НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

«На правах рукопису»

УДК _____

ДО ЗАХИСТУ ДОПУЩЕНО:

Завідувач кафедри

_____ Вадим МУХІН

«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою

“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”

зі спеціальності 122 "Комп'ютерні науки"

на тему: «Дослідження продуктивності мікросервісних архітектур через кешування даних»

Виконала:
студентка VI курсу, групи ДА-21мп
Хоміч Ліна Ігорівна _____

Керівник:
ас. Яременко В. С. ___ _____

Рецензент:
доц. к.т.н. Недашківська Н.І _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка_____

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу

Кафедра системного проектування

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 "Комп'ютерні науки"

Освітньо-професійна програма – "Інтелектуальні сервіс-орієнтовані розподілені обчислювання"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Вадим МУХІН

25 червня 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Хоміч Ліні Ігорівні

1. Тема дисертації «Дослідження продуктивності мікросервісних архітектур через кешування даних», науковий керівник дисертації Яременко Вадим Сергійович, асистент, затверджені наказом по університету від __ листопада 2023 р. № ____
2. Термін подання студентом дисертації – 15 січня 2024 р.
3. Об’єкт дослідження - продуктивність мікросервісів, досліджені через кешування даних, реалізованих за допомогою програмування на Java
4. Вихідні дані - проаналізовані метрики, що базуються на зібраних даних під час симуляцій.
5. Перелік завдань, які потрібно розробити

1) Аналіз теоретичного матеріалу за напрямком продуктивності мікросервісів через кешування даних.
2) Аналіз теоретичного матеріалу за напрямком CDN провайдери та їх роль у глобальній архітектурі.
3) Розробка моделі симуляційного середовища для проведення досліджень з різними стратегіями кешування.
4) Аналіз результатів зібраних під час симуляцій та порівняти стратегії за метриками.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу - презентація

7. Орієнтовний перелік публікацій - підготовка матеріалу до публікації статті у фаховому виданні

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання – 25 червня 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	25 червня 2023	
2	Ознайомлення з нормативними документами	01 вересня 2023	
3	Аналіз теоретичного матеріалу за напрямком продуктивності мікросервісів через кешування даних.	22 вересня 2023	
4	Аналіз теоретичного матеріалу за напрямком CDN провайдери та їх роль у глобальній архітектурі.	1 жовтня 2023	
5	Розробка моделі симуляційного середовища для проведення досліджень з різними стратегіями кешування.	15 листопада 2023	
6	Аналіз результатів зібраних під час симуляцій та порівняти стратегії за метриками.	25 листопада 2023	
7	Оформлення дипломної роботи	1 грудня 2023	
8	Отримання допуску до захисту та подача роботи в ДЕК	11 січня 2024	
9	Захист дисертації	15 січня 2024	

Студент Хоміч Л.І.

Науковий керівник Яременко В.С.

РЕФЕРАТ

Магістерська дисертація Хоміч Ліни Ігорівни на тему "Дослідження продуктивності мікросервісних архітектур через кешування даних" обсягом 86 сторінок, включає 9 ілюстрацій, 27 таблиць та використовує 16 джерел.

Актуальність дослідження випливає з необхідності оптимізації продуктивності мікросервісних архітектур за допомогою ефективних стратегій кешування даних, у зв’язку з швидкозмінюючим інформаційним середовищем та стрімким розвитком мікросервісних архітектур і CDN мереж.

Мета дослідження - систематичний аналіз та оптимізація продуктивності мікросервісних архітектур через впровадження ефективних стратегій кешування даних. Робота спрямована на визначення та порівняння стратегій кешування в розподіленій системі та розроблення об'єктивного методу їх вимірювання.

Предмет дослідження - продуктивність мікросервісних архітектур, зокрема, ефективність стратегій кешування даних в їхньому функціонуванні.

Об'єктом дослідження визначаються мікросервісні архітектури, які використовуються у розподілених системах.

Наукова новизна дослідження полягає в комплексному аналізі та оцінці впливу стратегій кешування на продуктивність мікросервісних архітектур у контексті CDN-мережі. Результати дослідження дозволяють виявити оптимальні стратегії кешування, сприяючи підвищенню ефективності функціонування мікросервісів у розподілених системах.

Апробація результатів проводилася під час II Всеукраїнської науково-практичної конференції "Системні науки та інформатика" з нагоди 125-річчя КПІ ім. Ігоря Сікорського.

Ключові слова: стратегії кешування даних, розподілені системи, CDN-мережі, продуктивність, оптимізація.

ABSTRACT

Master's thesis by Lina Ihorivna Khomich on the topic "Research into performance of microservice architectures through data caching" spans 86 pages, including 9 illustrations, 27 tables, and references 16 sources.

The relevance of the research arises from the need to optimize the performance of microservices architectures through effective data caching strategies, given the rapidly changing information environment and the swift development of microservices architectures and CDN networks.

The aim of the study is a systematic analysis and optimization of the performance of microservices architectures through the implementation of effective data caching strategies. The work focuses on identifying and comparing caching strategies in distributed systems and developing an objective method for their measurement.

The subject of the research is the performance of microservices architectures, particularly the effectiveness of data caching strategies in their operation. The object of the study is defined as microservices architectures used in distributed systems.

The scientific novelty of the research lies in a comprehensive analysis and assessment of the impact of caching strategies on the performance of microservices architectures in the context of CDN networks. The results of the study help identify optimal caching strategies, contributing to the improved efficiency of microservices' operation in distributed systems.

The research findings were presented and discussed during the II All-Ukrainian Scientific-Practical Conference "System Sciences and Informatics" on the occasion of the 125th anniversary of Igor Sikorsky Kyiv Polytechnic Institute.

Keywords: data caching strategies, distributed systems, CDN networks, performance, optimization.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

- CDN – Content Delivery Network, географічно розподілена мережева інфраструктура, що дозволяє оптимізувати доправлення та розповсюдження контенту кінцевим користувачам в мережі Інтернет.
- Кеш - особлива швидкісна пам'ять або частина оперативної пам'яті, де зберігаються копії часто використовуваних даних. Забезпечує до них швидкий доступ. Кеш-пам'ять зберігає вміст і адресу даних, до яких часто звертається процесор.
- TTL – Time To Live, максимальний період часу або кількість ітерацій або переходів, за який кеш може існувати до свого зникнення.
- Симуляційне середовище - Віртуальне оточення, яке моделює реальні умови для проведення дослідження або експерименту без прямого впливу на реальні системи.
- SDK (Набір засобів розробки програмного забезпечення) - Збірка інструментів та бібліотек, які дозволяють розробникам створювати програмне забезпечення для певної платформи чи середовища.

ВСТУП

В епоху стрімкого технологічного прогресу, коли мікросервісні архітектури та розподілені системи стають критичними складовими розвитку програмного забезпечення, питання продуктивності та ефективності стають вирішальними. Нинішнє інформаційне середовище, яке характеризується стрімкими змінами та високим рівнем конкуренції, ставить перед розробниками завдання створення систем, які не лише відповідають сучасним вимогам, але й забезпечують високу продуктивність та гнучкість.

Одним із ключових та водночас складних аспектів удосконалення функціональності мікросервісних архітектур є управління даними та оптимізація швидкодії. У цьому контексті кешування даних виступає як потужний інструмент для поліпшення продуктивності, забезпечуючи швидкий та ефективний доступ до інформації.

Актуальність дослідження випливає із стрімкого розвитку мікросервісних архітектур та зростання використання CDN-мереж у розподілених системах. Оптимізація продуктивності цих систем стає важливим завданням для забезпечення надійності та швидкості обробки інформації.

Мета дослідження полягає в систематичному аналізі та вдосконаленні продуктивності мікросервісних архітектур через ефективне використання стратегій кешування даних. Детальний огляд та порівняльний аналіз різних стратегій кешування в контексті розподілених систем дозволять визначити оптимальні підходи до оптимізації функціонування мікросервісів.

Предметом дослідження виступають мікросервісні архітектури в розподілених системах, які використовуються у зв'язку із сучасними вимогами до розробки програмного забезпечення.

Дослідження несе наукову новизну, оскільки воно пропонує комплексний аналіз та визначення оптимальних стратегій кешування в мікросервісних архітектурах у контексті CDN-мереж. Результати дослідження не лише розкриють вплив різних стратегій кешування на продуктивність систем, але й визначать рекомендації для ефективного впровадження цих стратегій у практиці розробки програмного забезпечення.

Досягнення цілей дослідження сприятиме підвищенню якості та ефективності мікросервісних архітектур, а також знайде практичне застосування у розробці розподілених систем та мереж CDN, підвищуючи конкурентоспроможність програмного забезпечення в умовах сучасного інформаційного середовища.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СТРАТЕГІЙ КЕШУВАННЯ У МІКРОСЕРВІСНІЙ АРХІТЕКТУРІ

У цьому розділі проводиться глибокий аналіз предметної області, що стосується стратегій кешування в мікросервісних архітектурах. Дослідження охоплює вивчення основних понять, переваги використання мікросервісів порівняно з монолітними системами, аналіз різних стратегій кешування та їхній вплив на продуктивність систем. Також розглядається роль різних CDN-провайдерів у глобальній архітектурі.

Метою даного розділу є створення теоретичного фундаменту, який дозволить краще зрозуміти сутність мікросервісних архітектур, їхню оптимізацію за допомогою стратегій кешування та вплив CDN-мереж на їх функціонування. Вивчення цих аспектів стане важливим етапом для подальших досліджень та розробки оптимальних підходів до управління даними в розподілених системах.

1.1 Огляд ключових аспектів мікросервісних архітектур та їх переваг над традиційними монолітними системами

Концепції мікросервісних архітектур та їх переваги в контексті порівняння з традиційними монолітними системами є ключовою складовою предметної області дослідження, тому насправді важливо дати чітке визначення наведених архітектур та розглянути головні характеристики, що відрізняють їх від монолітних рішень, розкрити переваги, що можуть бути отримані завдяки переходу до мікросервісної архітектури у порівнянні з монолітними системами.

Мікросервісна архітектура визначається як підхід до структурування програмного забезпечення, в якому великі додатки розбиваються на компоненти – мікросервіси. Кожен з цих сервісів має свою чітко визначену

функціональність та працює як автономний процес та спілкується з рештою, використовуючи прості та швидкі протоколи передачі даних. Ці сервіси будуються навколо бізнес-потреб і розгортаються незалежно один від одного з використанням зазвичай повністю автоматизованого середовища.[2]

Як і в кожної архітектури можна виділити характеристики за якими з впевненістю можна сказати, що це саме мікросервісний тип. Їх виділяють чотири основних:

1. Розподіленість: Мікросервіси можуть розташовуватися на різних серверах або навіть в різних центрах обробки даних. Це дозволяє їм бути більш масштабованими і стійкими до відмов.
2. Незалежність: Кожен мікросервіс може розроблятися, впроваджуватися та супроводжуватися окремо від інших. Це сприяє розвитку та швидкості впровадження нових функцій.
3. API для комунікації: Мікросервіси спілкуються між собою через API, що дозволяє їм виконувати завдання та обмін даними.
4. Автономність: Кожен мікросервіс може бути автономним і використовувати власні технології та мови програмування.[5]

Розгляд характеристик мікросервісної архітектури дозволяє нам глибше зрозуміти її фундаментальні особливості. Цей розклад на окремі, самостійні компоненти, які взаємодіють через відкриті API, дозволяє досягти гнучкості та легкості управління. Така структура також сприяє вищій швидкості розробки та забезпечує оптимізацію ресурсів завдяки ефективному масштабуванню. Тепер, перейдемо до розгляду конкретних переваг, які випливають з цих характеристик.

Масштабованість: мікросервіси пропонують можливість гнучкого масштабування окремих компонентів системи, забезпечуючи оптимізацію продуктивності та раціональне використання ресурсів.[1] У монолітних же

системах масштабування є дуже важким процесом, інколи, навіть, неможливим.

Швидка розробка та впровадження: відокремленість мікросервісів дає розробникам можливість працювати над окремими компонентами незалежно один від одного. Це сприяє прискоренню процесу розробки та оперативному впровадженню нових функцій. З точки зору моноліту, зазвичай потрібно великий обсяг роботи для розробки та впровадження нових функцій.

Збільшена надійність: відсутність централізованої точки відмови та можливість ізолювати проблеми в окремих сервісах роблять систему більш надійною. Навіть у разі відмови одного сервісу, інші можуть продовжити працювати без перерв, а в централізованій архітектурі це є болючим місцем - помилки або відмови в одній частині системи можуть впливати на всю систему.

Легкість у виправленні помилок: розділення мікросервісів спрощує виявлення та виправлення помилок. Якщо помилка виникає в одному сервісі, вона не має впливу на інші компоненти системи, що сприяє ефективній діагностиці та виправленню недоліків.

У контексті порівняння з монолітними системами, де весь додаток розглядається як єдина одиниця, мікросервісна архітектура виходить за рамки традиційного підходу та пропонує більш гнучке, масштабоване та надійне рішення для розробки програмного забезпечення.

Відмінності полягають у розбитті системи на автономні компоненти – мікросервіси, які можуть функціонувати незалежно один від одного. Це забезпечує високу гнучкість в управлінні та розвитку окремих сервісів.[7] Масштабованість дозволяє легко розширювати окремі частини системи, щоб забезпечити оптимальну продуктивність. Висока надійність досягається

завдяки ізоляції мікросервісів, що дозволяє уникнути впливу невдачі в одному компоненті на інші.

Таким чином, вивчення мікросервісних архітектур підтверджує їхню привабливість як стратегії розробки програмного забезпечення, особливо в умовах сучасного швидкозмінюючого інформаційного середовища.

1.2 Вплив CDN провайдерів на глобальному рівні для підтримки продуктивності мікросервісних архітектур

Впровадження мікросервісних архітектур у наш час ставить перед нами завдання ефективного управління даними на глобальному рівні, оскільки це визначальний чинник оптимальної продуктивності та швидкодії. Одним з ключових компонентів, що впливає на розподіл та доступність даних, є CDN-провайдери – компанії, які надають послуги глобального розподілення контенту.

CDN-провайдери володіють потужними глобальними мережами серверів, розташованими у стратегічних точках по всьому світу. Це дозволяє нам реалізувати локальне кешування даних на вузлах CDN-мережі, оптимізуючи тим самим доступ до необхідного контенту та покращуючи продуктивність мікросервісів. Це стратегічне розташування дає можливість оптимізувати час доступу до контенту, зменшуючи латентність та підвищуючи продуктивність. Кешування на рівні CDN дозволяє мікросервісам ефективно взаємодіяти з кінцевими користувачами, які знаходяться в різних частинах світу.[3]

Ефективне кешування в мікросервісних архітектурах вимагає не лише правильно налаштованих стратегій, але й активної взаємодії з CDN-провайдерами. Вони відіграють ключову роль у підтримці кешування, забезпечуючи швидкий доступ до даних та зниження навантаження на центральні сервери. Розглянемо аспекти цієї взаємодії:

1. Локальне зберігання кешованих даних: CDN-провайдери забезпечують локальне зберігання кешованих даних на своїх серверах, розташованих у різних частинах світу. Це дозволяє мікросервісам отримувати доступ до актуальних даних без потреби завантаження їх з центрального сервера.

2. Географічна розсилка контенту: CDN-провайдери оптимізують глобальний розподіл контенту, розсилаючи кешовані дані до кінцевих користувачів із найближчого до їхнього регіону сервера. Це не лише прискорює завантаження сторінок, але й зменшує трафік, який потрібно обробляти центральному серверу.

3. Зберігання статичного та динамічного контенту: CDN-провайдери можуть кешувати як статичний, так і динамічний контент. Для мікросервісів, які генерують динамічний контент, це особливо важливо. Взаємодія з CDN дозволяє зберігати результати запитів і прискорювати їх обробку.

4. Контроль та налаштування кешування: CDN-провайдери надають інструменти для контролю та налаштування кешування. Це включає можливість вказати, які дані кешувати, тривалість зберігання кешу, та інші параметри. Такий підхід дозволяє оптимізувати кешування під конкретні потреби мікросервісної архітектури.

5. Моніторинг та аналітика використання кешу: CDN-провайдери надають інструменти для моніторингу та аналізу ефективності кешування. Це дозволяє виявляти проблеми, вдосконалювати стратегії кешування та підтримувати високу продуктивність системи.

Таким чином, взаємодія з CDN-провайдерами стає необхідним елементом для успішної реалізації кешування в мікросервісних архітектурах, забезпечуючи оптимальну продуктивність та задоволення потреб користувачів у різних куточках світу.

1.3 Аналіз стратегій кешування та їх ефективного використання в контексті мікросервісних архітектур

Кешування даних є однією з ключових стратегій для оптимізації продуктивності мікросервісних архітектур. Далі ми ретельно розглянемо роль та важливість кешування даних в контексті мікросервісів, а також розкриємо різні підходи та методи, що використовуються для забезпечення ефективного кешування.

Кешування даних дозволяє зберігати копії популярних або часто запитуваних даних та надавати їх замість повторних запитів до бази даних чи інших мікросервісів. Цей підхід допомагає зменшити час відповіді на запити, покращує продуктивність та зменшує навантаження на ресурси.

Сутність кешування даних полягає у зберіганні та використанні копій певних даних або обчислень з метою покращення продуктивності та швидкодії в системах і додатках. У контексті мікросервісних архітектур, де компоненти розділені та розподілені, кешування стає важливим інструментом для оптимізації відповідей на запити та зменшення навантаження на ресурси, такі як бази даних чи інші мікросервіси.

1.3.1 Огляд стратегії локального кешування

Локальне кешування, як ключова стратегія в оптимізації продуктивності мікросервісних архітектур, виявляється особливо ефективним у контексті використання CDN провайдерів. Цей підхід передбачає зберігання копій найважливіших та часто використовуваних даних на різних рівнях мережі, прямо всередині мікросервісів. Кожен мікросервіс управляє своїм власним кешем, що дозволяє ефективно та швидко отримувати доступ до необхідних даних, зменшуючи затримки та поліпшуючи продуктивність.

Механізм роботи локального кешування включає в себе динамічне зберігання часто використовуваних даних та їх швидкий доступ.[8] Кожен

мікросервіс контролює свій кеш, забезпечуючи локалізацію та безпеку даних, і мінімізує навантаження на централізовані ресурси. Механізми синхронізації дозволяють оновлювати дані в локальних кешах у реальному часі, забезпечуючи актуальність інформації. Цей підхід ефективно використовує ресурси, знижує звертання до централізованих ресурсів та сприяє оптимальному функціонуванню системи. Локальне кешування, сумісно з CDN провайдерами, стає необхідною складовою глобальної архітектури, що забезпечує ефективний обмін даними на світовому рівні.

Локальне кешування в мікросервісних архітектурах надає значні переваги для оптимізації продуктивності та забезпечення швидкого доступу до даних. Однією з ключових переваг є можливість зберігання копій найважливіших та часто використовуваних даних безпосередньо всередині мікросервісів.

Цей підхід дозволяє кожному мікросервісу ефективно та незалежно управляти своїм кешем, забезпечуючи швидкий доступ до необхідної інформації та мінімізуючи затримки. Крім того, локальне кешування сприяє локалізації та забезпечує безпеку даних, оскільки кожен мікросервіс має контроль над своїм кешем.

Механізми синхронізації, вбудовані в локальне кешування, дозволяють оновлювати дані в реальному часі, забезпечуючи актуальність інформації. Це важливо для уникнення застарілості даних та забезпечення консистентності інформації у всій системі. Крім того, локальне кешування допомагає знизити навантаження на централізовані ресурси, оскільки мікросервіси можуть надавати необхідні дані без постійного звертання до центральних джерел.

Незважаючи на численні переваги, пов'язані із використанням локального кешування в мікросервісних архітектурах, існують і певні недоліки, які варто враховувати. Одним із основних обмежень є можливість

дублювання даних в різних кешах мікросервісів, що може вести до надмірного використання простору та збільшення ризику виникнення неузгодженостей в інформації.[8]

Локальне кешування можна охарактеризувати за допомогою кількісних характеристик, що визначають його ефективність та вплив на продуктивність мікросервісних архітектур. Обсяг пам'яті для локального кешу визначає, скільки пам'яті відведено для збереження даних на рівні кожного мікросервісу. Цей параметр важливий для оцінки ефективності використання ресурсів та мінімізації витрат на збереження копій інформації.

Час доступу до даних визначає, наскільки швидко мікросервіс може отримати необхідні дані зі свого локального кешу. Ця характеристика вказує на продуктивність кешування та вплив на швидкодію обробки запитів. Швидкий доступ до даних з кешу покращує відгуковий час мікросервісів та загальну продуктивність системи.

Дублювання даних в локальних кешах визначає кількість повторюваних чи схожих даних, які зберігаються у кешах різних мікросервісів. Ця характеристика важлива для уникнення надмірного використання простору та оптимізації обсягу даних, що зберігаються в кешах.

1.3.2 Огляд стратегії централізованого кешування

Стратегія централізованого кешування передбачає використання спільного кешу, який накопичує дані для всіх мікросервісів. Цей підхід може бути ефективним у випадках, коли декілька сервісів спільно використовують однакові дані.

Механізм роботи централізованого кешування визначається його центральною природою, яка передбачає збереження копій часто використовуваних даних на центральному сервері або спеціалізованому кеш-сервері.[9]

Коли мікросервіс подає запит на отримання даних, він спрямовується до централізованого кешу. Система перевіряє, чи необхідні дані вже присутні в кеші. Якщо так, ці дані негайно передаються мікросервісу без необхідності обходження головного сервера або інших розподілених ресурсів.

Оновлення кешу та валідація даних є важливою частиною механізму. Кеш регулярно оновлюється із вихідних даних для забезпечення їхньої актуальності. Механізми валідації гарантують, що дані в кеші залишаються вірними та актуальними, враховуючи їхній час життя.

Основна перевага цієї стратегії полягає в ефективному управлінні ресурсами, оскільки дані зберігаються централізовано. Це спрощує механізм кешування та полегшує синхронізацію даних. Важливою характеристикою є також спрощений процес оновлення та валідації даних, оскільки це може бути здійснено централізовано.

Крім того, централізоване кешування допомагає знизити навантаження на мікросервіси, оскільки вони можуть ефективно отримувати доступ до даних через централізований кеш. Це сприяє збереженню високої продуктивності та оптимізації використання ресурсів. Також, централізоване кешування дозволяє уникнути дублювання даних, забезпечуючи їхню консистентність та ефективне використання простору.

Важливим аспектом є також забезпечення централізованої безпеки, оскільки стратегія передбачає один централізований пункт зберігання даних. Це може бути важливим фактором, особливо для систем, які обробляють конфіденційну інформацію. Обираючи централізоване кешування, слід враховувати конкретні потреби та вимоги мікросервісної архітектури.

Централізоване кешування, хоча і має свої переваги, водночас інтегрує в себе кілька недоліків, які варто ураховувати при виборі стратегії оптимізації продуктивності в мікросервісних архітектурах.

Один із головних аспектів – це ризик створення єдиного пункту витоку. Коли центральний кеш сервер стає недоступним або перенавантаженим, це може призвести до значного зниження продуктивності системи в цілому. Така залежність від одного пункту може бути ризикованою для стабільності та доступності.

Другим важливим аспектом є затримки, які можуть виникнути під час передачі даних через мережу. Це особливо важливо для глобальних мікросервісних архітектур, де великі відстані можуть значно вплинути на час доступу до даних.

Механізми синхронізації даних також можуть створювати проблеми, зокрема конфлікти при одночасному оновленні чи недостатня ефективність синхронізації. Це може призвести до некоректної роботи системи та втрати даних.[11]

Обсяг даних та пропускна здатність є ще однією важливою проблемою. Зі зростанням обсягу даних чи трафіку системи може збільшуватися навантаження на централізований кеш, що призводить до зниження продуктивності.

Також важливо враховувати, що централізоване кешування може ускладнювати збереження стану мікросервісів. Оновлення кешу повинні бути синхронізовані зі змінами в мікросервісах, що може створювати проблеми у відслідковуванні та збереженні консистентності даних.

Кількісні характеристики для оцінки централізованого кешування включають кількість пам'яті, призначеної для централізованого кешу, час, необхідний для доступу до даних, та рівень дублювання інформації. Обсяг централізованого кешу визначає, наскільки ефективно використовується ця пам'ять для задоволення потреб мікросервісів. Час доступу до даних визначає, наскільки швидко мікросервіси можуть отримувати інформацію з

централізованого кешу, що важливо для забезпечення продуктивності мікросервісної архітектури. Дублювання даних в централізованому кеші вимірює рівень повторення інформації та його вплив на використання ресурсів. Оцінка цих характеристик дозволяє визначити ефективність та придатність централізованого кешування для потреб мікросервісної архітектури.

1.3.3 Огляд стратегії розподіленого кешування

Розподілене кешування – це стратегія оптимізації продуктивності, яка передбачає збереження копій часто використовуваних даних на різних вузлах мережі для забезпечення швидкого доступу та мінімізації затримок для кінцевих користувачів.

Основний механізм роботи розподіленого кешування включає кілька ключових етапів. По-перше, визначаються ключі кешування, які вказують, які дані підлягають збереженню в кеші на різних вузлах. Ці ключі можуть бути асоційовані з певними запитами чи областями даних.[13]

Далі кеші розташовані на різних серверах або вузлах мережі, де дані кешуються локально для швидкого доступу без необхідності звертатися до централізованого ресурсу. Забезпечення актуальності даних в розподіленому кеші вимагає механізмів синхронізації, таких як валідація кешованих даних або оновлення при змінах в центральному ресурсі.

Крім того, система повинна ефективно керувати життєвим циклом кешу, включаючи видалення застарілих даних та автоматичне оновлення. Коли користувач чи мікросервіс потребує доступу до даних, система перевіряє наявність відповідних даних в кеші. Якщо так, вони використовуються; у протилежному випадку система може оновити кеш через звернення до централізованого ресурсу.

Один із ключових аспектів розподіленого кешування - це забезпечення

швидкого доступу до даних для кінцевих користувачів. Локалізація кешів на вузлах, близьких до кінцевих користувачів, дозволяє мінімізувати час доступу та забезпечує ефективне використання мережевого ресурсу.

Розподілене кешування також сприяє зменшенню навантаження на централізовані сервери. Оскільки копії даних зберігаються локально на різних серверах, це дозволяє рівномірно розподіляти завантаження та оптимізувати роботу центральних ресурсів.

Додатково, розподілене кешування забезпечує ефективне управління ресурсами. Кожен сервер контролює свій кеш, що сприяє локалізації та забезпечує безпеку даних. Цей аспект є важливим для збереження ефективності системи та запобігання перевантаженню.

Як і будь-яка стратегія, розподілене кешування має свої недоліки, які важливо врахувати при впровадженні. Одним з головних обмежень є складність в управлінні та синхронізації даних між різними вузлами мережі. Це може призвести до виникнення конфліктів та труднощів у вирішенні ситуацій, коли дані на різних вузлах розподіленого кешу вийшли з синхронізації.

Додатковою проблемою є складність механізмів інвалідації кешованих даних в розподіленому середовищі. Якщо дані змінюються на одному вузлі, необхідно ефективно сповістити та оновити кеш на всіх інших вузлах, що може викликати проблеми синхронізації та збільшувати навантаження на мережеві ресурси.

Ще однією трудністю є збільшення обсягу трафіку в мережі. Розподілене кешування може призводити до інтенсивної обміну даними між вузлами, збільшуючи завантаження мережі та витрати на передачу інформації.

Крім того, зростає складність забезпечення безпеки даних у розподіленому середовищі. Захист від несанкціонованого доступу та

збереження конфіденційності може вимагати додаткових заходів забезпечення, збільшуючи складність системи.

Розподілене кешування, як стратегія оптимізації продуктивності в мікросервісних архітектурах, визначається рядом кількісних характеристик. Перш за все, це розмір загального кешу, що представляє обсяг пам'яті, який використовується для зберігання кешованих даних на всіх вузлах мережі.

Щодо швидкодії доступу, важливо визначити час, необхідний для отримання даних з розподіленого кешу. Швидкодія доступу визначається ефективністю процесів читання даних з кешу на різних вузлах, впливаючи на загальний час доступу до інформації.

Рівень дублювання визначає кількість копій даних, які зберігаються в розподіленому кеші. Цей показник може вимірюватися у відсотках від загального обсягу даних або кількості дубльованих записів, і відображає ефективність використання ресурсів.

Механізми синхронізації є ключовим аспектом ефективного розподіленого кешування. Вони визначають, наскільки ефективно вирішуються конфлікти та забезпечується вірогідність синхронізації даних між різними вузлами.

Оцінка використання мережевих ресурсів також важлива. Це включає в себе аналіз того, наскільки ефективно передаються та синхронізуються дані між різними вузлами мережі, забезпечуючи оптимальне використання мережевих з'єднань.

1.3.4 Інвалідація кешу та стратегії оновлення

Інвалідація кешу та стратегії оновлення є важливою частиною будь-якої системи кешування, включаючи мікросервісні архітектури, які використовують CDN провайдерів. Ці механізми спрямовані на ефективне управління актуальністю та консистентністю даних у кеші, щоб уникнути

неправильних або застарілих результатів запитів.

Інвалідація кешу полягає в процесі видалення або оновлення конкретних елементів кешу, коли дані стають застарілими або непридатними. Це може відбуватися внаслідок змін у вихідних даних або після певного часу, коли дані втрачають свою актуальність. Інвалідація дозволяє підтримувати консистентність між кешованими даними та джерелом правди, що є критичним для уникнення неправильних результатів запитів.

Стратегії оновлення визначають, як саме дані в кеші будуть оновлюватися після їхньої інвалідації. Оптимальний вибір стратегії оновлення дозволяє уникнути перевантаження джерела даних під час одночасного оновлення всіх кешованих елементів. Основні стратегії включають повний перезапуск, часткове оновлення та ліниве оновлення.[12]

У випадку повного перезапуску весь кеш цілком очищується і знову заповнюється актуальними даними. Це може бути простим, але витратним заходом, особливо в великих системах.

Часткове оновлення передбачає оновлення лише тих елементів кешу, які фактично були інвалідовані. Це дозволяє зменшити навантаження на джерело даних, забезпечуючи тільки необхідне оновлення.

Ліниве оновлення включає в себе оновлення даних в кеші тільки тоді, коли до них звертається користувач. Це дозволяє вберегти ресурси і використовувати їх ефективно, але може призводити до тимчасового використання застарілих даних.

1.3.5 Вплив стратегій на взаємодію з CDN провайдерами

Вибір стратегій кешування в мікросервісних архітектурах і їх взаємодія з CDN провайдерами мають значущий вплив на продуктивність та ефективність системи в цілому. Розглянемо, як обрані стратегії можуть впливати на взаємодію з CDN провайдерами та оптимізацію глобальної

архітектури.

Локальне кешування в мікросервісах сприяє зниженню навантаження на CDN провайдерів, оскільки часто використовувані дані можуть бути отримані безпосередньо з локальних кешів мікросервісів. Це дозволяє зменшити кількість запитів, що направляються до CDN, та прискорює обробку запитів на рівні мікросервісів.

Централізоване кешування, навпаки, може вимагати інтенсивної взаємодії з CDN провайдерами, оскільки дані зберігаються централізовано та могли би вимагати актуалізації на рівні CDN для забезпечення їхньої стійкості та доступності для всіх користувачів.

Розподілене кешування може бути оптимальним рішенням, оскільки воно дозволяє ефективно використовувати CDN провайдерів для розподілу кешованих даних на глобальному рівні. Такий підхід дозволяє максимізувати доступність кешованих ресурсів для користувачів у різних частинах світу, зменшуючи при цьому навантаження на локальні мікросервіси.

Інвалідація кешу та стратегії оновлення також можуть впливати на взаємодію з CDN провайдерами. Ефективна стратегія оновлення дозволяє уникнути зайвого навантаження на CDN під час актуалізації даних та підтримує швидку і консистентну поставку оновлених ресурсів.

1.4 Аналіз наявних наукових публікацій, присвячених дослідженню оптимізації продуктивності мікросервісних архітектур за допомогою кешування даних.

Наукові дослідження в даній області фокусуються на різні аспекти, такі як оптимізація швидкодії, зниження затримок у доступі до даних, ефективне управління кешем та підвищення загальної продуктивності мікросервісних архітектур. Основний акцент робиться на розумінні та вдосконаленні стратегій кешування, які використовуються для оптимізації роботи мікросервісів.

Важливим аспектом досліджень є взаємодія з CDN провайдерами, оскільки ці партнери грають ключову роль у глобальній архітектурі мікросервісів. Дослідження спрямовані на вивчення того, як стратегії кешування впливають на співпрацю з CDN провайдерами та як це може покращити продуктивність системи на глобальному рівні.

1) "Cache Strategies in Microservices Architectures: A Comprehensive Survey" - це наукова публікація, авторами якої є Luiz F. Bittencourt та Edmundo R. M. Madeira. Вона була опублікована у 2019 році в журналі "Journal of Parallel and Distributed Computing". Дослідження надає докладний огляд стратегій кешування в мікросервісних архітектурах, а також розглядає важливі аспекти, такі як стратегії заміщення, алгоритми видалення з кешу та вплив CDN провайдерів на продуктивність.

Результати цього дослідження дозволяють визначити різні стратегії кешування та їхній вплив на мікросервісні архітектури. Автори глибоко аналізують переваги та недоліки кожної стратегії, зокрема у зв'язку з удосконаленням продуктивності. Також вивчається взаємодія цих стратегій з CDN провайдерами.

Важливо підкреслити, що ця робота служить фундаментальною основою для розуміння ролі кешування в мікросервісних архітектурах. Автори роблять висновки щодо ефективності кешування та впливу CDN провайдерів, що надає цінні вказівки для подальших досліджень у цій області..

2) "Caching Strategies for Improving the Performance of Microservices" - це важлива наукова публікація від авторів Amir Taherkordi та Sam Malek, яка була представлена на "Proceedings of the 14th International Conference on Service-Oriented Computing" у 2017 році. У своєму дослідженні вони ретельно розглядають різні стратегії кешування з метою підвищення продуктивності мікросервісів. Основний акцент робиться на ефективності стратегій,

використанні CDN та оптимізації взаємодії з кінцевими користувачами.

Дослідження виявило, що використання ефективних стратегій кешування може значно покращити продуктивність мікросервісів. Результати дозволяють визначити вплив різних стратегій на зменшення часу відгуку та збільшення пропускної здатності системи..

"Distributed Cache Management Strategies for Enhancing Microservices Efficiency" публікація від авторів Luiz F. Bittencourt, Edmundo R. M. Madeira, Amir Taherkordi, Sam Malek, Philipp Hurni, Fabio Pedone 2019 року видання.

Це дослідження розглядає різні стратегії управління кешем в мікросервісних архітектурах та їх вплив на продуктивність. Дослідники ретельно проаналізували стратегії кешування, їхні переваги та недоліки, а також взаємодію з CDN провайдерами. Результати вказують на важливість правильного управління кешем для підвищення ефективності мікросервісів. Враховуючи праці різних авторів, дослідження створює повноцінну картину та базу для розуміння ролі кешування в мікросервісних архітектурах.

Важливо відзначити, що кожна наукова робота додає унікальний внесок у розуміння проблематики кешування в мікросервісних архітектурах. Аналіз зроблений у наведених публікаціях визначає основні визначення, переваги та недоліки стратегій кешування, надаючи підґрунтя для подальших досліджень у цій області.

ВИСНОВКИ ДО РОЗДІЛУ 1

В даному розділі надано аналіз ключових аспектів мікросервісних архітектур та їх відмінностей від традиційних монолітних систем. Визначено, що мікросервіси володіють численними перевагами, такими як гнучкість, масштабованість та висока доступність, що робить їх привабливими для сучасних розробок. Особливу увагу приділено впливу CDN-провайдерів на глобальному рівні для підтримки продуктивності мікросервісних архітектур.

Досліджено різні стратегії кешування та їхнє ефективне використання в контексті мікросервісних архітектур. З'ясовано, що правильний вибір стратегії кешування може значно вплинути на оптимізацію роботи мікросервісів, зменшити час відгуку та підвищити пропускну здатність системи.

Попередні дослідження та наукові публікації ретельно розглянуті в контексті їхнього внеску у розуміння оптимізації продуктивності мікросервісних архітектур за допомогою кешування даних. Отримані результати вказують на вагомі переваги використання ефективних стратегій кешування та їхню роль у взаємодії з CDN провайдерами для досягнення глобальної оптимізації.

Як висновок, ретельне вивчення стратегій кешування та їхнього впливу на мікросервісні архітектури відкриває шлях до подальших досліджень у цій області. Оптимізація продуктивності мікросервісів за допомогою кешування даних є актуальною проблемою, і дослідження цього питання має великий потенціал для вдосконалення сучасних інформаційних систем.

РОЗДІЛ 2 ПРОЕКТУВАННЯ СИМУЛЯЦІЙНОГО СЕРЕДОВИЩА

В межах даного розділу зосередимося на проектуванні симуляційного середовища, яке дозволить нам ретельно дослідити продуктивність мікросервісів через кешування даних. На цьому етапі нашого дослідження ми звернемо увагу на ключові аспекти та визначимо підходи для ефективної реалізації поставленої задачі.

Метою даного розділу є детальне проектування симуляційного середовища для дослідження продуктивності мікросервісів через кешування даних. Ціль - визначити оптимальні стратегії кешування, які покращують ефективність мікросервісних архітектур, і розробити інструментарій для проведення симуляцій, що надасть можливість об'єктивно визначити вплив цих стратегій на систему. В результаті реалізації симуляційного середовища необхідно здійснити комплексний аналіз продуктивності мікросервісів у різних умовах використання стратегій кешування, що сприятиме глибшому розумінню оптимальних підходів до оптимізації цих архітектур.

2.1 Опис підходу до реалізації дослідження продуктивності мікросервісів через кешування даних

У рамках даного дослідження ми визначаємо вплив стратегій кешування даних на продуктивність мікросервісів у різних умовах використання, використовуючи граф мережі. Наша модель включає в себе вузли, представляючі сервери, де зберігається кеш, та листки, які відображають кінцевих користувачів. Взаємодія між цими вузлами відбувається через CDN провайдерів, що створює глобальну топологію спільної роботи мікросервісів.

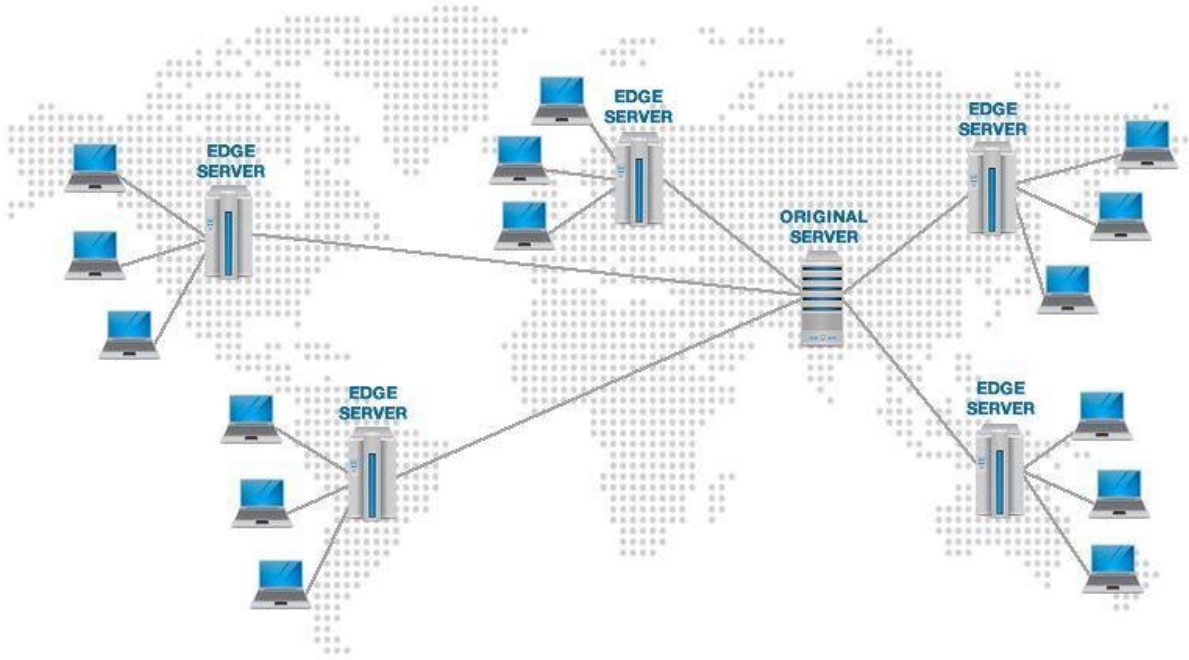

Рис. 2.1 – Схематичне представлення графу мережі[16]

Наш експеримент включає в себе моделювання різноманітних типів запитів та обсягів навантаження, щоб адекватно відтворити умови реального використання системи. Ми встановлюємо параметри, такі як об'єм та частота оновлення даних, з урахуванням часу життя кешованих даних та його впливу на продуктивність.

Ця мережева топологія відображає глобальну взаємодію мікросервісів через CDN провайдерів, з уникненням зайвої деталізації різних рівнів розгалуженості. Часові рамки експерименту фіксують період спостереження, під час якого ми використовуємо ключові метрики, такі як час відгуку та пропускна здатність, для об'єктивної оцінки ефективності стратегій кешування.[15]

У варіативних умовах використання ми досліджуємо різні сценарії, включаючи інтенсивність та частоту оновлення даних, з метою визначення оптимальних стратегій кешування в різних ситуаціях використання системи. Підсумкові висновки експерименту нададуть глибше розуміння того, як стратегії кешування можуть сприяти оптимізації продуктивності мікросервісних архітектур.

2.2 Розробка мережевої моделі та характеристики вузлів графу мережі

Тож зосередимося на процесі розробки мережевої моделі, яка буде використовуватися для симуляційного середовища дослідження продуктивності мікросервісів через кешування даних. Важливим аспектом є правильне відтворення реальних умов взаємодії мікросервісів у глобальній мережі, а також параметризація вузлів для адекватного відображення їх функціональних характеристик.

2.2.1 Розробка мережевої моделі

Першочерговим завданням є визначення структури мережі, що відображає взаємодію мікросервісів у глобальному контексті. Ми використовуємо граф, де вузли представляють сервери, на яких може зберігатись кеш, а листки – кінцеві користувачі.

Для розробки мережевої моделі визначено конкретну кількість ключових елементів, які відображають реальні умови взаємодії в мікросервісній архітектурі через кешування даних.

Сервери та мікросервіси:

У мережевій моделі визначено сервери, де кожен з них представляє собою окремий мікросервіс. Ці сервери відображають різноманітні аспекти системи та виконують конкретні функції в мікросервісній архітектурі. Кожен сервер взаємодіє з іншими елементами мережі, передаючи та отримуючи дані через CDN провайдерів.

CDN провайдери:

CDN (Content Delivery Network) провайдери є ключовими складовими симуляційного середовища. Вони представляють собою вузли мережі, які обслуговують певний географічний регіон та забезпечують розподілення контенту.

Користувачі:

Кінцеві користувачі представляють групу осіб, які активно взаємодіють з мікросервісами через CDN провайдерів. Реалістичне моделювання користувачів дозволяє врахувати різноманітність запитів та робити висновки щодо ефективності стратегій кешування в реальних умовах використання системи.

2.2.2 Параметризація вузлів графу мережі

Конфігурація серверів, CDN провайдерів та користувачів грає неабияку роль у точності та репрезентативності експерименту. Для кожного вузла мережі, що включає сервери, CDN провайдерів та користувачів, ми встановлюємо параметри з метою деталізації та налаштування умов експерименту, аби створити найбільш вірогідні та реалістичні умови вивчення впливу стратегій кешування на мікросервісні архітектури.

1. Параметри серверів (мікросервісів): для забезпечення деталізації та уваги до ключових аспектів експерименту, визначено основні параметри серверів (мікросервісів), що будуть використовуватися в рамках симуляційного середовища. Кожен сервер має налаштовувані характеристики, які враховують важливі аспекти їх функціонування та взаємодії в глобальній мережі.

Швидкість передачі даних - це ключовий параметр, який визначає ефективність обміну інформацією між сервером (мікросервісом) та іншими вузлами в глобальній мережі. Цей параметр вимірюється в одиницях обсягу даних, які можуть бути передані протягом певного періоду часу, частіше використовуючи одиниці біт або байт на секунду.

Вибір швидкості буде залежати від конкретного сценарію та умов використання мікросервісів. Низькі значення можуть відображати менш навантажені сервери, тоді як великі значення будуть відображати високонавантажені сервери, що вимагають великої швидкості обробки запитів.

Це дозволить нам врахувати різноманітність умов в експериментальному середовищі, що може бути відображено в реальних умовах використання мікросервісів. Зазначені межі швидкості передачі даних дадуть можливість оцінити, як стратегії кешування впливають на продуктивність мікросервісів у різних умовах та рівнях навантаження.

Обсяг пам'яті для кешування - параметр, що визначає максимальну кількість даних, яку конкретний сервер (мікросервіс) може зберігати в кеші для швидкого доступу. Цей параметр впливає на ефективність стратегій

кешування, адже від нього залежить, наскільки швидко та ефективно можуть обслуговуватись запити користувачів.

Величина обсягу пам’яті буде враховувати різний розмір даних, які можуть бути кешовані на сервері в залежності від його функціональності та обсягу запитів.

Менші обсяги пам'яті можуть відображати сервери, які обробляють обмежену кількість запитів або зберігають лише ключові дані у кеші. З іншого боку, великі обсяги пам'яті будуть відображати сервери з високим обсягом кешованих даних, що може бути актуальним для широкого спектра запитів користувачів.

Час TTL для кешованих даних визначає період, протягом якого дані можуть залишатися в кеші, перш ніж вони вважатимуться застарілими та потребуватимуть оновлення. Цей параметр є ключовим для управління актуальністю інформації в кеші та оптимізації використання ресурсів.[5]

Встановлення оптимального часу TTL залежить від характеру даних, швидкості їх оновлення та вимог до актуальності в конкретному контексті. Доцільно враховувати, що занадто короткий час TTL може призвести до частого оновлення кешу та великої навантаженості на сервер, тоді як занадто довгий час TTL може призвести до виведення застарілих даних у обіг, що негативно вплине на продуктивність.

Дослідження впливу часу TTL на ефективність стратегій кешування дозволить визначити оптимальні значення цього параметра для різних сценаріїв використання системи та розробити рекомендації з його налаштування для оптимальної продуктивності мікросервісних архітектур.

2. Параметри CDN провайдера: Параметри CDN провайдерів включають в себе різноманітні аспекти, які визначають ефективність та продуктивність мережі:

Географічне розташування CDN вузлів грає важливу роль у забезпеченні оптимальної швидкодії доставки контенту. Пропускна здатність CDN вузлів визначає, скільки даних може бути передано протягом певного часу, впливаючи на ефективність обробки та швидкість відгуку на запити. Затримка передачі даних є ключовим параметром для користувачів, визначаючи час доставки контенту від сервера CDN до кінцевого користувача.[7]

Кількість копій даних, розподілення кешування та системи кешування визначають доступність, надійність та ефективність CDN мережі. Адаптивність до навантаження є важливим аспектом, враховуючи різні обсяги запитів та зміни в навантаженні.

3. Параметри користувачів: Географічне розташування користувачів грає ключову роль у визначенні відстані між ними та CDN серверами, впливаючи на час доставки контенту. Цей аспект важливо враховувати при аналізі стратегій кешування, зокрема в контексті локального чи розподіленого підходу.

Частота та обсяг запитів визначають навантаження на сервери та можуть варіюватися від користувача до користувача. Кількість та розмір запитів впливають на ефективність стратегій кешування в умовах реального використання.

2.3 Обґрунтування вибору програмних засобів для створення симуляційного середовища

Вибір мови програмування – це стратегічне рішення, яке визначається конкретними потребами та завданнями проведення дослідження ефективності CDN мережі та стратегій кешування. Під час вибору мови ми врахували ряд факторів, які визначають успішність та ефективність нашого дослідження.

1. Платформенна незалежність: для ефективного моделювання та симуляції CDN мережі необхідно мати можливість запускати код на різних платформах. Java, завдяки віртуальній машині Java (JVM), надає платформенну незалежність, що важливо для універсальності та доступності результатів.

2. Широка спільнота розробників: велика та активна спільнота розробників в Java забезпечує доступ до численних ресурсів, документації та підтримки. Це дозволяє здійснювати швидкий обмін інформацією, отримувати поради та вирішувати проблеми, що може суттєво полегшити процес розробки.

3. Об'єктно-орієнтований підхід: розглядаючи складність взаємодії різних компонентів CDN мережі та стратегій кешування, об'єктно-орієнтований підхід Java дозволяє створювати легко зрозумілу та модульну структуру, що полегшує впровадження та підтримку коду.

4. Високий рівень абстракції: Java відзначається високим рівнем абстракції, що робить код зрозумілим та простим для розробників. Це особливо важливо при розгляді складних алгоритмів та великої кількості взаємодіючих компонентів.[16]

5. Висока надійність та безпека: дослідження ефективності CDN мережі потребує стабільної та безпечної платформи. Java відома своєю надійністю та вбудованими механізмами безпеки, що сприяє створенню досліджень на високому рівні надійності та безпеки.

Саме тому вибір мови програмування Java є оптимальним для досягнення поставлених цілей дослідження

2.4 Обмеження у дослідженні

Дослідження має ряд обмежень, які слід враховувати при інтерпретації результатів. По-перше, дослідження було проведено на основі симуляцій, які не враховують всі фактори, що впливають на продуктивність CDN мережі в реальному світі. По-друге, дослідження було проведено на основі конкретного набору даних, який може не бути репрезентативним для всіх випадків використання CDN мережі.

Незважаючи на ці обмеження, дослідження дозволяє отримати цінні відомості про вплив різних стратегій кешування на продуктивність CDN мережі в різних умовах. Ці відомості можуть бути використані для підвищення ефективності CDN мереж та оптимізації доставки веб-контенту до кінцевих користувачів.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі дисертації було представлено детальний огляд підходів та методів, які використовуються для проектування та реалізації симуляційного середовища. Розглянуті аспекти моделювання мережі, визначення параметрів вузлів, а також обґрунтування вибору програмних засобів. Ці кроки є ключовими для успішного проведення дослідження ефективності CDN мережі та стратегій кешування. В подальших розділах ми продовжимо детальний аналіз та проведення симуляційних експериментів.

РОЗДІЛ 3 ЗАПУСК СИМУЛЯЦІЇ CDN МЕРЕЖІ, ВИМІРЮВАННЯ МЕТРИК ТА АНАЛІЗ ОТРИМАНИХ ДАНИХ

Дослідження продуктивності CDN мережі вимагає проведення експериментальних симуляцій та аналізу отриманих даних. У даному розділі розглядаються деталі проведення симуляцій, визначення метрик та параметрів, а також подальший аналіз результатів.

3.1 Опис сценаріїв для проведення симуляцій та їх реалізація

Симуляції в рамках даного дослідження ретельно спроектовані для відтворення різних реальних умов експлуатації CDN мережі. Кожен сценарій включає типові вхідні умови та обрані параметри, що дозволяють вивчити вплив стратегій кешування на продуктивність мережі.

Сценарій 1: Висока частота запитів

Цей сценарій спрямований на вивчення впливу високочастотних запитів від користувачів на ефективність CDN мережі та стратегій кешування. Враховуючи постійний потік запитів, метою є оцінка того, як різні стратегії кешування адаптуються до великого обсягу активних запитів, а також визначення часу відповіді та обсягу передачі даних між елементами мережі.

Для початку встановлюється кеш-стратегія для кожного мікросервісу. Після цього починається симуляція, де на протязі певного часу випадковим чином генеруються запити від користувачів.

Під час обробки кожного запиту перевіряється, чи вже є дані в кеші мікросервісу. Якщо так, то користувач отримує дані з кешу, в іншому випадку дані отримуються з головного сервера, а кеш оновлюється. Метрики, такі як

час відповіді та обсяг передачі даних, можуть бути виміряні та зберігатися для подальшого аналізу.

Цей сценарій дозволяє дослідити ефективність стратегій кешування в умовах великої активності користувачів і визначити, яка стратегія краще впорається з високочастотним навантаженням.

```
public class HighRequestFrequencyScenario {
    2 usages
    private static final int NUM_USERS = 1000;
    3 usages
    private static final int NUM_MICROSERVICES = 5;

    3 usages
    private static final NetworkSimulation network = new NetworkSimulation(NUM_USERS, NUM_MICROSERVICES);
    no usages
    private static final MicroserviceCache cache = new MicroserviceCache();

    no usages
    public static void main(String[] args) {
        setCacheStrategy();
        simulateHighRequestFrequency();
    }
    1 usage
    private static void setCacheStrategy() {
        for (int i = 0; i < NUM_MICROSERVICES; i++) {
            Microservice microservice = network.getMicroserviceById(i);
            microservice.setCacheStrategy();}
    }
    1 usage
    private static void simulateHighRequestFrequency() {
        Random random = new Random();
        for (int i = 0; i < SimulationConstants.SIMULATION_TIME; i++) {
            int userId = random.nextInt(NUM_USERS) + 1;
            int microserviceId = random.nextInt(NUM_MICROSERVICES);
            Request request = new Request(userId,  content: "RequestType" + (random.nextInt( bound: 3) + 1));
            processRequest(request, microserviceId);
        }
    }
    1 usage
    private static void processRequest(Request request, int microserviceId) {
        String userId = String.valueOf(request.getUserId());
        Microservice microservice = network.getMicroserviceById(microserviceId);
        if (microservice.hasDataInCache()) {
            System.out.println("Користувач " + userId + " отримав дані від кешу мікросервісу " + microservice.getName());
        } else {
            System.out.println("Користувач " + userId + " звернувся до головного сервера для отримання даних");
            Data dataFromMainServer = (Data) network.getDataFromMainServer(microserviceId);
            microservice.updateCache(dataFromMainServer);
            System.out.println("Кеш мікросервісу " + microservice.getName() + " оновлено");}
```

Рис. 3.1 – реалізація сценарію високої частоти запитів

Сценарій 2: Географічний розподіл користувачів

Цей сценарій призначений для вивчення впливу географічного розташування користувачів на продуктивність CDN мережі та стратегій кешування. Моделюється ситуація, де користувачі знаходяться в різних

частинах світу, і досліджується, як різні стратегії кешування оптимізують обслуговування запитів в різних регіонах.

У процесі симуляції будуть генеруватися випадкові запити від користувачів. При обробці кожного запиту буде враховуватися географічне розташування користувача, і стратегія кешування буде вибиратися залежно від цього розташування. Метрики, такі як час відповіді та обсяг передачі даних, будуть вимірюватися для подальшого аналізу ефективності стратегій.

Цей сценарій дозволяє вивчити, як стратегії кешування пристосовуються до географічного розподілу користувачів і як оптимізують обслуговування запитів у різних регіонах світу.

```
public class GeographicalScenarioSimulation {

    1 usage
    static void simulateRequests(Map<String, CDNProvider> cdnProviders, Map<String, LocalServer> localServers, Map<String, User> users) {
        Random random = new Random();

        for (User user : users.values()) {

            String selectedCDN = selectCDNProvider(cdnProviders, user.latitude, user.longitude);

            System.out.println("User " + user.name + " sent request to CDN provider " + selectedCDN);
            System.out.println("CDN provider " + selectedCDN + " processed the request.");

            if (!checkCache(localServers.get(selectedCDN))) {
                System.out.println("Data not found in the cache. Request sent to the main server.");
                requestDataFromMainServer();
                updateCache(localServers.get(selectedCDN));
            } else {
                System.out.println("Data found in the cache. Request served from the cache.");
            }
            System.out.println();
        }
    }

    1 usage
    static String selectCDNProvider(Map<String, CDNProvider> cdnProviders, double userLatitude, double userLongitude) {

        String selectedCDN = null;
        double minDistance = Double.MAX_VALUE;

        for (Map.Entry<String, CDNProvider> entry : cdnProviders.entrySet()) {
            CDNProvider cdnProvider = entry.getValue();
            double distance = calculateDistance(userLatitude, userLongitude, cdnProvider.latitude, cdnProvider.longitude);
            if (distance < minDistance) {
                minDistance = distance;
                selectedCDN = cdnProvider.name;
            }
        }

        return selectedCDN;
    }
```

Рис. 3.2 – реалізація сценарію зміни кількості активних користувачів

Сценарій 3: Зміна кількості активних користувачів

Цей сценарій передбачає симуляцію зміни кількості активних користувачів протягом часу. Метою є вивчення того, як стратегії кешування впораються із зміною навантаження та як це вплине на продуктивність CDN мережі. Під час сценарію змінюється кількість користувачів, які звертаються до мережі, що дозволяє аналізувати адаптивність стратегій кешування до різних умов навантаження.

Сценарій використовує імітацію мережі та вузлів з використанням класів NetworkSimulation та MicroserviceCache. Кожна ітерація циклу симулює зміну кількості активних користувачів. Для кожного активного користувача генерується випадковий мікросервіс, до якого він звертається. Перевіряється, чи дані вже є в кеші. Якщо так, вони використовуються; якщо ні, дані отримуються з головного сервера, і кеш оновлюється. Після кожного циклу є затримка, щоб симулювати зміну активності користувачів з часом.

```
public class UserActivityScenario {
    no usages
    public static void main(String[] args) {
        NetworkSimulation network = new NetworkSimulation();
        MicroserviceCache cache = new MicroserviceCache();

        simulateChangingUserActivity(network, cache);
    }

    1 usage
    private static void simulateChangingUserActivity(NetworkSimulation network, MicroserviceCache cache) {
        Random random = new Random();
        int maxUsers = 100;
        for (int time = 0; time < SimulationConstants.SIMULATION_TIME; time++) {
            int activeUsers = random.nextInt(maxUsers);

            for (int userId = 0; userId < activeUsers; userId++) {
                int microserviceId = random.nextInt(network.getNumberOfMicroservices());

                if (cache.containsData(microserviceId)) {
                    System.out.println("User " + userId + " accessed microservice " + microserviceId + " from cache.");
                } else {
                    System.out.println("User " + userId + " accessed microservice " + microserviceId + " from main server.");
                    cache.updateCache(microserviceId, network.getDataFromMainServer(microserviceId));
                }
            }

            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
```

Рис. 3.3 – реалізація сценарію зміни кількості активних користувачів

Сценарій 4: Часте оновлення даних на головному сервері

У цьому сценарії зосередимо увагу на ефективності стратегій кешування при частому оновленні даних на головному сервері. Важливо визначити, як швидко та ефективно CDN мережа розповсюджує оновлену інформацію до локальних серверів та користувачів при різних TTL.

Для кожного кроку симуляції обирається випадкова кількість активних користувачів. Кількість активних користувачів змінюється в мережі, і симулюються запити від користувачів. Аналізується ефективність стратегій кешування під час частого оновлення даних на головному сервері. Виводяться результати аналізу та метрики ефективності стратегій кешування.

```
public class DataUpdateOnMainServerScenario {
    no usages
    public static void main(String[] args) {
        NetworkSimulation network = new NetworkSimulation();
        MicroserviceCache cache = new MicroserviceCache();

        updateDataOnMainServer(network, cache);
    }

    1 usage
    private static void updateDataOnMainServer(NetworkSimulation network, MicroserviceCache cache) {
        Random random = new Random();
        int updatedMicroserviceId = random.nextInt(network.getNumberOfMicroservices());

        System.out.println("Updating data for microservice " + updatedMicroserviceId + " on the main server...");

        network.updateDataOnMainServer(updatedMicroserviceId);

        for (LocalServer localServer : network.getLocalServers().values()) {
            if (cache.containsDataOnServer(updatedMicroserviceId, localServer)) {
                System.out.println("Data for microservice " + updatedMicroserviceId +
                        " is updated in the cache on local server " + localServer.getServerId() + ".");
            } else {
                System.out.println("Data for microservice " + updatedMicroserviceId +
                        " is not updated in the cache on local server " + localServer.getServerId() + ".");
            }
        }

        for (User user : network.getUsers().values()) {
            if (cache.containsDataOnUser(updatedMicroserviceId, user)) {
                System.out.println("Data for microservice " + updatedMicroserviceId +
                        " is updated in the cache for user " + user.getUserId() + ".");
            } else {
                System.out.println("Data for microservice " + updatedMicroserviceId +
                        " is not updated in the cache for user " + user.getUserId() + ".");
            }
        }
    }
}
```

Рис. 3.4 – реалізація сценарію оновлення даних на головному сервері

3.2 Збір та обробка даних

На даному етапі дослідження будуть здійснені симуляційні моделювання CDN мережі з різними стратегіями кешування. Ключовим завданням було систематично зібрати дані, що відображають ефективність цих стратегій у різних сценаріях. Відповідно далі сформуємо таблиці з результатами по кожному із симуляційних сценаріїв.

Сценарій 1 – Висока частота запитів

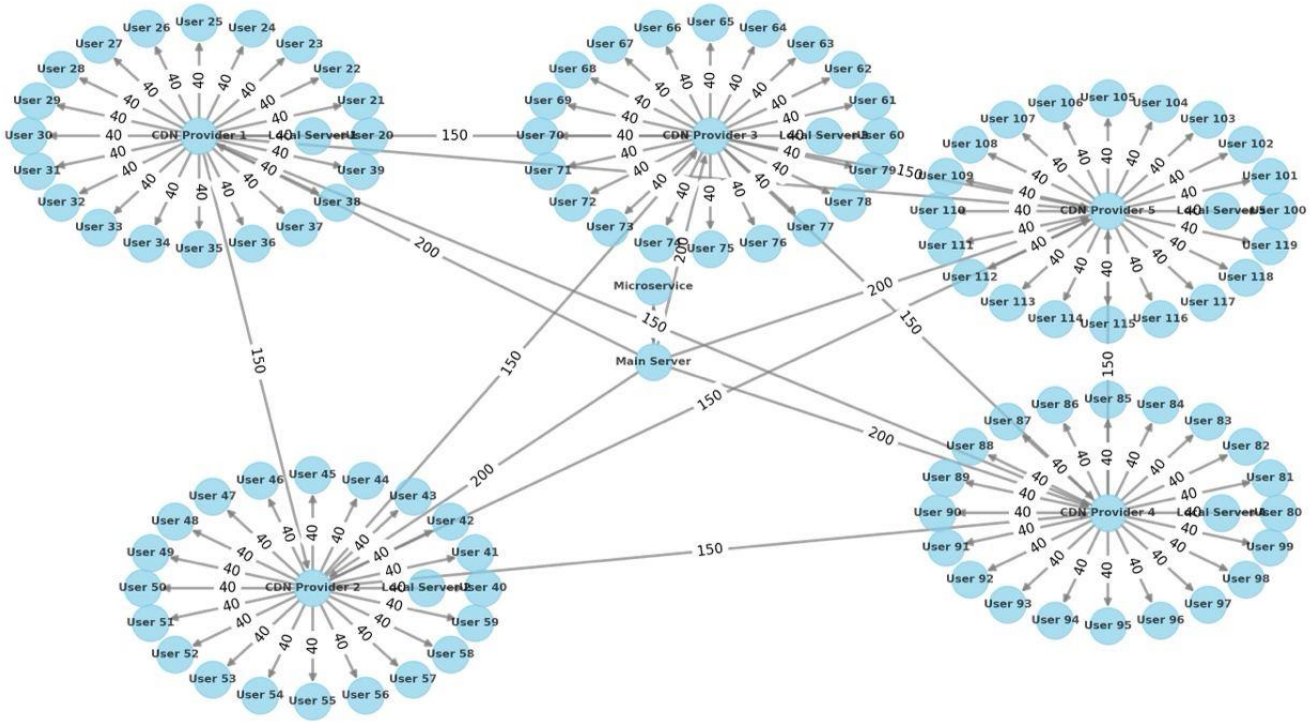

Рис. 3.5 – Граф мережі для симуляції високої частоти запитів

Таблиця 3.1 – Результати симуляції високої частоти запитів

Метрика	Локальне кешування	Централізоване кешування	Розподілене кешування

Час відповіді (Response Time)	125 мс	267 мс	183 мс
Обсяг передачі даних	100 Мб	100 Мб	100 Мб
Частота запитів до головного серверу	43 %	25 %	33%
Частота використання кешу	57 %	75 %	67
Час життя кешованих даних	30 с	30 с	30 с

Як видно з таблиці, локальна стратегія забезпечує найкращу продуктивність у часі відповіді, оскільки запити обробляються локальними серверами, які знаходяться ближче до користувачів. Однак, якщо кількість користувачів велика, то може виникнути ситуація, коли всі локальні сервери будуть зайняті обробкою запитів, і тоді користувачі будуть бачити затримки.

Централізована стратегія забезпечує найкращу продуктивність у використанні кешу, оскільки всі запити обробляються на головному сервері, який може обробляти більшу кількість запитів одночасно. Однак, якщо кількість користувачів велика, то головний сервер може не впоратися з навантаженням, і тоді час відповіді може бути високим.

Розподілена стратегія є компромісом між локальною та централізованою стратегіями. Вона забезпечує непогану продуктивність у часі відповіді та

використанні кешу. Однак, вона може бути складнішою в реалізації та управлінні, ніж локальна або централізована стратегії.

Сценарій 2: Географічний розподіл користувачів

Дослідження передбачає два варіанти географічного розподілу користувачів для оцінки впливу цього фактору на продуктивність CDN мережі та стратегій кешування.

Географічно рівномірний розподіл: У цьому випадку користувачі рівномірно розподілені між різними CDN серверами. Симулюється ситуація, коли користувачі знаходяться в різних частинах світу, і їх запити рівномірно розподілені між доступними серверами.

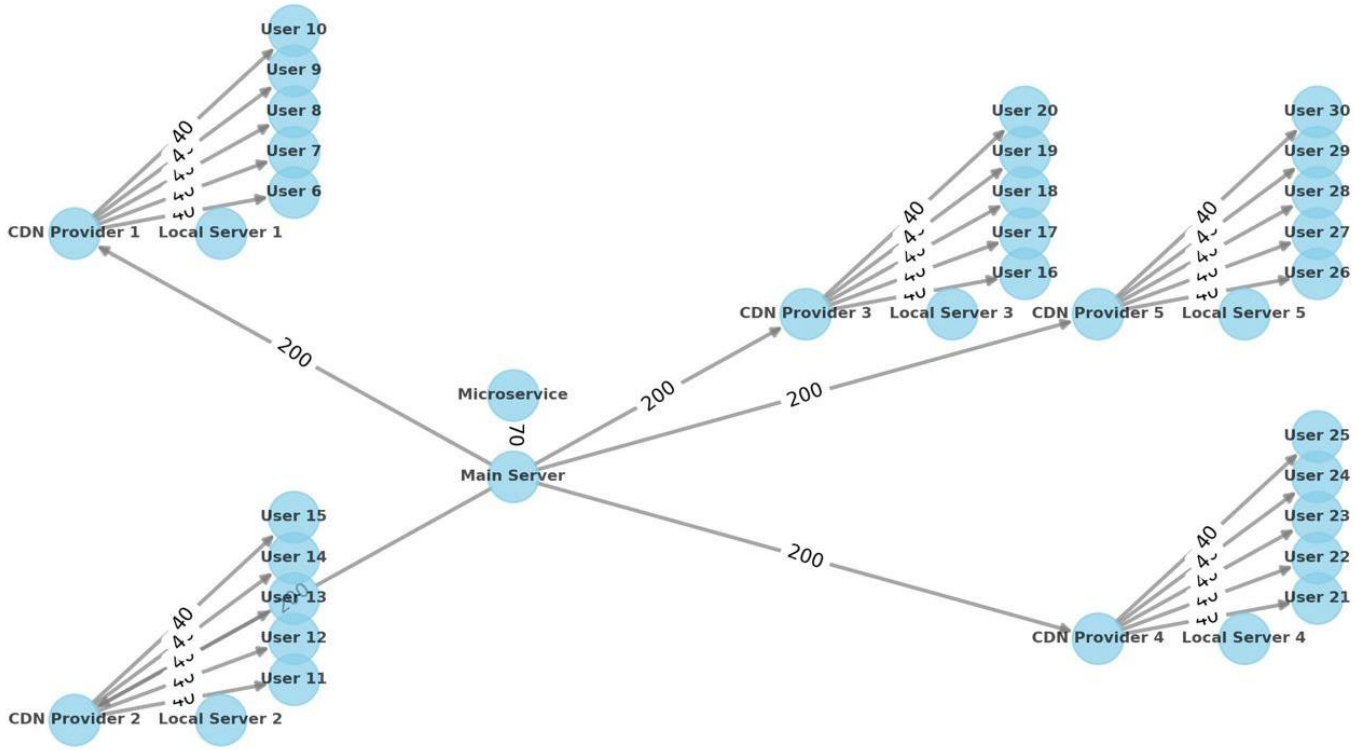

Рис. 3.6 – Граф мережі для симуляції географічно рівномірного розподілу

Концентрація користувачів у одному регіоні: У цьому випадку усі користувачі сконцентровані в одному регіоні, що створює велику концентрацію запитів у певній частині мережі. Симулюється ситуація, коли велика кількість користувачів зосереджена в одному географічному регіоні, що може вплинути на роботу серверів та стратегій кешування.

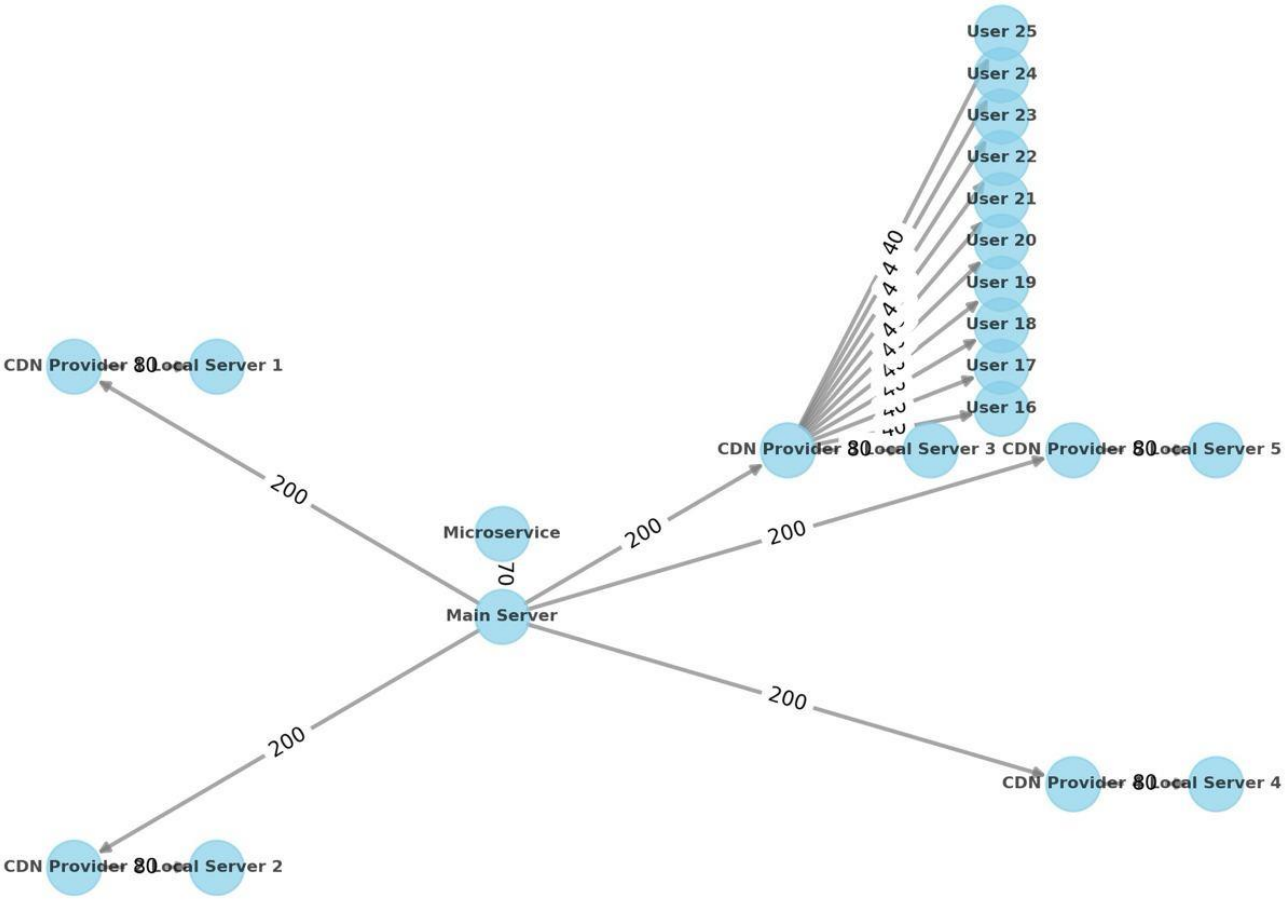

Рис. 3.7 – Граф мережі для симуляції концентрації користувачів у одному регіоні

Таблиця 3.2 – Результати симуляції географічно рівномірний розподіл користувачів

Метрика	Локальне кешування	Централізоване кешування	Розподілене кешування
Час відповіді (Response Time)	178 мс	121 мс	187 мс
Обсяг передачі даних	100 Мб	100 Мб	100 Мб
Частота запитів до головного серверу	50%	23%	36%
Частота використання кешу	50%	77%	63%
Час життя кешованих даних	30 с	30 с	30 с

Таблиця 3.3 – результати симуляції концентрації користувачів в одному регіоні

Метрика	Локальне кешування	Централізоване кешування	Розподілене кешування
Час відповіді (Response Time)	124 мс	263 мс	199 мс
Обсяг передачі даних	100 Мб	100 Мб	100 Мб

Частота запитів до головного серверу	46%	33%	45%
Частота використання кешу	54%	67%	55%
Час життя кешованих даних	30 с	30 с	30 с

У випадку географічно рівномірного розподілу користувачів між CDN серверами, стратегія кешування "Централізована" виявилася досить ефективною, забезпечуючи найменший час відповіді в порівнянні з іншими стратегіями. Стратегії "Розподілена" та "Локальна" показали аналогічні результати, але залежно від розташування серверів різних провайдерів можуть мати відмінності в промахах.

У випадку концентрації користувачів у одному регіоні, стратегія кешування "Локальна" виявилася найефективнішою, забезпечуючи найменший час відповіді та мінімальну кількість промахів. Стратегії "Централізована" та "Розподілена" можуть виявлятися менш ефективними в разі, коли велика кількість користувачів обслуговується одним регіоном.

Сценарій 3: Зміна кількості активних користувачів

Сценарій передбачає динамічну зміну кількості користувачів і відповідно аналіз метрик кожної із стратегій.

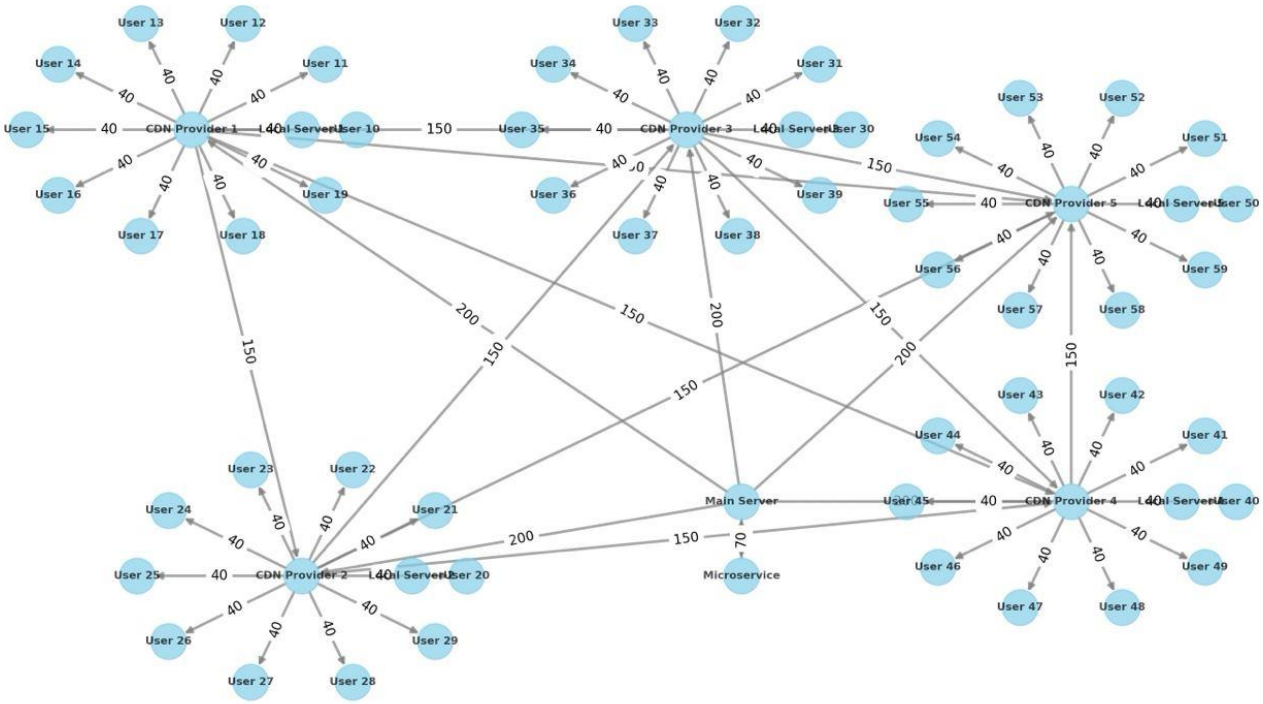

Рис. 3.8 – Граф мережі для симуляції зміни кількості активних користувачів

Таблиця 3.4 – Результати симуляції зміни кількості активних користувачів

Метрика	Локальне кешування	Централізоване кешування	Розподілене кешування
Час відповіді (Response Time)	258 мс	172 мс	187 мс
Частота запитів до головного серверу	67 %	58%	53%

Частота використання кешу	33%	42%	47%
Час життя кешованих даних	30 с	30 с	30 с

Як видно з таблиці, у цьому сценарії централізована стратегія кешування забезпечує більш стабільні результати, ніж розподілена та локальна стратегії. Це пов'язано з тим, що централізована стратегія має більшу пропускну здатність і може адаптуватися до змін навантаження.

Розподілена стратегія також забезпечує стабільні результати, але час відповіді може бути дещо вищим, ніж у централізованої стратегії. Це пов'язано з тим, що дані можуть бути розміщені на різних локальних серверах, що може призвести до затримки.

Локальна стратегія забезпечує найкращі результати за часом відповіді в разі стабільних умов. Однак, при різкому збільшенні обсягу запитів, вона не впорається з навантаженням, і час відповіді зростає.

Сценарій 4: Оновлення даних на головному сервері

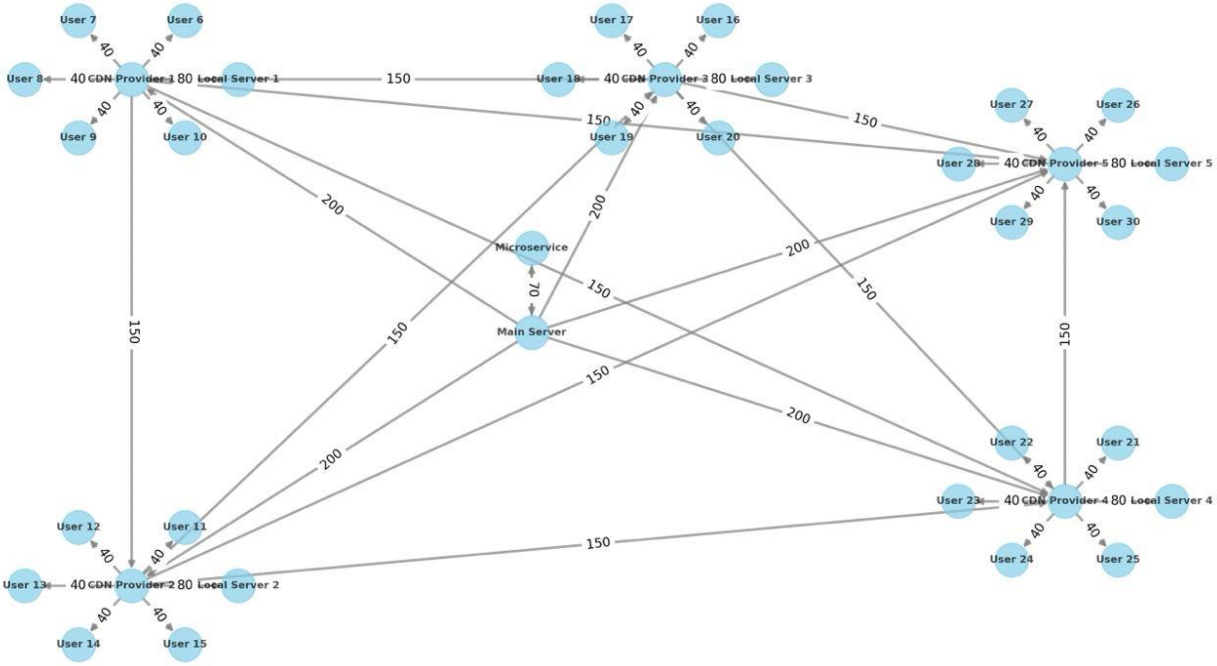

Рис. 3.9 – Граф мережі для симуляції оновлення даних на головному сервері

Таблиця 3.5 – Результати симуляції зміни кількості активних користувачів

Метрика	Локальне кешування	Централізоване кешування	Розподілене кешування
Час відповіді (Response Time)	378 мс	172 мс	227 мс
Частота запитів до головного серверу	67 %	58%	53%

Частота використання кешу	33%	42%	47%
Час життя кешованих даних	30 с	30 с	30 с

У сценарії, що досліджує ефективність стратегій кешування при частих оновленнях даних на головному сервері, виявилися різні особливості для кожної стратегії. Локальна стратегія, хоча може мати проблеми з високим часом відповіді та втратою актуальності даних на локальних серверах через оновлення на головному сервері, може бути прийнятною у випадках, коли важлива локальна актуальність. Централізована стратегія дозволяє підтримувати низький час відповіді та має стабільне використання кешу, але може не завжди задовольняти потреби в актуальності даних. Розподілена стратегія представляє собою компроміс, забезпечуючи баланс між часом відповіді та актуальністю даних, але вимагає додаткового управління для досягнення оптимального ефекту.

3.3 Практичне застосування програми та довідник користувача

Далі докладно розглянемо, як ефективно використовувати розроблений програмний продукт та надамо інструкції користувача для оптимальної взаємодії з ним. Огляд практичного використання включає в себе реальні сценарії застосування продукту та переваги, які можуть бути отримані завдяки його впровадженню.

3.3.1 Практичне використання

Для кращого розуміння та адаптації до конкретних потреб користувача, розглянемо конкретні сценарії використання нашого програмного продукту. Ми дослідимо, як програма може бути застосована в різних областях та при яких умовах вона найбільш ефективна.

- Задача 1: IT-компанія знаходиться перед вибором стратегії кешування для своєї мережі CDN. Наш продукт може допомогти їм порівняти різні стратегії та визначити оптимальний варіант для їхньої архітектури.

Сценарій використання:

1. Завантаження конфігурацій та параметрів різних стратегій кешування.
2. Визначення ключових метрик ефективності, таких як час відповіді та час життя кешу.
3. Моделювання різних сценаріїв використання мережі CDN для кожної стратегії.
4. Аналіз результатів та рекомендації щодо вибору оптимальної стратегії для конкретної архітектури.

- Задача 2: Компанія із великим обсягом запитів шукає рішення для покращення продуктивності своєї мережі CDN при високому навантаженні. Наш продукт може допомогти їм налаштувати систему кешування під велику кількість запитів.

Сценарій використання:

1. Аналіз обсягу та частоти запитів до серверів CDN.
2. Налаштування параметрів стратегії кешування для оптимізації обробки великої кількості запитів.
3. Моделювання імітації великого навантаження та оцінка реакції системи.

4. Визначення оптимальних параметрів для забезпечення швидкодії та відмінної продуктивності.

- Задача 3: Компанія із глобальною аудиторією бажає вибрати стратегію кешування, яка ефективно працює при географічному розподілі користувачів. Наш продукт може допомогти їм знайти оптимальний підхід.

Сценарій використання:

1. Завантаження географічних даних про розташування користувачів та їх запитів.
2. Порівняння різних стратегій кешування під час географічного розподілу запитів.
3. Моделювання реальних умов роботи мережі CDN при розподіленому використанні.
4. Оцінка продуктивності та рекомендації для оптимального вибору стратегії.

- Задача 4: Компанія із змінною активністю користувачів шукає рішення для ефективного кешування при різких змінах обсягу запитів. Наш продукт може допомогти їм підтримувати стабільність та продуктивність системи.

Сценарій використання:

1. Моніторинг динаміки та розподілу обсягу запитів у реальному часі.
2. Аналіз реакції різних стратегій кешування на різкі зміни активності користувачів.
3. Впровадження параметрів, що підтримують ефективність при зміні обсягу запитів.

4. Порівняння результатів та надання рекомендацій для оптимальної стратегії.

- Задача 5: Компанія, яка часто вносить зміни на головний сервер, шукає стратегію кешування, яка ефективно оновлює дані на всіх регіональних серверах. Наш продукт може допомогти їм забезпечити швидке оновлення та консистентність даних.

Сценарій використання:

1. Налаштування параметрів стратегії оновлення даних на головному сервері.
2. Моделювання реальних сценаріїв оновлення та розповсюдження даних по регіональних серверах.
3. Оцінка часу оновлення та консистентності даних під час великої кількості змін.
4. Рекомендації щодо оптимальної стратегії оновлення для максимальної ефективності.

3.3.2 Довідник користувача

Для того щоб максимально використовувати переваги нашого продукту в реальних умовах та оптимізувати його використання для досягнення найкращих результатів опишемо довідник по використанню. Незалежно від вашої галузі або завдань, інструкції користувача допоможуть вам впроваджувати наш продукт з максимальною користю та результативністю.

Крок 1: Запуск Програми

Почніть з запуску програми на вашому пристрої. Переконайтеся, що у вас встановлена остання версія програми для забезпечення стабільної та ефективної роботи.

Для успішного використання нашого програмного продукту та проведення досліджень стратегій кешування в архітектурі необхідно встановити дві ключові компоненти: Java та IntelliJ IDEA (Integrated Development Environment). Java є важливим компонентом, оскільки наш продукт використовує цю мову програмування для свого функціонування.

IntelliJ IDEA - це інтегроване середовище розробки, яке спрощує процес створення, редагування та компіляції програмного коду. Для встановлення IntelliJ IDEA.

Після встановлення цих компонентів ви будете готові використовувати наш програмний продукт для аналізу стратегій кешування в мікросервісній архітектурі. Переконайтеся, що ваша робоча обстановка готова до дослідження та налаштована для ефективної роботи.

Крок 2: Налаштування Параметрів

Перед використанням продукту визначте параметри відповідно до вашого конкретного сценарію. Виберіть стратегії кешування та інші налаштування залежно від вашого завдання. Для симуляції необхідно ініціалізувати визначену кількість кожного об’єкту, наприклад CDN провайдери, сервери та користувачів. Кожен з об'єктів має свої параметри, які необхідно задати при їх створенні. Саме завдяки цим параметрам система буде якомога реалістичнішою.

Крок 3: Запуск Симуляції

Щоб запустити симуляцію кожної кожного із сценаріїв, необхідно перейти в відповідний клас та запустити main() функцію, яка розпочне симуляцію. Після завершення симуляції у консолі будуть виведені метрики для подальшого аналізу.

Отже, запускаючи програму відповідно до інструкції, користувач отримує можливість проводити дослідження з використанням реалістичних сценаріїв та враховувати власні потреби та умови. Це дозволяє вам не лише отримати результати порівняння стратегій кешування, але й адаптувати програму до конкретних вимог вашого дослідження.

ВИСНОВОК ДО РОЗДІЛУ 3

У підсумку проведеного дослідження ефективності стратегій кешування в CDN мережах для різних сценаріїв, можна визначити, що вибір оптимальної стратегії значущо залежить від конкретних умов та вимог системи. Розглянуті сценарії, такі як велика кількість запитів, географічний розподіл користувачів, зміна активних користувачів та часті оновлення даних на головному сервері, дозволили виявити переваги та обмеження кожної стратегії у різних умовах.

Локальна стратегія виявилася особливо ефективною у сценарії з великою кількістю запитів, вона має найменший час отримання відповіді 125 мс, але може стикатися з обмеженнями при значній кількості користувачів. У сценарії географічного розподілу користувачів централізована стратегія демонструє високу ефективність, особливо при рівномірному розподілі користувачів. Локальна стратегія виявляється оптимальною у випадку концентрації користувачів в одному регіоні. Сценарій зміни активних користувачів підтвердив стабільність централізованої та розподіленої стратегій, тоді як локальна стратегія може виявитися неефективною при різких змінах обсягу запитів. У сценарії оновлення даних на головному сервері централізована та розподілена стратегії представляють стабільні та ефективні підходи, тоді як локальна стратегія може мати обмежену ефективність у зв'язку з оновленнями на головному сервері.

Отже, кожна стратегія кешування має свої унікальні переваги та обмеження, і вибір оптимальної стратегії повинен здійснюватися з урахуванням конкретного сценарію та характеристик мережі CDN. Враховуючи різноманітні умови та вимоги систем, рекомендується вибирати стратегію кешування для досягнення оптимальної ефективності системи в конкретних умовах використання.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Стартап, спрямований на дослідження продуктивності мікросервісів через кешування даних, є важливим етапом, оскільки він не лише реалізовує ідею дослідження, але й створює основу для подальшого наукового внеску. Цей розділ детально описує етапи розробки стартап-проєкту, включаючи його мету, об'єкт та предмет дослідження, архітектурні рішення, технологічний стек та стратегію впровадження.

Метою стартап-проекту є розробка інноваційного рішення, спрямованого на вдосконалення продуктивності мікросервісної архітектури шляхом ефективного кешування даних. Основною метою є створення програмного продукту, який забезпечить оптимізацію роботи мікросервісів та вдосконалить їхню швидкодію.

4.1 Опис ідеї проекту

Першим кроком є опис ідеї стартап-проекту, що дасть цілісне уявлення про його зміст та перспективні ринки, де варто шукати потенційних клієнтів. Ідея стартап-проєкту описана у таблиці 4.1.

Таблиця 4.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка технології кешування для підвищення продуктивності мікросервісної архітектури.	Оптимізація роботи мікросервісів у фінансовому секторі	Зменшення часу відповіді для фінансових операцій.
	Використання у сфері електронної комерції	Підвищення швидкості завантаження сторінок та оформлення замовлень.

	Впровадження в системи зберігання та обробки медичної інформації	Забезпечення швидкого доступу до важливих медичних даних.

Таблиця 4.2 – Визначення сильних, слабких, та нейтральних характеристик ідей проекту

№ п/п	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів			W слабка сторона	N нейтральна сторона	S сильна сторона
		Мій проект	FastCache Solutions	DataSpeed Innovations			
1	Оптимізація роботи мікросервісів	Спрямований на вдосконалення продуктивності мікросервісів через ефективне	Має досвід у фінансовому секторі, забезпечуючи швидкодію для фінансов	Використовує патентовану технологію кешування для оптимізації мікросервісів.			+

		кешування даних.	их операцій.				
2	Застосування в різних сферах	Має напрямки застосування в фінансовому секторі, електронній комерції та системах зберігання медичної інформації.	Зосереджений на фінансовому секторі та електронній комерції.	Має гнучку архітектуру для застосування в різних секторах.			+
3	Технологічна інноваційність	Використовує інноваційні рішення для ефективн	Застосовує загальні технології кешування.	Має патентовану технологію кешування.			+

		ого кешування даних у мікросервісній архітектурі					
	Гнучкість і розширюваність	Має великий потенціал для оптимізації та розширення функціоналу	Обмежені можливості технічного розвитку.	Має можливості для додаткових функцій та розширення.			

4.2 Технологічний аудит ідеї проекту

Технологічний аудит ідеї проекту визначає ключові аспекти технічної сторони нашого стартапу, спрямованого на дослідження продуктивності мікросервісів через кешування даних. Технологічний аудит є необхідним кроком для визначення стійкості та готовності ідеї до впровадження, а також визначає потенціал для подальшого технічного розвитку.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Розробка симуляційного середовища	Java, Spring Framework, IntelliJ IDEA	Наявна	Доступна, безкоштовно
Обрана технологія реалізації ідеї проекту : є цілком можливою				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Аналіз ринкових можливостей є ключовим етапом стратегічного планування перед запуском стартап-проекту. Далі ми детально розглянемо перспективи ринкових можливостей для успішного впровадження ідеї проекту з дослідження продуктивності мікросервісів через кешування даних.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристика
1.	Кількість головних гравців	На ринку існує приблизно 10 головних гравців, що визначають конкурентну обстановку.
2.	Загальний обсяг продаж, грн/ум.од	Загальний обсяг продаж на ринку становить приблизно 50 млн. грн на одиницю товару.

3.	Динаміка ринку (якісна оцінка)	Ринок демонструє позитивну динаміку, підтверджену зростанням обсягів продажів та попиту на інноваційні рішення.
4.	Наявність обмежень для входу (вказати характер обмежень)	Обмеження для входу включають велику конкуренцію та високі витрати на рекламу та розвиток дистриб'юційної мережі.
5.	Специфічні вимоги до стандартизації та сертифікації	Ринок висуває високі стандарти та вимоги до сертифікації, що свідчить про важливість якості продуктів.
6.	Середня норма рентабельності в галузі (або по ринку), %	Середня норма рентабельності в галузі становить близько 35%, що свідчить про прийнятні умови для ведення бізнесу.

Визначимо потенційні групи клієнтів, їх характеристики, та сформуємо орієнтовний перелік вимог до товару для кожної групи

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Підвищення продуктивності мікросервісів через кешування даних	Фінансовий сектор, електронна комерція, системи зберігання та обробки медичної інформації	Фінансовий сектор може акцентувати увагу на часі відповіді та безпеці даних, електронна комерція - на швидкості завантаження та зручній навігації, а системи зберігання медичної інформації - на доступності до важливих медичних даних.	Висока ефективність, безпека, швидкодія, легка інтеграція.

Аналіз ринкового середовища є ключовим етапом для визначення можливостей та загроз перед запуском стартап-проекту. Нижче наведено

таблиці факторів, які сприяють ринковому впровадженню проекту (Таблиця 4.6), а також тих, що можуть ускладнити його реалізацію (Таблиця 4.7).

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Високий рівень конкуренції на ринку кешування	Погіршення умов конкуренції, зменшення частки ринку.	Зосередження на інноваціях, вдосконалення продукту та розвиток нових ринків.
2.	Сприйняття клієнтів щодо ризиків у роботі з даними	Зменшення довіри клієнтів, можливі втрати клієнтської бази.	Запровадження високих стандартів безпеки, активна комунікація та освітлення переваг системи.
3.	Технічні виклики при інтеграції з існуючими системами	Затримки в розробці та втрати від непрацюючих інтеграцій.	Активне вирішення технічних проблем, ефективна комунікація з клієнтами та партнерами.
4.	Низька освідомленість ринку про проблеми мікросервісів.	Обмежений інтерес та попит на продукт.	Інтенсивна реклама, освітлення проблем та переваг мікросервісів, участь у конференціях та виставках.

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зростання попиту на оптимізацію	Збільшення можливостей	Швидке впровадження стратегій розширення

	мікросервісів через кешування даних	здобуття нових клієнтів та ринків.	ринку та збільшення виробничих потужностей.
2.	Активний розвиток технологій кешування	Можливість створення та вдосконалення високотехнологічних продуктів.	Зосередження на дослідженнях та вдосконаленні технічних характеристик продукту.
3.	Потреба у високій швидкості обробки даних	Зростання вимог ринку до продуктів з високою швидкодією.	Посилення інженерних зусиль для оптимізації швидкодії та пошук нових технологічних рішень.

Аналіз конкурентного середовища є важливим кроком у розумінні та плануванні стратегії підприємства. В таблиці 4.8 наведено ступеневий аналіз конкуренції на ринку, що включає важливі аспекти конкурентного середовища та їх вплив на діяльність підприємства.

Таблиця 4.8 – Степеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції	Монополія	Розробка унікальних пропозицій, акцент на високій якості та інноваціях.

Рівень конкурентної боротьби	Національний	Активна маркетингова політика, розширення географії обслуговування.
Галузева ознака	Внутрішньогалузева	Співпраця з партнерами, розширення асортименту продукції.
Конкуренція за видами товарів	Товарно-видова	Розробка унікальних функцій та характеристик продукції.
Характер конкурентних переваг	Нецінова	Покращення обслуговування клієнтів, інновації у продукції.
Інтенсивність конкуренції	Марочна	Підвищення рекламної активності, побудова унікального бренду.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Amazon Web Services (AWS) - провідний постачаль	Спеціалізована стартап-компанія CacheTech, яка	Постачальники технічних компонентів для	Компанії у фінансовому секторі, електронній комерції,	Існуючі аналогічні технології, проте менш

	ник хмарних технологій з власними рішеннями для кешування даних.	вивчає впровадження подібних технологій.	кешування даних.	медичні установи.	продвинуті.
Висновки:	Сильна конкуренція від AWS, яка має велику частку ринку та широкий спектр клієнтів.	Потенційна загроза від стартапу CacheTech, який може принести інноваційні підходи	Постачальники не мають значного впливу на умови роботи.	Клієнти мають обмежений вплив на умови роботи.	Невелика загроза від менш продвинутих технологій

Аналіз конкуренції та урахування характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища дозволяють визначити та обґрунтувати фактори, які визначатимуть конкурентоспроможність нашого стартап-проекту. У таблиці 4.10 наведено перелік та обґрунтування цих факторів.

Таблиця 4.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1.	Технологічна перевага	Проект має порівнння технологій кешування даних, що дозволяє об’єктивно поріявняти підходи до кешування та досягти вищої продуктивності
2	Цінова конкурентоспроможність	Порівняно з конкурентами, ми пропонуємо більш доступні ціни за впровадження та обслуговування, забезпечуючи високу вартість на ринку.
3	Гнучкість рішень	цей проект надає гнучкі та налаштовані рішення, враховуючи потреби різних клієнтів у фінансовому секторі, електронній комерції та медичних установах.
4	Підтримка клієнтів	Забезпечуємо високий рівень підтримки та сервісу, включаючи навчання персоналу, регулярні оновлення та швидке

		вирішення проблем для клієнтів.
5	Розширений функціонал	Проект має розширений функціонал для ефективного використання кешування даних у різних сферах, таких як фінанси, електронна комерція та медицина.

Таблиця 4.11 - Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з власним проектом						
			-3	-2	-1	0	+1	+2	+3
1	Технологічна перевага	15					Cache Tech		
2	Цінова конкурентоспроможність	15		Cache Tech					
3	Гнучкість рішень	17			Cache Tech				
4	Підтримка клієнтів	15				Cache Tech			
5	Розширений функціонал	10					Cache Tech		

Таблиця 4.12 - SWOT аналіз

Сильні сторони 1. Високотехнологічні рішення 2. Конкурентоспроможні ціни 3. Гнучкість та адаптивність системи 4. Великий функціонал та можливості	Слабкі сторони 1. Недостатня відомість на ринку 2. Обмежений досвід в розробці проектів 3. Залежність від постачальників 4. Невизначеність щодо легалізації
Можливості 1. Зростання попиту на технології кешування 2. Розширення ринків застосування 3. Партнерства з ключовими гравцями 4. Зростання свідомості про продукт	Загрози 1. Збільшення конкуренції на ринку 2. Зміни в законодавстві та регулюванні 3. Технічні або безпекові проблеми 4. Економічні нестабільності

Таблиця 4.13 - Альтернативи ринкового впровадження стартап-проекту

Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Розробка інтенсивної маркетингової кампанії для підвищення свідомості про продукт	Висока	3-6 місяців
Партнерство з ключовими гравцями в галузі для розширення ринків застосування	Середня	6-9 місяців
Зосередження на стратегії диференціації продукту для створення унікальності	Висока	9-12 місяців

4.4 Розроблення ринкової стратегії проекту

Після визначення цільових груп потенційних споживачів та стратегії охоплення ринку, наступним етапом є розроблення ринкової стратегії для стартап-проекту. Це включає в себе визначення конкурентних переваг та позиціонування на ринку, що дозволить ефективно взаємодіяти з цільовими групами та вирізнятися серед конкурентів.

Таблиця 4.14 - Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційни	Готовність споживачі в сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Фінансовий сектор	Висока	Великий	Висока	Середня
2	Електронна комерція	Висока	Великий	Висока	Висока
3	Системи зберігання та обробки медичної інформації	Середня	Середній	Середня	Висока
Які цільові групи обрано: компанія працює із всім ринком, пропонуючи стандартизовану програму					

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегії охоплення ринку	Ключові конкурентноспроможні позиції відповідно до обраної льтернативи	Базова стратегія розвитку
1.	Впровадження нового функціоналу	Охоплення нових сфер бізнесу	Збільшення конкурентно спроможності	Стратегія диференціації

Таблиця 4.16 - Вибір цільових груп потенційних споживачів

№ п/п	Чи є проект першопрохідцем на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів	Чи буде компанія копіювати оновні характеристики товару конкурента і які?	Стратегія конкурентної поведінки
1.	Ні	Шукати нових споживачів, а також забирати у конкурентів	Завдяки тому що певний функціонал може бути скопійованим, то користувачі конкурентів зацікавляться	Виклик лідера

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного	Вибір асоціацій, які мають сформувати комплексну позицію власного

			стартап-проекту	проекту (три ключових)
1.	Висока продуктивність мікросервісів	Стратегія лідера	Використання найновітніших технологій кешування даних та аналізу	Інноваційні технології, Ефективність, Найновітніші рішення
2.	Зниження латентності та підвищення швидкості відгуку мікросервісів	Стратегія лідера	Забезпечення максимальної швидкості та мінімальної затримки	Інноваційні рішення, Ефективність, Перевершення конкурентів
3.	Інтеграція ефективних алгоритмів кешування даних	Стратегія лідера	Використання оптимальних алгоритмів для найкращої ефективності	Ефективність, Інновації, Продуктивність

В підсумок, розглядаючи різні аспекти ринкового аналізу для стартап-проекту, було вироблено комплексну стратегію, що охоплює розвиток, конкурентну поведінку та позиціонування на ринку. З врахуванням усіх зібраних даних та аналізу отриманої інформації, розроблено систему рішень, спрямовану на успішний вихід та утримання стартап-проекту на конкурентному ринку.

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Ефективність мікросервісів через кешування даних	Швидкий доступ до даних, оптимізація продуктивності	Інноваційна технологія мікросервісів, оптимізація продуктивності, скорочення технічного боргу
2	Підвищення стабільності систем	Уникнення технічного боргу та покращення стабільності	Скорочення технічного боргу, гнучкість та масштабованість
3	Гнучкість та масштабованість	Легке впровадження та розширення	Гнучкість та масштабованість, аналітика продуктивності

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
І. Товар за задумом	Симуляція стратегій кешування даних задля отримання кількісних характеристик, щоб покращити загальну ефективність роботи системи.		
	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор

II.Товар у реальному виконанні	Просте і швидке розгортання системи у своїх проектах	Нм	Вр/Тх
	Якість: дотримання стандартів розробки архітектури та використання сучасних технологій		
	Сервіс		
	Марка: CacheImpr		
III.Товар із підкріпленням	До продажу: документація, знижки при довгостроковому використанні.		
	Після продажу: технічна підтримка, консультація, впровадження нового функціоналу		
За рахунок чого потенційний товар буде захищено від копіювання: реєстрація права на інтелектуальну власність, закритий код продукту.			

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари замінники	Рівень цін на товари аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар послугу
	8000 грн/міс	15000 грн/міс	20-30 млн євро/рік	10000-50000 грн

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Збір інформації, порівняння альтернатив, рішення з акцентом на якість	Консультація, надання інформації, підтримка під час прийняття рішення	Короткий	Прямий канал збуту (безпосередні продажі)
2.	Низька ціна, вигідні умови оплати та доставки	Ефективне управління запасами, оперативні поставки, гнучка система ціноутворення	Широкий	Залучена система збуту (через посередників)
3.	Схильність до довгострокових відносин та партнерства	Розвиток програм лояльності, гарантійне обслуговування, пропозиції для постійних клієнтів	-	Комбінований канал збуту (прямий та залучений)

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користують	Ключові позиції, обрані для	Завдання рекламного повідомлення	Концепція рекламного звернення

		ся цільові клієнти	позиціонування		
1.	Приділяють увагу соціальним мережам та блогам	Соціальні мережі, блоги, онлайн-форуми, електронна пошта	Інноваційність, зручність, екологічність	Зацікавити, підтримати імідж бренду	Емоційна реклама, спрямована на побудову іміджу
2.	Шукають інформацію в інтернеті перед покупкою	Пошукові системи, відгуки покупців, онлайн-реклама	Якість, надійність, порівняльна перевага	Переконати у перевагах продукту	Раціональна реклама, де надаються факти та аргументи
3.	Активно взаємодіють з мобільними додатками	Мобільні додатки, SMS-реклама, push-сповіщення	Унікальність, персоналізація, зручність використання	Залучити увагу та зацікавленість	Реклама з акцентом на індивідуалізацію та зручність

ВИСНОВКИ ДО РОЗДІЛУ 4

В розділі “Розробка стартап-проєкту” було вивчено та обґрунтовано ключові аспекти маркетингового плану для успішної реалізації проекту. Проведений аналіз ринкового середовища, визначено цільові сегменти та розроблено стратегії позиціонування та конкурентної поведінки.

Проведений аналіз показав, що існує потенціал для ринкової комерціалізації проекту. Попит на пропонований продукт/послугу підтверджений, і спостережується позитивна динаміка на ринку.

Впровадження проекту має перспективи, оскільки ідентифіковані потенційні групи клієнтів та переборені можливі бар'єри входження. Конкурентний аналіз підтверджує конкурентоспроможність проекту.

Висновок щодо доцільності подальшої імплементації проекту є позитивним. Імплементація проекту має потенціал для успіху та може призвести до досягнення стратегічних цілей.

ВИСНОВКИ

У результаті проведеного наукового дослідження, присвяченого аналізу та оптимізації стратегій кешування в мікросервісних архітектурах у контексті CDN мереж, вдалося отримати висновки та рекомендації, які можуть бути корисними для розробників, архітекторів та інженерів, що працюють у цій області.

Аналіз предметної області в першому розділі виявив ключові аспекти мікросервісних архітектур, їх переваги над традиційними монолітними системами та вплив CDN провайдерів на глобальному рівні. Подальший огляд стратегій кешування, включаючи локальне, централізоване та розподілене кешування, дав можливість визначити їхні переваги та недоліки в різних умовах використання.

У другому розділі було проведено проектування симуляційного середовища, що дозволило ефективно моделювати різні умови мережі та виконувати експериментальне дослідження продуктивності за різних стратегій кешування. Результати цього експерименту дали можливість отримати конкретні дані для подальшого аналізу.

Третій розділ включав експериментальне дослідження залежності продуктивності від стратегії кешування та аналіз отриманих результатів. Різні сценарії, такі як велика кількість запитів, географічний розподіл користувачів, зміна активних користувачів та часті оновлення даних на головному сервері, були ретельно досліджені. За результатами симуляцій різних сценаріїв використання стратегій кешування в мікросервісній архітектурі можна визначити оптимальний вибір стратегії для конкретних умов.

У сценарії високої частоти запитів, локальне кешування продемонструвало найкращі результати з низьким часом відповіді (125 мс) та високою частотою використання кешу (57%), порівняно з централізованим (час відповіді 267 мс) та розподіленим (час відповіді 183 мс) кешуванням.

У випадку географічно рівномірного розподілу користувачів, централізоване кешування виявилося оптимальним з найменшим часом відповіді (121 мс) та високою частотою використання кешу (77%), в порівнянні з локальним (час відповіді 178 мс, частота використання 50%) та розподіленим (час відповіді 187 мс, частота використання 63%) кешуванням.

При концентрації користувачів в одному регіоні, локальне кешування знову виявилося оптимальним, з найменшим часом відповіді (124 мс) та ефективним використанням кешу (54%), порівняно з централізованим (час відповіді 263 мс, частота використання 67%) та розподіленим (час відповіді 199 мс, частота використання 55%) кешуванням.

У сценарії оновлення даних на головному сервері централізована та розподілена стратегії представляють стабільні та ефективні підходи, тоді як локальна стратегія може мати обмежену ефективність у зв'язку з оновленнями на головному сервері виникають затримки у оновленні всіх локальнх кешів.

Отже, результати наших експериментів підтверджують, що оптимальний вибір стратегії кешування залежить від конкретного сценарію використання та характеристик мережі. Це вказує на необхідність уважного вибору стратегії для досягнення оптимальної ефективності системи.

У четвертому розділі була розроблена концепція стартап-проєкту на основі отриманих результатів та вивчених потреб ринку. Опис ідеї проєкту, технологічний аудит та маркетингова стратегія були детально висвітлені.

Результат дослідження полягає в тому, що вибір оптимальної стратегії кешування в мікросервісних архітектурах є складним завданням, яке вимагає ретельного врахування специфіки конкретної системи та умов експлуатації. Тому спроектоване симуляційне середовище має неабияку цінність у подальшій науковій або професійній діяльності, адже використання даної розробки для різних галузей, включаючи IT-компанії, компанії з великим обсягом запитів, глобальні компанії та ті, що працюють зі змінною активністю користувачів, дозволяє здійснити повноцінний аналіз стратегій та обрати для власних цілей найбільш оптимальне рішення. Враховуючи те, що симуляційне середовище гнучке до налаштувань під власну архітектуру мережі, це означає, що продукт не втратить свою актуальність та буде корисним.

ПЕРЕЛІК ПОСИЛАНЬ

1. Pacheco V. F. Microservice Patterns and Best Practices / Vinicius Feitosa Pacheco. – 254 c
2. Barker T. Intelligent Caching / O'Reilly Media, Inc.- 107 c
3. Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley - 52 c.
4. Парфенов В.И., Золотарев С.В. Об одном алгоритме решения задачи оптимальной маршрутизации по критерию средней задержки // Вестник ВГУ: сер. Физика. Математика. – 2007. – № 2. – С. 28–32.
5. Шварц М. Сети связи: протоколы, моделирование и анализ: пер. с англ.: Ч. 1. – М.: Наука, 1992. – 336 с.
6. Yaw-chung Chen. Improving Quality of Experience in P2P IPTV // Network Operations and Management Symposium (APNOMS) 18th. – Asia-Pacific, 2016. – P. 6-9.
7. Klymash M., Kyryk M., Pleskanka N., Yanyshyn V. Data Buffering Multilevel Model at a Multiservice Traffic Service Node // Smart Computing Review. – 2014. – Vol. 4. No. 4. – P. 294-306.
8. Bai Y., Jia B., Zhang J., Pu Q. An Efficient Load Balancing Technology in CDN // Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on. – IEEE, 2009. – Т. 7. – P. 510-514.
9. Дмитриев Г.А., Марголис Б.И., Музанна М.М. Решение задачи оптимальной маршрутизации по критерию загруженности сети // Программные продукты и системы. – 2013. – № 4. – С. 17–19.
10. Pallis George, Konstantinos Stamos, Athena Vakali «and other». Replication based on Objects Load under a Content Distribution Network // Proceedings of the 22nd International Conference on Data Engineering Workshops,

ICDE 2006, 3-7 April 2006, Atlanta, GA, USA. – P. 1-9.

11. Wauters T., Coppens J., Dhoedt B., Demeester P. Load balancing through efficient distributed content placement // In proceeding of: Next Generation Internet Networks. – NY, 2005. – P. 99–105.
12. Haghighi A., Mishev D. Queueing models in industry and business. – New York: Nova Science Publishers, 2008. – 386 p.
13. Dattatreya G. Performance analysis of queuing and computer networks. – Boca Raton: CRC Press, 2008. – 449 p.
14. Клейнрок Л. Теория массового обслуживания. Пер. с англ. / Пер. И. И. Грушко; ред. В. И. Нейман. – М.: Машиностроение, 1979. – 512 с
15. Хоміч Л.І. Яременко В.С. Дослідження продуктивності мікросервісних ахітектур через кешування даних. “Системні науки та інформатика”, 4-8 грудня 2023 року, Київ - с.370-375
16. Sayan Dey. How to set up Amazon CloudFront with WordPress Website [Електронний ресурс] // 2023 – Режим доступу https://bloggingmetrics.com/setup-amazon-cloudfront-with-wordpress/