експлуатації.

Також, визначені вимоги щодо функціонування програмного засобу та розроблені структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.

В третьому розділі виконано реалізацію розробленого методу у програмному засобі. Розроблені вхідні та вихідні параметри програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях. Виконані обчислення вхідних параметрів для обробки даних у програмному засобі, що

виконані за допомогою різних методів та інструментів, залежно від конкретної задачі та програмного середовища. При цьому, ретельно розглянуті деякі загальні кроки та методи для обчислення таких вхідних параметрів.

Список використаної літератури

1. Sychenko V., Khoshaba O. General methods for investigating performance bottlenecks in game software //III Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів "Комютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023". Збірник матеріалів - Одеса, 2023. – С. 153-154.

2. Khoshaba O. Problems of evaluating and eliminating performance bottleneck in computer games //III Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів "Комп`ютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023". Збірник матеріалів - Одеса, 2023. – С. 133-152.

3. Khoshaba O.M. The problem of identifying performance bottlenecks in distributed structures /Матеріали XXII конференції Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. "Стан, досягнення та перспективи інформаційних систем і технологіх". Одеса, 21-22 березня 2022 р. - Одеса, Видавництво ОНАХТ, 2022 р. - с.99.

4. Khoshaba O.M. Models and criteria for the efficiency of nodes in distributed systems //Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – С. 14-19.

5. Oleksandr Khoshaba, Viktor Grechaninov, Tetiana Molodetska, Anatoliy Lopushanskyi & Kostiantyn Zavertailo. Study of the Workspace Model in Distributed Structures Using CAP Theorem. Mathematical Modeling and Simulation of Systems, Volume 667, Springer International Publishing, pp. 229–242, 2023. doi: 10.1007/978-3-031-30251-0_18

6. Oleksandr Khoshaba, Viktor Grechaninov, Anatoliy Lopushanskyi, Kostiantyn Zavertailo. Studying the Dynamic Bottlenecks of a Load Balancer in Distributed Systems. Lecture Notes in Networks and Systems, Volume 344, Springer International Publishing, pp. 199–211, 2022. doi: 10.1007/978-3-030-89902-8_16

7. Khoshaba, O., Grechaninov, V., Molodetska, T., Lopushanskyi, A., Zavertailo, K. Study of Impact and Reflected Waves in Computer Echolocation. In: Karuppusamy, P., García Márquez, F.P., Nguyen, T.N. (eds) Ubiquitous Intelligent Systems. ICUIS 2021. Smart Innovation, Systems and Technologies, vol 302., pp. 543-557, 2022. Springer, Singapore. https://doi.org/10.1007/978-981-19-2541-2_45

8. Oleksandr Khoshaba, Viktor Grechaninov, Anatoliy Lopushanskyi, Kostiantyn Zavertailo. Modeling the Process of Loading Impact on Web Servers in Computer Systems. 2021 IEEE 3rd Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2021, pp. 519-524, doi: 10.1109/UKRCON53503.2021.9575639.

9. Viktoriia V. Voitko, Svitlana V. Bevz, Sergii M. Burbelo, Pavlo V. Stavytskyi, Oleksandr M. Khoshaba, etc. Analysis of the development approaches of the system of audio synthesis and recognition with the option of using photonic processors. Proceedings of SPIE - The International Society for Optical Engineering. Vol.120400N, 2021. doi: 10.1117/12.2611464

10. Sergey I. Vyatkin, Olexander N. Romanyuk, Oleksandr M. Khoshaba, etc. Modeling the passage of light through surfaces and volumes. Proceedings of SPIE - The International Society for Optical Engineering. Vol.12040, 2021. doi: 10.1117/12.2617356

11. Khoshaba O.M. The main aspects of using gamification in the education process /Матеріали І Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. Частина І. Одеса, 25-26 березня 2021 р. - Одеса, Видавництво ОНАХТ, 2021 р. - с.7-9.

12. Хошаба О.М., Гречанінов В.Ф., Лопушанський А.В. Особливості аналізу та проектування взаємодій інформаційних систем в сервіс-орієнтованих архітектурах /На шляху до індустрії 4.0:інформаційні технології, моделювання, штучний інтелект, автоматизація. Монографія.Одеса, 2021. -С.232-242.

13. Завертайло К.С., Хошаба О.М. Підвищення продуктивності в операційних системах шляхом вирішення конфліктних ситуацій між процесами /На шляху до індустрії 4.0:інформаційні технології, моделювання, штучний інтелект, автоматизація. Монографія.Одеса, 2021. -С.115-125.

14. Khoshaba O. M. Automated system to support the process of servicing maintenance stations [Електронний ресурс] / O. M. Khoshaba, D. Y. Chernega // Матеріали L науково-технічної конференції підрозділів ВНТУ, Вінниця, 10-12 березня 2021 р. – Електрон. текст. дані. – 2021. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/12252.

15. Oleksandr Khoshaba, Vitalii Lytvynov, Viktor Grechaninov & Kostiantyn Zavertailo. Performance of the Reverse Load Balancer Method in Cluster and Cloud Infrastructures. Advances in Intelligent Systems and Computing, Springer International Publishing, pp. 186–196, 2020. doi = 10.1007/978-3-030-58124-4_18

16. Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. Spark sql: Relational data processing in spark. In Proceedings of the 2021 ACM international conference on management of data (SIGMOD), ACM, 2021. - pp. 1383–1394.

17. Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In International Conference on Language Resources and Evaluation (LREC), volume 10, 2020. pp. 2200–2204.

18. Mona Attariyan and Jason Flinn. Automating configuration troubleshooting with dynamic information flow analysis. In USENIX Symposium on Operating Systems Design and Implementation (OSDI), volume 10, 2020.-pp. 31–34.

19. Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. Synthesis lectures on computer architecture, 8(3):1–154, 2020.-pp.126-129.

20. Muli Ben-Yehuda, David Breitgand, Michael Factor, Hillel Kolodner, Valentin Kravtsov, and Dan Pelleg. Nap: a building block for remediating performance bottlenecks via black box network analysis. In Proceedings of the 6th international conference on Autonomic computing (ICAC), ACM, 2019. pp. 179–188.

21. Ivan Beschastnikh, Patty Wang, Yuriy Brun, and Michael D Ernst. Debugging distributed systems. Queue, 14(2):50, 2016. pp. 171-178.

22. Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. Site Reliability Engineering: How Google Runs Production Systems. " O'Reilly Media, Inc.", 2021. P. 194.

23. Betsy Beyer, Niall Richard Murphy, David K Rensin, Kent Kawahara, and Stephen Thorne. The Site Reliability Workbook: Practical Ways to Implement SRE. " O'Reilly Media, Inc.", 2018. pp. 163-172.

24. Christopher M Bishop. Pattern recognition and machine learning. Springer Science, Business Media, 2021. P. 196.

25. Peter Bodík, Rean Griffith, Charles A Sutton, Armando Fox, Michael I Jordan, and David A Patterson. Statistical machine learning makes automatic control practical for internet datacenters. USENIX Workshop on Hot Topics in Cloud Computing (HotCloud), 2019.-pp. 12–18.

26. Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2021. P. 284.

27. Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. 2019. P. 295.

28. Marcio Castro, Luis Fabricio Wanderley Goes, Christiane Pousa Ribeiro, Murray Cole, Marcelo Cintra, and Jean-Francois Mehaut. A machine learning-based approach for thread mapping on transactional memory applications. In 18th International Conference on High Performance Computing (HiPC), IEEE, 2021. pp. 44–52.

29. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3):15, 2019.-P. 391.

30. Loon-Been Chen and I-Chen Wu. Detection of summative global predicates. IEICE Transactions on Information and Systems, 86(5):976–980, 2018.-P.194.

31. Mike Chen, Alice X Zheng, Jim Lloyd, Michael I Jordan, and Eric Brewer. Failure diagnosis using decision trees. In Proceedings of International Conference on Autonomic Computing (ICAC), IEEE, 2018.-pages 36–43.

32. I-Hsin Chung, Guojing Cong, David Klepacki, Simone Sbaraglia, Seetharami Seelam, and Hui-Fang Wen. A framework for automated performance bottleneck detection. In IEEE International Symposium on Parallel and Distributed Processing (IPDPS), IEEE, 2018. pp. 182–193.

33. Ira Cohen, Jeffrey S Chase, Moises Goldszmidt, Terence Kelly, and Julie Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In USENIX Symposium on Operating Systems Design and Implementation (OSDI), volume 4, 2022.-pp. 276-287.

34. Література: Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

128

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
19 " вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доцент О.М. Хошаба
"19" 09 2023 р.

Виконав:
студент гр. 2ПІ-22м В. В. Сиченко
"19" 09 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях».

Галузь застосування – дослідження bottlenecks у високопродуктивних обчисленнях.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18 вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності використання програмних засобів з метою зменшення наслідків bottlenecks у високопродуктивних обчисленнях.

Призначення роботи – розробка методів і засобів підвищення ефективності використання програмних засобів з метою зменшення наслідків bottlenecks.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Sychenko V., Khoshaba O. General methods for investigating performance bottlenecks in game software //III Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів "Комютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023". Збірник матеріалів - Одеса, 2023. – С. 153-154.
2. Khoshaba O. Problems of evaluating and eliminating performance bottleneck in computer games //III Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів "Комп`ютерні ігри і мультимедіа як

інноваційний підхід до комунікації - 2023". Збірник матеріалів - Одеса, 2023. – С. 133-152.

3. Khoshaba O.M. The problem of identifying performance bottlenecks in distributed structures /Матеріали XXII конференції Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. "Стан, досягнення та перспективи інформаційних систем і технологіх". Одеса, 21-22 березня 2022 р. - Одеса, Видавництво ОНАХТ, 2022 р. - с.99.

4. Khoshaba O.M. Models and criteria for the efficiency of nodes in distributed systems //Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – С. 14-19.

4. Технічні вимоги

формалізація методів підвищення ефективності обчислювальних систем, визначення продуктивності обчислювальних систем у розподілених системах - не менш 500 запитів за секунду, визначення розміру обробки запитах на обчислювальних системах - не більш ніж 300 мс, визначення кількості відмов в обробці запитів на сервері - не більш ніж 100 відмов за секунду, визначення кількості коллізій на мережевому інтерфейсі серверу - не більш ніж 2 за 5 хвилин, визначення кількості віртуальних машин на одному серверу - не більш ніж 20.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред’являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз сучасних методів та розробки програмних засобів досліджень bottlenecks у високопродуктивних обчисленнях.	19.09.2023-02.10.2023
2	Задачі розробки методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях.	03.10.2023-16.10.2023
3	Визначення вимог щодо програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.	17.10.2023-28.10.2023
4	Розробка вхідних та вихідних параметрів програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях.	29.10.2023-12.11.2023
5	Економічна частина	13.11.2023-01.12.2023

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

132

Додаток Б. Протокол перевірки навчальної (кваліфікаційної) роботи

Назва роботи: **Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях.**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 22м

Науковий керівник: к.т.н. доц. Хошаба О. М.

Unicheck	
Оригінальність	**96,3%**
Схожість	3,7 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях.».

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення: **допустити до захисту**

Автор _____ Сиченко В.В.

Керівник роботи _____ Хошаба О.М.

Додаток В. Лістинг програми

Класи-сутності для обробки даних з бази даних, репозиторії для взаємодії з базою даних та контролери для обробки запитів:

```
@Entity
public class PerformanceData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String performanceMetric;
    // other fields, getters, and setters
}

@Entity
public class VisualizationData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String visualizationMetric;
    // other fields, getters, and setters
}
// Інші класи сутностей

@Entity
public class PerformanceData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String performanceMetric;
```

```
    // other fields, getters, and setters
}

@Entity
public class VisualizationData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String visualizationMetric;
    // other fields, getters, and setters
}
// Інші класи сутностей

public interface PerformanceDataRepository extends
JpaRepository<PerformanceData, Long> {
    // custom queries if needed
}

public interface VisualizationDataRepository extends
JpaRepository<VisualizationData, Long> {
    // custom queries if needed
}
// Інші репозиторії

@RestController
@RequestMapping("/performance")
public class PerformanceDataController {
    private final PerformanceDataRepository repository;

    public PerformanceDataController(PerformanceDataRepository repository) {
```

```
        this.repository = repository;
    }

    // Mapping for different endpoints related to performance data
}

@RestController
@RequestMapping("/visualization")
public class VisualizationDataController {
    private final VisualizationDataRepository repository;

    public VisualizationDataController(VisualizationDataRepository repository) {
        this.repository = repository;
    }

    // Mapping for different endpoints related to visualization data
}
// Інші контролери для решти модулів
```

Файл, що відповідає за властивості системи application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/main_database
spring.datasource.username=userName
spring.datasource.password=passwordName
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

Створення сутностей:

```
@Entity
@Table(name = "performance_data")
```

```
public class PerformanceData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String metricName;
    private Double value;

    // constructors, getters, setters, etc.
}
```

Створення репозиторіїв:

```
@Repository
public interface PerformanceDataRepository extends
JpaRepository<PerformanceData, Long> {
    // Можете додати власні методи, якщо потрібно
}
```

Створення сервісів:

```
@Service
public class ProductivityAnalysisService {
    private final PerformanceDataRepository performanceDataRepository;

    public ProductivityAnalysisService(PerformanceDataRepository
performanceDataRepository) {
        this.performanceDataRepository = performanceDataRepository;
    }

    // Логіка для аналізу продуктивності
}
```

Створення контролерів:

```
@RestController
@RequestMapping("/performance")
public class PerformanceDataController {
    private final PerformanceDataRepository performanceDataRepository;

    public PerformanceDataController(PerformanceDataRepository performanceDataRepository) {
        this.performanceDataRepository = performanceDataRepository;
    }

    // Mapping для збору даних про продуктивність та інші операції
}

@RestController
@RequestMapping("/analysis")
public class ProductivityAnalysisController {
    private final ProductivityAnalysisService productivityAnalysisService;

    public ProductivityAnalysisController(ProductivityAnalysisService productivityAnalysisService) {
        this.productivityAnalysisService = productivityAnalysisService;
    }

    // Mapping для аналізу продуктивності та інші операції
}
// Та інші контролери для решти модулів

@Entity
```

```
@Table(name = "performance_data")
public class PerformanceData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String metricName;
    private Double value;

    // Конструктори, гетери, сетери та інші методи

    public PerformanceData() {
    }

    public PerformanceData(String metricName, Double value) {
        this.metricName = metricName;
        this.value = value;
    }

    // Гетери та сетери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getMetricName() {
        return metricName;
```

```
    }

    public void setMetricName(String metricName) {
        this.metricName = metricName;
    }

    public Double getValue() {
        return value;
    }

    public void setValue(Double value) {
        this.value = value;
    }
}

@Entity
@Table(name = "performance_analysis")
public class PerformanceAnalysis {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String analysisType;
    private String result;

    // Конструктори, гетери, сетери та інші методи

    public PerformanceAnalysis() {
    }
```

```
    public PerformanceAnalysis(String analysisType, String result) {
        this.analysisType = analysisType;
        this.result = result;
    }

    // Гетери та сетери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getAnalysisType() {
        return analysisType;
    }

    public void setAnalysisType(String analysisType) {
        this.analysisType = analysisType;
    }

    public String getResult() {
        return result;
    }

    public void setResult(String result) {
        this.result = result;
    }
}
```

```
@Entity
@Table(name = "data_visualization")
public class DataVisualization {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String visualizationType;
    private String data;

    // Конструктори, гетери, сетери та інші методи

    public DataVisualization() {
    }

    public DataVisualization(String visualizationType, String data) {
        this.visualizationType = visualizationType;
        this.data = data;
    }

    // Гетери та сетери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
```

```
    public String getVisualizationType() {
        return visualizationType;
    }

    public void setVisualizationType(String visualizationType) {
        this.visualizationType = visualizationType;
    }

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }
}

@Entity
@Table(name = "platform_support")
public class PlatformSupport {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String platformName;
    private String supportedFeatures;

    // Конструктори, гетери, сетери та інші методи

    public PlatformSupport() {
```

```
}

public PlatformSupport(String platformName, String supportedFeatures) {
    this.platformName = platformName;
    this.supportedFeatures = supportedFeatures;
}

// Гетери та сетери
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getPlatformName() {
    return platformName;
}

public void setPlatformName(String platformName) {
    this.platformName = platformName;
}

public String getSupportedFeatures() {
    return supportedFeatures;
}

public void setSupportedFeatures(String supportedFeatures) {
    this.supportedFeatures = supportedFeatures;
```

```
    }
}

@Entity
@Table(name = "settings_management")
public class SettingsManagement {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String settingName;
    private String settingValue;

    // Конструктори, гетери, сетери та інші методи

    public SettingsManagement() {
    }

    public SettingsManagement(String settingName, String settingValue) {
        this.settingName = settingName;
        this.settingValue = settingValue;
    }

    // Гетери та сетери
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
```

```
    }

    public String getSettingName() {
        return settingName;
    }

    public void setSettingName(String settingName) {
        this.settingName = settingName;
    }

    public String getSettingValue() {
        return settingValue;
    }

    public void setSettingValue(String settingValue) {
        this.settingValue = settingValue;
    }
}

@Entity
@Table(name = "result_saving")
public class ResultSaving {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String resultName;
    private String resultValue;

    // Конструктори, гетери, сетери та інші методи
```

```
public ResultSaving() {
}

public ResultSaving(String resultName, String resultValue) {
    this.resultName = resultName;
    this.resultValue = resultValue;
}

// Гетери та сетери
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getResultName() {
    return resultName;
}

public void setResultName(String resultName) {
    this.resultName = resultName;
}

public String getResultValue() {
    return resultValue;
}
```

```
    public void setResultValue(String resultValue) {
        this.resultValue = resultValue;
    }
}

@Entity
@Table(name = "real_time_monitoring")
public class RealTimeMonitoring {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String monitoringType;
    private String monitoringData;

    // Конструктори, гетери, сетери та інші методи

    public RealTimeMonitoring() {
    }

    public RealTimeMonitoring(String monitoringType, String monitoringData) {
        this.monitoringType = monitoringType;
        this.monitoringData = monitoringData;
    }

    // Гетери та сетери
    public Long getId() {
        return id;
    }
```

```
    public void setId(Long id) {
        this.id = id;
    }

    public String getMonitoringType() {
        return monitoringType;
    }

    public void setMonitoringType(String monitoringType) {
        this.monitoringType = monitoringType;
    }

    public String getMonitoringData() {
        return monitoringData;
    }

    public void setMonitoringData(String monitoringData) {
        this.monitoringData = monitoringData;
    }
}
```

Додаток Г. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Слайд 1 – Тема, автор, науковий керівник бакалаврської дипломної роботи:

Розробка методів та програмних засобів дослідження bottlenecks у високопродуктивних обчисленнях

Виконав:
студент групи 2ПІ-22м

Сиченко В. В.

Керівник:
к.т.н., доц. каф. ПЗ Хошаба О.М.

Слайд 2 – Мета, об'єкт та предмет дослідження:

Метою роботи є підвищення ефективності використання програмних засобів з метою зменшення наслідків bottlenecks у високопродуктивних обчисленнях.

Об’єктом дослідження процеси появи та розвитку bottlenecks у високопродуктивних обчисленнях.

Предметом дослідження є методи та засоби підвищення ефективності використання програмних засобів з метою зменшення наслідків bottlenecks у високопродуктивних обчисленнях.

Слайд 3 – Задачі дослідження:

Основними задачами дослідження є:

- виконати обгрунтування методу аналізу дослідження bottlenecks у високопродуктивних обчисленнях та розглянути необхідність використання сучасних методів та інструментів розробки програмних засобів;
- розробити задачі розробки та виконано аналіз сучасних методів та програмних засобів дослідження bottlenecks;
- розглянути загальну характеристику та дослідити особливість використання методів дослідження bottlenecks;
- запропонований метод діагностики bottlenecks у програмному засобі;
- визначити вимоги щодо функціонування програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях;
- розробити структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks;
- виконати реалізацію розробленого методу у програмному засобі;
- розробити вхідні та вихідні параметри програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях;
- виконати обчислення вхідних параметрів для обробки даних у програмному засобі.

Слайд 4 – Актуальність розробки:

Дослідження bottlenecks у високопродуктивних обчисленнях має велику наукову, практичну значимість та актуальність.

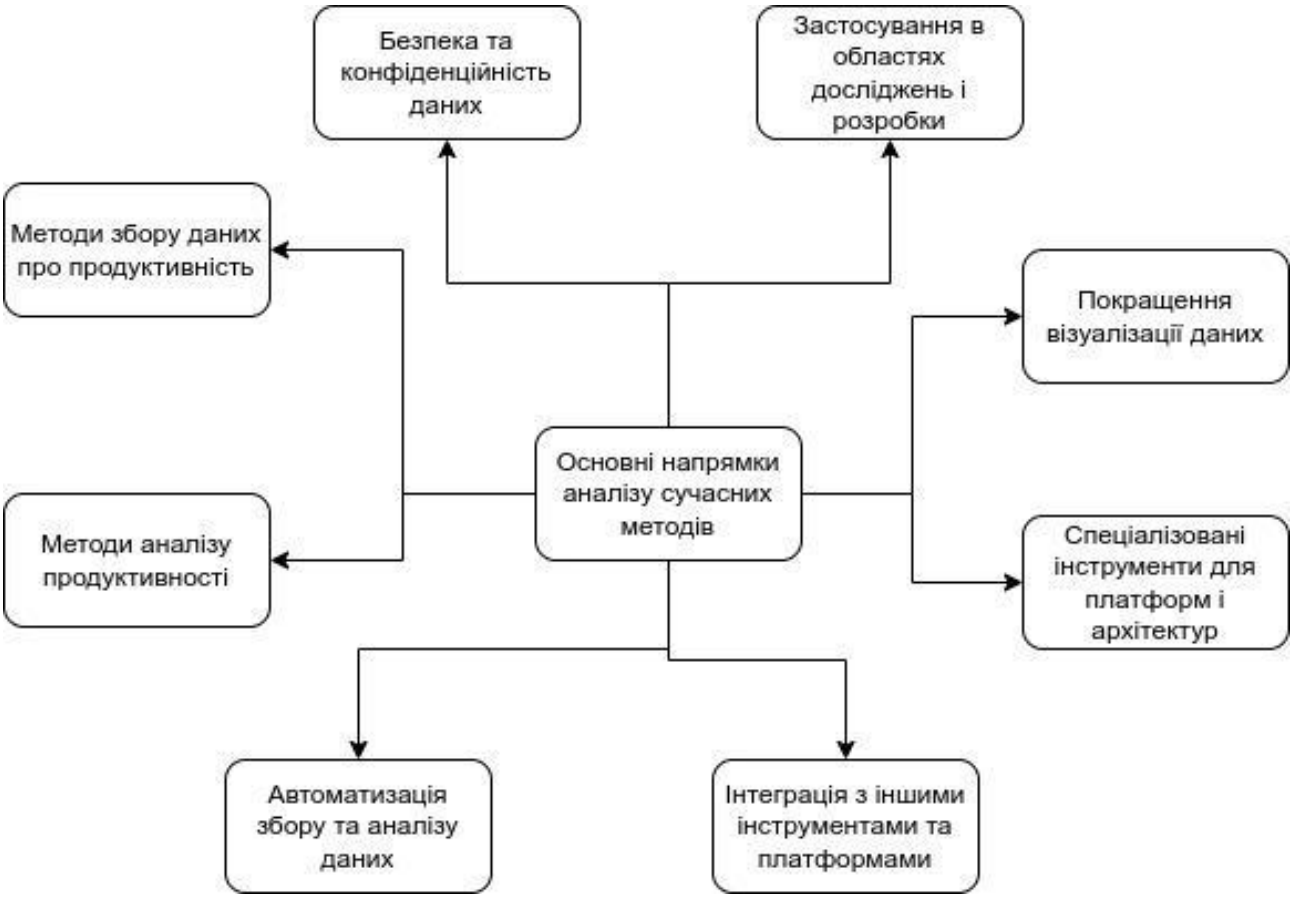
Це пов'язано з тим, що з ростом технологічного прогресу і виникненням нових архітектур обчислювальних систем з'являються нові вузькі місця. Дослідження їх може допомогти адаптувати програмні рішення до нових технологій та визначити оптимальні стратегії для їх використання.

Також, в багатьох наукових областях, включаючи фізику, біологію, кліматологію та інші галузі наук, високопродуктивні обчислення є ключовим інструментом.

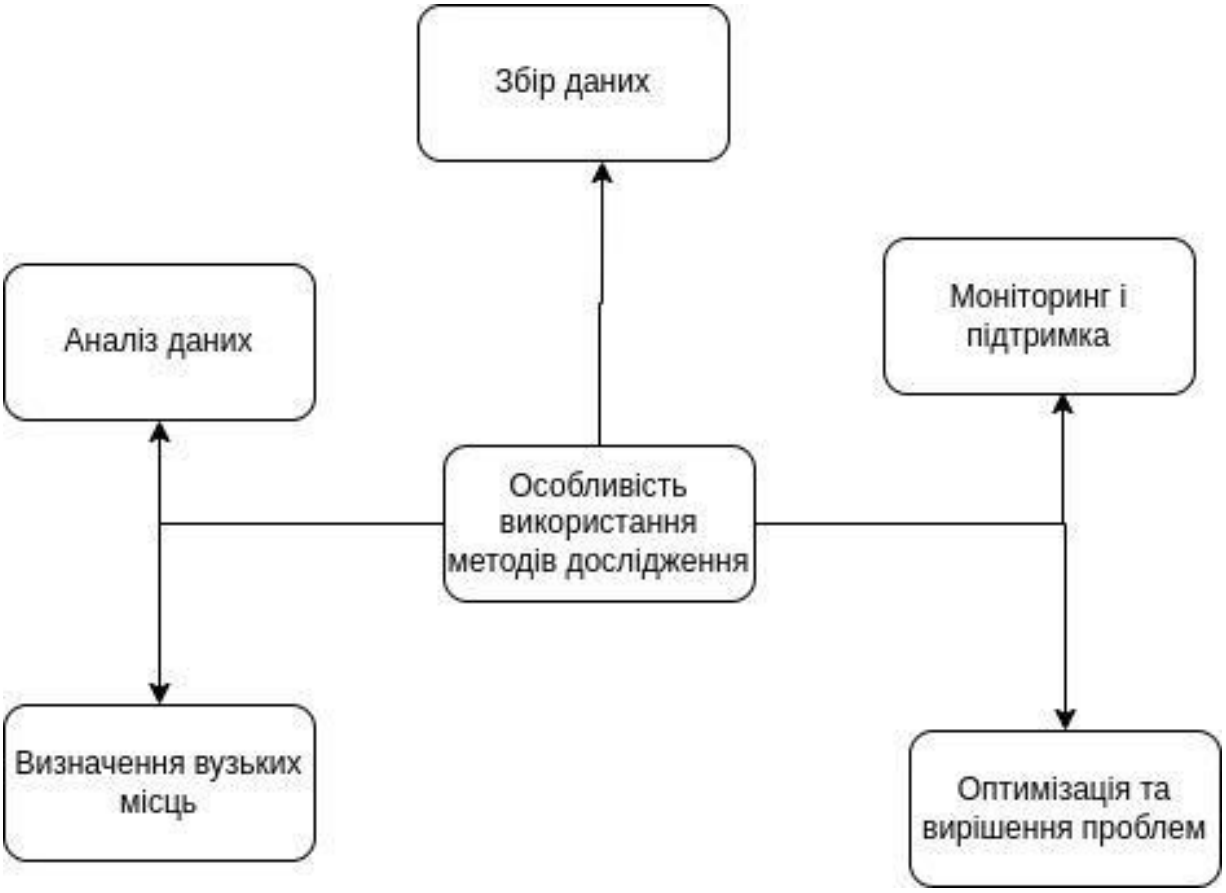
Тому, збільшення продуктивності дозволяє вченим аналізувати дані та виконувати моделювання в більших масштабах і з великою точністю.

Слайд 5 – Основні задачі розробки методів та програмних засобів дослідження bottlenecks

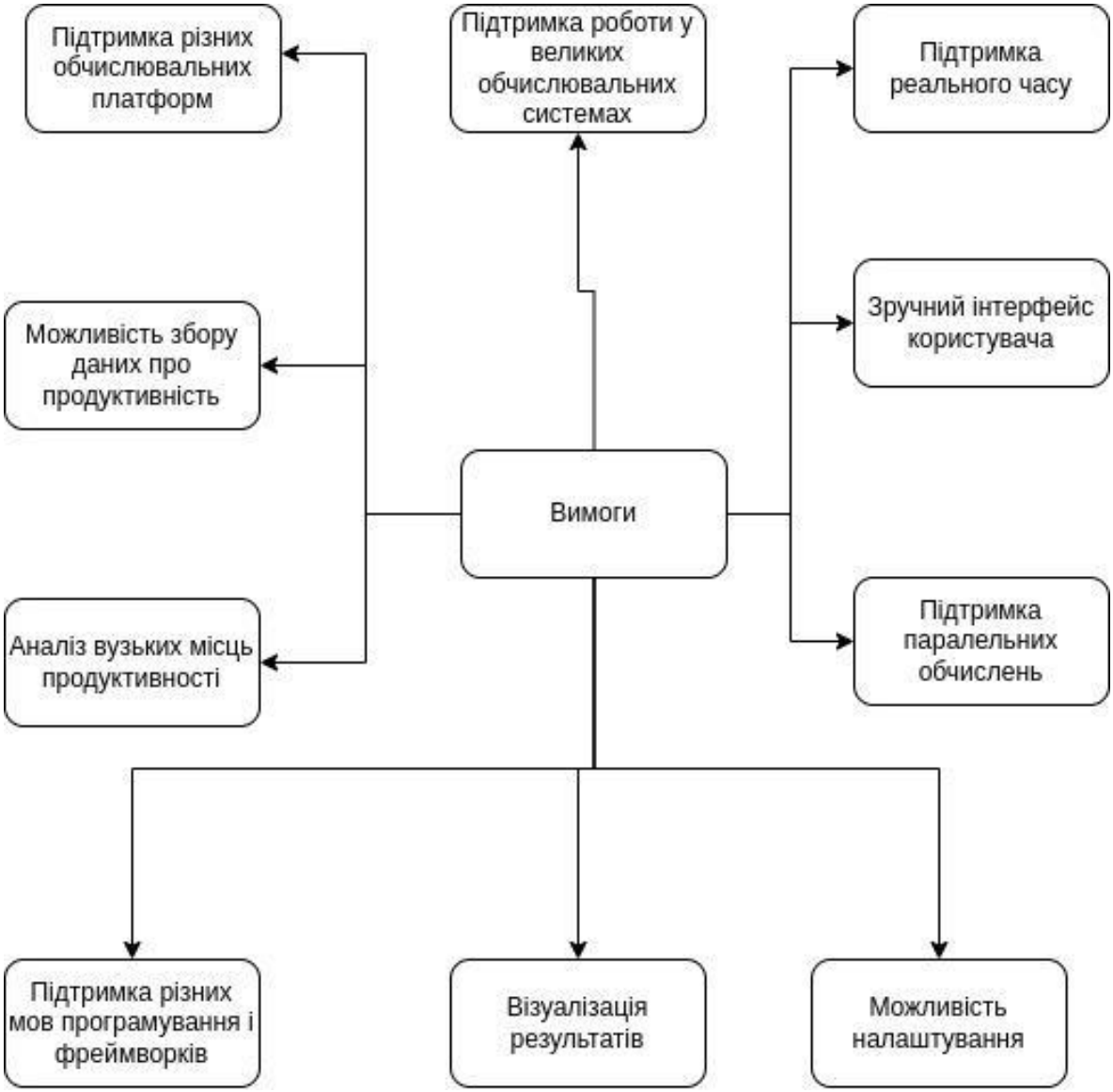
Слайд 6 — Аналіз сучасних методів та розробки програмних засобів для дослідження bottlenecks



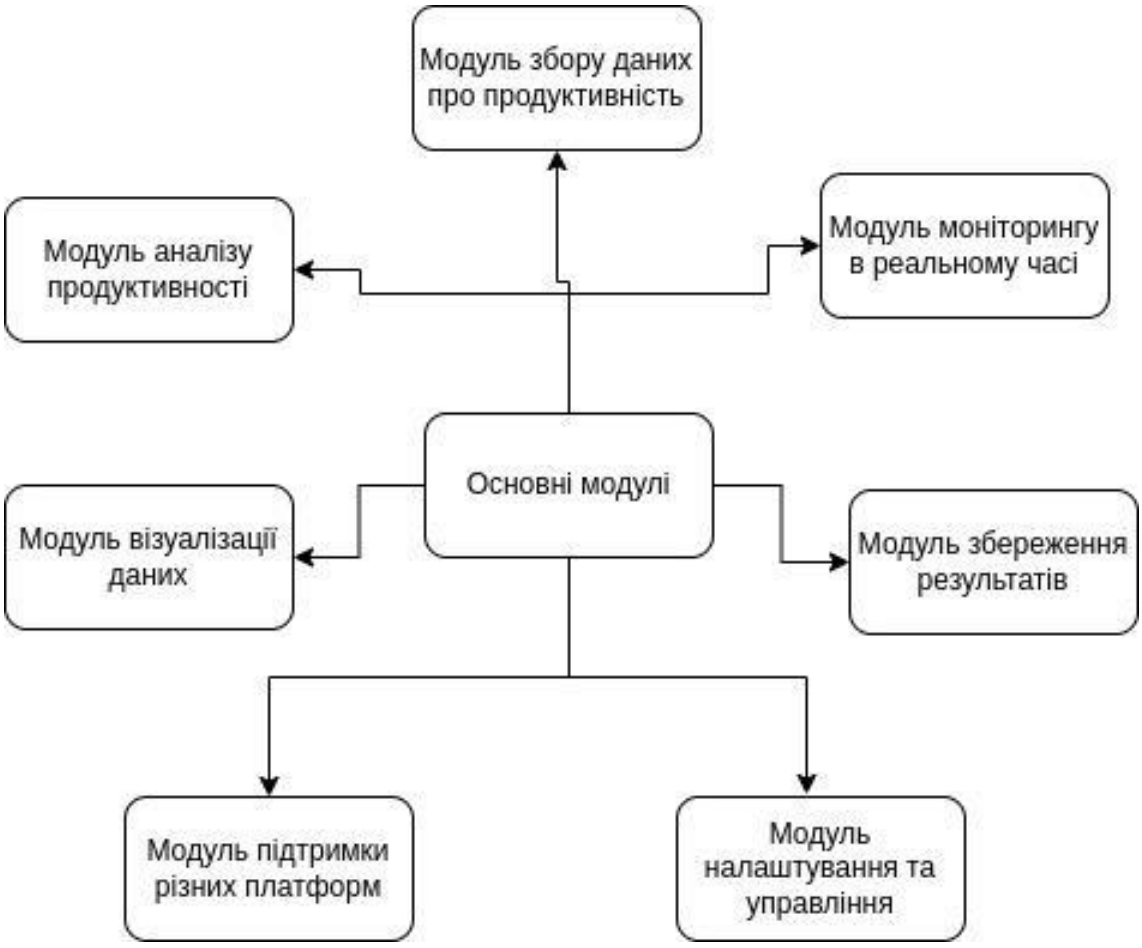
Слайд 7 — Особливість використання методів дослідження bottlenecks



Слайд 8 — Вимоги щодо розробки програмного засобу дослідження bottlenecks



Слайд 9 – Структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks



Слайд 10 — Обчислення вхідних параметрів для обробки даних у програмному засобі

Слайд 11 – Наукова новизна одержаних результатів:

- вперше запропоновано метод використання системи сповіщень про негативні наслідки розвитку bottlenecks, на основі отримання повідомлень про критичні точки, що дозволяє приймати ефективні рішення в галузі горизонтального масштабування обчислювальних ресурсів (на вузлах розподіленої системи);

- здобуло подальший розвиток моніторинг використання інформаційних ресурсів, де збираються дані в реальному режимі часу про виконання програмних модулів в робочому середовищі високопродуктивних обчислень на основі відправлення попереджувальних повідомлень до системи сповіщень, яка допомагає виявляти небезпечний розвиток навантажувальних впливів.

Слайд 12 – Висновки:

В роботі:

- виконано обгрунтування методу аналізу дослідження bottlenecks у високопродуктивних обчисленнях та розглянути необхідність використання сучасних методів та інструментів розробки програмних засобів;
- розроблено задачі розробки та виконано аналіз сучасних методів та програмних засобів дослідження bottlenecks;
- розглянуто загальну характеристику та дослідити особливість використання методів дослідження bottlenecks;
- запропоновано метод діагностики bottlenecks у програмному засобі;
- визначені вимоги щодо функціонування програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях;
- розроблені структурно-функціональні особливості основних модулів програмного засобу дослідження bottlenecks;
- виконані реалізацію розробленого методу у програмному засобі;
- розроблені вхідні та вихідні параметри програмного засобу дослідження bottlenecks у високопродуктивних обчисленнях;
- виконані обчислення вхідних параметрів для обробки даних у програмному засобі.

Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів "Комютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2023", (Одеса, 2023).

Основні результати досліджень опубліковано в науковій праці у матеріалах цієї конференції.