МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГАДЕВИЧ Володимир Юрійович

Метод вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server за допомогою машинного навчання / Method for Selecting Hardware Resources for Streaming Services Based on Ant Media Server Using Machine Learning

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
В.Ю. Гадевич
_____

Науковий керівник:
к.т.н., доцент П.Є. Биковий
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.М. Васильків

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Гадевич Володимир Юрійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи

Метод вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server за допомогою машинного навчання / Method for Selecting Hardware Resources for Streaming Services Based on Ant Media Server Using Machine Learning
керівник роботи к.т.н., доцент П.Є. Биковий,
затверджені наказом по університету від 12 грудня 2023 року № 753.

2. Строк подання студентом закінченої кваліфікаційної роботи 4 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- огляд предметної області стрімінгових сервісів;
- аналіз методів машинного навчання для вибору апаратних ресурсів;
- огляд існуючих рішень для автоматизації управління апаратними ресурсами для стрімінгових сервісів;
- постановка завдання дослідження;
- розробка методології створення системи автоматизованого вибору апаратних ресурсів;
- опис використаних даних та методів їхньої обробки;
- реалізація порівняльного аналізу методів прогнозування навантаження апаратних ресурсів;
- тестування та оцінка ефективності системи вибору апаратних ресурсів;

5. Перелік графічного матеріалу у роботі
- схема архітектури системи вибору апаратних ресурсів для Ant Media Server;
- графіки результатів порівняння методів прогнозування навантаження на ресурси.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.05.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту.	до 30.11.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2024 р.	

Студент _____ В.Ю. Гадевич
підпис

Керівник роботи _____ к.т.н., доцент П.Є. Биковий
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server за допомогою машинного навчання» виконана для здобуття освітнього ступеня «Магістр» за спеціальністю 122 «Комп'ютерні науки» в межах освітньо-професійної програми «Комп'ютерні науки» написана обсягом в 87 сторінок і містить 19 ілюстрацій, 3 додатки та 54 використаних джерела.

Метою роботи є створення методу вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server для ефективного управління навантаженням, оптимізації витрат на інфраструктуру, забезпечення стабільної роботи стрімінгових трансляцій та прогнозування навантаження на систему за допомогою методів машинного навчання.

Методи дослідження, використані у роботі, методи машинного навчання (Auto Regression Integrated Moving Average, Bayesian Ridge Regression, Gradient Boosting Regressor, Support Vector Regressor, Long Short-Term Memory).

Результати дослідження: Кваліфікаційна робота включає розробку методу вибору апаратних ресурсів для стрімінгових сервісів і прогнозування навантаження на системи. Основним досягненням дослідження стало створення моделі, яка здатна точно прогнозувати потреби в ресурсах для оптимізації роботи стрімінгових сервісів на базі Ant Media Server.

Практичне значення роботи полягає в розробці методу для оптимального вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server. Рекомендації для подальших досліджень включають удосконалення моделей прогнозування навантаження шляхом впровадження нових методів машинного навчання.

Ключові слова: ANT MEDIA SERVER, АНАЛІЗ ДАНИХ, ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ, АНСАМБЛЕВІ МЕТОДИ, АВТОМАТИЗАЦІЯ АНАЛІЗУ.

ABSTRACT

The qualification work on the topic «Method for Selecting Hardware Resources for Streaming Services Based on Ant Media Server Using Machine Learning» for Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 87 pages and it contains 19 figures, 3 annexes and 54 sources.

The aim of the work is to create a method for selecting hardware resources for streaming services based on Ant Media Server for efficient load management, optimising infrastructure costs, ensuring stable operation of streaming broadcasts and predicting the system load using machine learning methods.

Research methods used in the work, machine learning methods (Auto Regression Integrated Moving Average, Bayesian Ridge Regression, Gradient Boosting Regressor, Support Vector Regressor, Long Short-Term Memory).

Research results: The qualification work includes the development of a method for selecting hardware resources for streaming services and forecasting the load on systems. The main achievement of the study was the creation of a model that can accurately predict the resource requirements to optimise the performance of streaming services based on Ant Media Server.

The practical significance of this work is the development of a method for the optimal selection of hardware resources for streaming services based on Ant Media Server. This solution is useful for ensuring stable operation of streaming platforms and reducing infrastructure costs.

Recommendations for further research include improving load prediction models by introducing new machine learning methods.

Keywords: ANT MEDIA SERVER, DATA ANALYSIS, FORECASTING, MACHINE LEARNING, ENSEMBLE METHODS, ANALYSIS AUTOMATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ARIMA - Auto Regression Integrated Moving Average

AS - Autonomous System

BGP - Border Gateway Protocol

BRR - Bayesian Ridge Regression

CDN - Content Delivery Network

DASH - Dynamic Adaptive Streaming over HTTP

HLS - HTTP Live Streaming

LSTM - Long Short-Term Memory

MCDA - Multi-Criteria Decision Analysis

MPD - Media Presentation Description

MS-RTSP - Microsoft Real-Time Streaming Protocol

NN - Time Delay Neural Network

PM - Physical Machine

QoE - Quality of Experience

QoS - Quality of Service

RTMP - Real-Time Messaging Protocol

RTSP - Real-Time Streaming Protocol

SLA - Service Level Agreement

SVR - Support Vector Regression

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

VM - Virtual Machine

ВСТУП

Стрімінгові сервіси сьогодні займають важливе місце в індустрії цифрових технологій, пропонуючи користувачам доступ до контенту в реальному часі. Зростання популярності відео- і аудіо-стрімінгу вимагає від провайдерів послуг високої ефективності та стабільності роботи платформ. Водночас, для забезпечення якісного стрімінгу в умовах високого навантаження необхідна складна інфраструктура, яка здатна динамічно адаптуватися до змінюваного попиту. Стрімінгові сервіси потребують потужних апаратних ресурсів, здатних обробляти великі обсяги даних, що включають відео- і аудіопотоки, а також забезпечувати безперебійне транслювання контенту для численних користувачів. Традиційні методи аналізу, такі як технічний і фундаментальний аналіз, здебільшого базуються на історичних даних і вимагають значних витрат часу та ресурсів для досягнення оптимальних результатів. У зв’язку з цим, виникає необхідність в нових підходах до аналізу ринку, здатних автоматизувати цей процес і підвищити точність прогнозів. Сучасні технології машинного навчання дозволяють зберегти ефективність прогнозів при значно менших витратах часу і ресурсів, що відкриває нові можливості для автоматизації управління ресурсами в стрімінгових сервісах та інших подібних індустріях.

Актуальність теми дослідження зумовлена зростаючою популярністю стрімінгових сервісів та необхідністю ефективного управління апаратними ресурсами для забезпечення високої якості обслуговування користувачів. Сучасні платформи, що надають послуги відео- та аудіо-стрімінгу, стикаються з проблемами, пов'язаними з динамічними змінами навантаження, що виникають через різкі коливання кількості користувачів, пикові години чи масштабування контенту. Для забезпечення безперебійного стрімінгу необхідно розв'язувати задачу оптимального вибору та масштабування апаратних ресурсів, що здатні забезпечити стабільну роботу платформи при будь-якому рівні навантаження. Невірний вибір ресурсів або неефективне їх використання може призвести до затримок, зниження якості трансляцій, або навіть до збоїв у роботі системи. У

зв'язку з цим, актуальність теми дослідження полягає в розробці та впровадженні методів автоматизованого вибору апаратних ресурсів на основі алгоритмів машинного навчання. Такий підхід дозволить точніше прогнозувати навантаження на систему, що, у свою чергу, забезпечить економію ресурсів, зниження витрат на інфраструктуру та підвищення стабільності роботи стрімінгових сервісів. Технології машинного навчання, які використовуються для прогнозування навантаження та автоматичного масштабування ресурсів, здатні значно покращити ефективність роботи таких платформ в умовах змінного попиту та високих вимог до швидкості обробки даних.

Метою даного дослідження є розробка та тестування методу вибору апаратних ресурсів для стрімінгових сервісів на основі прогнозування навантаження за допомогою машинного навчання. Завдяки цьому методу планується оптимізувати процес масштабування інфраструктури, забезпечивши стабільну роботу стрімінгових платформ при змінному навантаженні, знизити витрати на інфраструктуру та підвищити ефективність використання апаратних ресурсів.

Об’єктом дослідження є інфраструктура стрімінгових сервісів, зокрема процеси вибору та масштабування апаратних ресурсів на основі прогнозування навантаження на системи. Предметом дослідження є алгоритми та методи, застосовувані для прогнозуванням навантаження на стрімінгові системи та алгоритми автоматичного вибору апаратних ресурсів для забезпечення оптимальної роботи стрімінгових платформ на базі Ant Media Server.

Для досягнення поставленої мети були використані наступні методи дослідження: аналіз існуючих рішень для прогнозування навантаження та вибору апаратних ресурсів на стрімінгових платформах, огляд алгоритмів машинного навчання для прогнозування навантаження, моделювання навантаження на Ant Media Server, розробка методу автоматичного вибору апаратних ресурсів на основі прогнозів, а також оцінка ефективності запропонованого методу порівняно з існуючими підходами.

Наукова новизна дослідження полягає в розробці методу вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server з використанням машинного навчання для прогнозування навантаження на сервіс, що дозволяє значно зменшити час на аналіз та прийняття рішень.

Практичне значення роботи полягає в розробці методу автоматизованого вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server з використанням методів машинного навчання для прогнозування навантаження.

Апробація результатів дослідження здійснювалася шляхом представлення основних теоретичних положень і практичних результатів на Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі» (м. Тернопіль, Україна), а також на Міжнародній науково-практичній інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (м.Тернопіль, Україна).

1 ТЕОРЕТИКО-МЕТОДИЧНІ ЗАСАДИ АНАЛІЗУ СТРІМІНГОВИХ СЕРВІСІВ

1.1 Аналіз предметної області

Стрімінгові сервіси, такі як YouTube, Netflix, Twitch і Facebook Live, відіграють важливу роль у сучасному цифровому середовищі, забезпечуючи користувачів доступом до аудіо- та відеоконтенту в реальному часі [2-5]. Їх популярність продовжує зростати завдяки інтерактивним можливостям і високій якості обслуговування. Однак із ростом попиту виникають нові виклики: як ефективно управляти ресурсами, адаптуватися до пікових навантажень і забезпечувати оптимальну якість для користувачів.

Стрімінгові сервіси стали основними платформами для надання контенту в реальному часі. Вони включають відео- та аудіо-стрімінг, який забезпечує миттєвий доступ до мультимедійного контенту. Такі сервіси вимагають масштабованої інфраструктури для обробки великих обсягів даних та забезпечення низької затримки для користувачів у будь-який час.

Стрімінгові сервіси, зокрема платформи, що використовують Ant Media Server, потребують постійного моніторингу навантаження на сервери та адаптації до змінного попиту [1]. З кожним роком зростає кількість користувачів, роздільна здатність відео, кількість одночасних потоків і обсяги переданих даних. Це створює потребу в ефективному управлінні інфраструктурою, щоб забезпечити стабільну роботу без перевантаження.

Окрім цього, Ant Media Server відкриває можливості для створення інтерактивного відеоконтенту. Наприклад, у програмах Store Manager можна впровадити динамічні відеокерівництва, які наочно демонструють функціонал і особливості продуктів. Це допоможе користувачам швидше адаптуватися до нових функцій та працювати з продуктами максимально ефективно.

Використання симуляцій для перевірки ефективності систем є ще одним перспективним кроком для компанії. Розробка інструментів моделювання дозволила б тестувати роботу продуктів перед їх впровадженням, оцінювати розподіл ресурсів та мінімізувати ризики. Такий підхід сприятиме створенню ще

більш надійних та ефективних продуктів, які відповідатимуть високим очікуванням клієнтів.

1.2 Методи машинного навчання для аналізу стрімінгових сервісів

Важливим завданням системи стрімінгового відео є вибір серверів, які доставляють контент таким чином, щоб ефективно використовувалися мережеві та серверні ресурси і надавалась бажана якість досвіду користувача (QoE). Це завдання можна формалізувати як проблему багатокритерійної оптимізації, яка враховує різні статичні та динамічні атрибути компонентів системи (мережі, сервери, клієнти).

Загалом ці проблеми є NP-повними, однак можуть бути ефективно розв'язані за допомогою евристичних методів, таких як алгоритми багатокритерійного прийняття рішень (MCDA) [6]. Ці алгоритми потребують інформації про поточні ресурси мережі та серверів, що вимагає дороговартісної інфраструктури моніторингу ресурсів.

Багато сучасних досліджень фокусуються на використанні машинного навчання для прогнозування параметрів мережі та вибору оптимального протоколу. Алгоритми машинного навчання активно застосовуються для аналізу таких параметрів, як затримка, втрата пакетів та пропускна здатність в реальному часі. Зокрема, LSTM-моделі демонструють високу ефективність при роботі з часовими рядами та прогнозуванні змін у стані мережі. Наприклад, дослідження показують, що нейронні мережі можуть автоматично обирати оптимальний протокол, враховуючи поточні умови мережі, що сприяє стабільності та підвищенню якості потокової передачі [7].

Для розв'язання задачі оптимального вибору серверів у стрімінгових системах, де важливі як продуктивність, так і витрати, було запропоновано підходи, які комбінують використання клієнтських та серверних даних. Це дозволяє знизити вимоги до інфраструктури моніторингу та покращити результати

за рахунок інтеграції інформації з боку клієнтів. Дослідження підтверджують, що поєднання MCDA з методами машинного навчання дозволяє створювати адаптивні системи, здатні забезпечувати високу якість роботи навіть за умови змінних навантажень.

У контексті хмарних обчислень актуальним є питання прогнозування навантаження на сервери для оптимізації розподілу ресурсів. Використання методів машинного навчання, таких як LSTM, ARIMA, або BRR, дозволяє досягти високої точності прогнозування для завантаження процесора та пам'яті. При цьому кластеризаційні підходи демонструють суттєве покращення точності шляхом групування завдань зі схожими паттернами навантаження перед побудовою моделей прогнозування .

Хмарні обчислення є широко використовуваною IT-послугою, яка надає різноманітні сервіси в рамках однієї платформи. Одним постачальником хмарних послуг (CSP) можуть надаватися такі послуги, як зберігання даних, обчислення та веб-хостинг. Багато компаній переходять до використання хмарних сервісів через їхню гнучкість, зокрема бізнес-модель "оплата за використання" (pay-as-you-go) [8]. Така еластичність моделі дозволяє значно знижувати витрати, оскільки відпадає необхідність створення, підтримки та масштабування великої приватної інфраструктури.

Хмарні обчислення відіграють важливу роль, дозволяючи клієнтам використовувати ресурси за моделлю pay-as-you-go. Це дозволяє уникнути надлишкового виділення ресурсів, які можуть бути не використані, що призводить до марнотратства, або недостатнього виділення ресурсів для популярних послуг, що може спричинити втрату потенційного прибутку [10].

Роботи, присвячені прогнозуванню попиту на ресурси (тобто навантаження), можна класифікувати на три категорії: статистичні підходи, підходи машинного навчання та підходи глибокого навчання.

Статистичні підходи є популярними методами прогнозування навантаження. Khan та ін. [11] дослідили повторювані шаблони навантаження віртуальних машин (VM) із точністю понад 80% і запропонували метод, заснований на прихованій

моделі Маркова, для характеристики та прогнозування цих шаблонів. Jiang та ін. [12] представили онлайн-систему для аналізу часових даних під назвою ASAP, яка моделює та прогнозує попит на віртуальні машини за допомогою моделі ковзного середнього (Moving Average, MA). Morais та ін. [13] запропонували фреймворк для реалізації авто-масштабування послуг, заснованого на методах прогнозування використання CPU, таких як автокореляція (Auto Correlation, AC), лінійна регресія (LR), авторегресія (AR), інтегрована модель авторегресії ковзного середнього (ARIMA) тощо. Gong та ін. [14] розробили систему PRESS, яка використовує підхід до прогнозування на основі відповідності шаблонів і станів. Вона спочатку застосовує методи обробки сигналів для визначення, чи є у використанні CPU повторювані шаблони. Якщо такі шаблони існують, вони використовуються для прогнозування; інакше застосовується статистичний підхід, заснований на дискретно-часовому марковському ланцюзі.

Однак більшість наборів даних є нестабільними (наприклад, мають велику варіацію між сусідніми точками або пропуски даних), а методи прогнозування часових рядів, які базуються на лінійних структурах, більше підходять для стабільних даних.

Підходи машинного навчання також широко застосовуються для прогнозування навантаження VM. Imam та ін. [15] представили нейронну мережу із затримкою часу (Time Delay Neural Network, NN) і методи регресії для прогнозування навантаження кожної VM.

Farahnakian та ін. [16] розробили стратегії вимірювання ресурсів і їхнього забезпечення, використовуючи нейронні мережі та лінійну регресію для прогнозування попиту.

Bankole та ін. [17] створили модель прогнозування клієнтів хмарних обчислень із використанням трьох моделей машинного навчання: підтримуючої векторної регресії (SVR), нейронних мереж і лінійної регресії.

Islam та ін. [18] запропонували підхід для прогнозування майбутнього завантаження CPU віртуальних машин, використовуючи нейронні мережі та алгоритми лінійної регресії. Вони дійшли висновку, що нейронні мережі

забезпечують вищу точність порівняно з лінійною регресією, а також продемонстрували залежність точності від розміру вхідного вікна.

Nikravesh та ін. [19, 20] досліджували методи прогнозування на основі SVM, нейронних мереж і лінійної регресії. Вони виявили, що для періодичних змін ресурсів VM метод SVM перевершує інші підходи за точністю. Однак на великих наборах даних (наприклад, Google Cluster Trace) точність SVM суттєво знижується, як зазначено в [31].

Gopal та ін. [22] запропонували байєсівську модель для прогнозування ресурсів VM і порівняли її з методами лінійної регресії та підтримуючої векторної регресії. Вони встановили, що за допомогою байєсівської моделі можна ефективно прогнозувати навантаження приблизно на 75% серверів у центрі даних.

Підходи глибокого навчання останніми роками активно застосовуються для прогнозування навантаження. Так Qiu та ін. [22] запропонували підхід, який включає мережу глибоких вірувань (*Deep Belief Network, DBN*) та регресійний шар для прогнозування навантаження віртуальних машин у хмарних системах.

Zhang та ін. [23] представили ефективну модель глибокого навчання, засновану на канонічній поліатомній декомпозиції, для прогнозування навантаження кожної віртуальної машини. Завдяки використанню цієї декомпозиції модель значно стискає параметри, забезпечуючи високу швидкість навчання зі зниженням точності класифікації лише на мінімальному рівні. Zhang та ін. [24] також запропонували підхід на основі DBN для прогнозування запитів на хмарні ресурси для кожного завдання. Цей підхід може застосовуватися як для довгострокового, так і для короткострокового прогнозування, забезпечуючи покращену точність порівняно з існуючими методами.

Kumar та ін. [25] розробили моделі прогнозування, засновані на мережах довготривалої короткочасної пам’яті (*Long Short-Term Memory, LSTM*) [33]. Запропоновану модель було протестовано на трьох еталонних наборах даних: журнали веб-серверів та HTTP-трейси серверів NASA, Calgary і Saskatchewan.

Song та ін. [26] застосували модель на основі LSTM для прогнозування середнього навантаження впродовж наступних часових інтервалів і реального

навантаження на кілька кроків уперед, використовуючи дані Google Cluster Trace. Ця модель досягає високої точності прогнозування в традиційній розподіленій системі.

1.3 Постановка завдання

Завдання також передбачає аналіз існуючих програмних рішень для управління апаратними ресурсами стрімінгових сервісів та прогнозування навантаження на такі системи. У сучасних стрімінгових платформах, таких як Ant Media Server, важливу роль відіграє здатність ефективно управляти інфраструктурою, що дозволяє забезпечити високу якість обслуговування користувачів та стабільність роботи при змінному навантаженні. Існуючі програмні рішення пропонують різні підходи до моніторингу ресурсів і адаптації серверів під поточний попит. Вони включають як традиційні методи масштабування, так і більш складні технології, засновані на використанні алгоритмів машинного навчання для прогнозування навантаження.

Аналіз таких рішень дозволяє оцінити їх ефективність і точність у контексті реальних умов роботи стрімінгових платформ. Програмні інструменти, що підтримують автоматичне масштабування та інтеграцію з інфраструктурою, можуть суттєво знижувати навантаження на операторів і дозволяти здійснювати реальний моніторинг і прогнозування. Однак деякі з них мають обмеження, пов'язані з масштабованістю або точністю прогнозів, особливо коли йдеться про обробку великих обсягів даних або адаптацію до швидко змінюваних умов. Розгляд цих рішень дозволяє виявити найбільш підходящі підходи та технології для оптимізації процесів управління ресурсами в умовах постійно зростаючого попиту на стрімінгові послуги.

На основі проведених аналізів сформульована наступна постановка проблеми: для забезпечення стабільної роботи стрімінгових сервісів в умовах змінного навантаження необхідно розробити ефективний метод вибору та

масштабування апаратних ресурсів. Сучасні стрімінгові платформи, зокрема ті, що використовують Ant Media Server, потребують високоефективних рішень для адаптації інфраструктури до динамічних змін у попиті на ресурси, щоб уникнути перевантажень і забезпечити якісне обслуговування користувачів.

Існуючі методи управління ресурсами, зокрема на основі традиційних алгоритмів масштабування або ручних налаштувань, не завжди здатні оперативно і точно реагувати на коливання навантаження. Для цього необхідно впровадити технології машинного навчання, які дозволяють прогнозувати майбутнє навантаження на систему та автоматично адаптувати кількість і потужність серверів, що використовуються для обробки стрімінгових даних.

Проблема полягає в необхідності розробити метод, який би забезпечував точне прогнозування навантаження, ефективно управляв апаратними ресурсами і дозволяв знизити витрати на інфраструктуру при збереженні високої якості обслуговування користувачів. Для цього потрібно розробити систему, яка б поєднувала моніторинг, прогнозування на основі великих обсягів даних та автоматичне масштабування ресурсів в режимі реального часу.

Висновки до розділу 1

1. Стрімінгові сервіси є складними системами, що потребують ефективного управління апаратними ресурсами для забезпечення безперебійної роботи та високої якості обслуговування користувачів, особливо в умовах змінного навантаження.
2. Сучасні технології машинного навчання надають можливості для автоматизації вибору апаратних ресурсів, що дозволяє точно прогнозувати навантаження на стрімінгові платформи та адаптувати їх інфраструктуру відповідно до змінюваного попиту.
3. Алгоритми машинного навчання, такі як ансамблеві методи, нейронні мережі та методи регресії, показують різний рівень ефективності у прогнозуванні

навантаження на сервери та визначенні оптимальних ресурсів для підтримки стабільної роботи стрімінгових сервісів.

4. Автоматизація процесу вибору та масштабування апаратних ресурсів за допомогою алгоритмів машинного навчання дозволяє значно підвищити ефективність управління інфраструктурою, зменшуючи витрати на інфраструктуру та покращуючи якість обслуговування користувачів.

5. Інтеграція машинного навчання для прогнозування навантаження та автоматизація масштабування ресурсів відкриває нові можливості для створення гнучких і ефективних рішень, що можуть адаптуватися до змінних умов попиту та забезпечувати безперебійну роботу стрімінгових платформ.

2 РЕЗУЛЬТАТИ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ У МЕЖАХ НАУКОВОЇ ЗАДАЧІ

2.1 Аналіз потреб в апаратних ресурсах для стрімінгових сервісів

В умовах постійного зростання популярності стрімінгових сервісів і збільшення обсягів відеоконтенту, основною задачею є забезпечення ефективного управління апаратними ресурсами платформи. Це включає вибір оптимальної конфігурації апаратних компонентів, таких як процесорна потужність, обсяг оперативної пам’яті та пропускна здатність мережі. Водночас важливо враховувати хмарні витрати, які значно впливають на загальну вартість інфраструктури, особливо за умов динамічних змін навантаження протягом доби або в періоди пікової активності користувачів.

Рисунок 2.1 показує основні компоненти системи потокового відео та взаємодії між ними [27]. Клієнти запитують відеоконтент у постачальника послуг (SP). SP відповідає за управління сервісом стрімінгового відео. Відеоконтент доставляється клієнтам за допомогою колекції серверів контенту (CS). Однією з основних функцій SP є вибір серверу контенту, коли клієнт ініціює сесію стрімінгу відео.

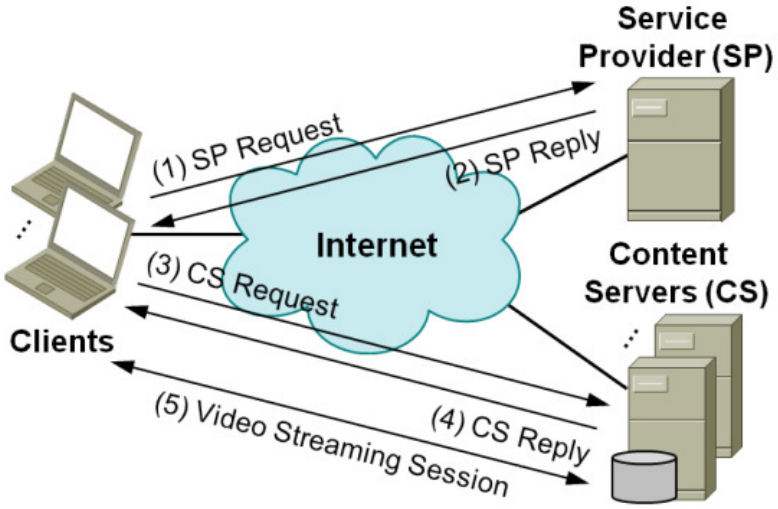


Рисунок 2.1 – Основні компоненти системи потокового відео.

Сесія починається з таких взаємодій:

1. Клієнт надсилає запит до постачальника послуг (SP), вказуючи бажаний відеоконтент. У запиті можуть також передаватися додаткові параметри, такі як ідентифікатор клієнта, пріоритети чи характеристики запитуваного контенту (наприклад, якість відео).

2. SP виконує алгоритм вибору сервера на основі доступної інформації (топології мережі, завантаженості серверів, доступної пропускної здатності шляхів тощо). У відповідь SP може надати клієнту:

– впорядкований список серверів, які можуть доставити відеоконтент (зазвичай сервери із найбільшою доступною пропускною здатністю або найближчі до клієнта);

– повідомлення про неможливість виконання запиту (якщо ресурси системи перевантажені або потрібний контент недоступний).

3. Клієнт запитує відеоконтент у одного із серверів, рекомендованих у списку, наданому SP. Запит зазвичай включає вибраний сервер і параметри сеансу.

4. Клієнт може вибрати сервер у списку кількома способами:

– підключитися до першого сервера у списку, рекомендованого SP;

– виконати свій власний алгоритм вибору, з використанням додаткової інформації (1) поточна затримка з'єднання до серверів; (2) завантаженість серверів на момент підключення; (3) інші метрики продуктивності (наприклад, час відгуку чи пропускна здатність).

5. Розпочинається сесія потокового відео між клієнтом і вибраним сервером. Сервер доставляє відеоконтент у режимі реального часу через оптимальний шлях у мережі. Якщо виникають проблеми (наприклад, затримки або перевантаження сервера), клієнт може переключитися на інший сервер зі списку (якщо він доступний) або звернутися до SP для нового списку серверів.

Якщо сервіс реалізовано за допомогою динамічного адаптивного стрімінгу через HTTP (MPEG-DASH) [28], постачальник послуг відповідає файлом Опису медіа-презентації (MPD), який містить інформацію про кодування відеоконтенту

(включаючи доступні роздільні здатності та бітрейт), а також URL-адреси серверів, які можуть доставити цей контент.

Рисунок 2.2 надає більше деталей щодо інфраструктури, використаної для доставки відеоконтенту. Ми припускаємо доставку "over-the-top" за допомогою відео-стрімінгу через HTTP або подібну технологію [28]. Тому не потрібна спеціальна підтримка з боку основної транспортної мережі (наприклад, резервування пропускної здатності).

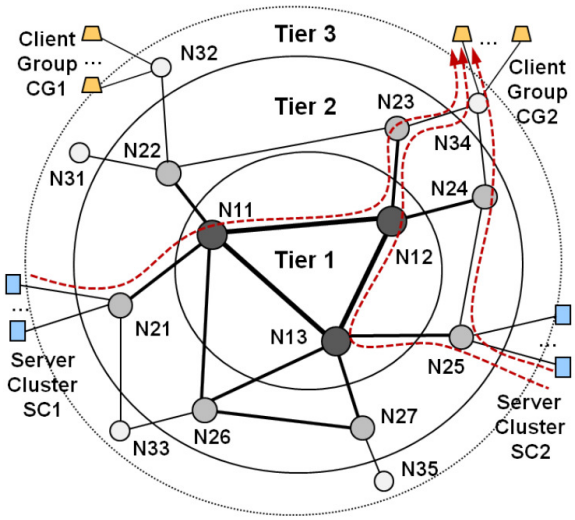


Рисунок 2.2 – Мережева інфраструктура

Сервери контенту зазвичай функціонують у межах Мережі доставки контенту (CDN) [29]. Вважається, що відеоконтент зберігається на колекції серверів, розгорнутих по всьому Інтернету, причому кожен відеофайл дублюється на кількох серверах. Передбачається, що CDN ефективно адаптується до потоків відеотрафіку, забезпечуючи оптимальне розташування серверів та реплікацію контенту. Постачальники послуг відеостримінгу можуть використовувати як власні CDN, так і послуги спеціалізованих постачальників CDN [30, 31]. Крім того, вони можуть володіти власною транспортною мережею або отримувати підтримку від постачальників мережевих послуг.

Вважається, що основними вузькими місцями транспортної мережі є лінії зв’язку між автономними системами (AS) постачальників мережевих послуг. У

зв’язку з цим мережа моделюється як набір взаємопов’язаних вузлів, які формують топологію багаторангових автономних систем, подібну до структури Інтернету (див. рисунок 2.1). Клієнти та сервери контенту підключаються до вузлів цієї мережі.

Протокол прикордонного шлюзу (BGP), що забезпечує маршрутизацію між AS в Інтернеті, визначає оптимальні шляхи між автономними системами, орієнтуючись на політики постачальників мережевих послуг і довжину маршруту [32]. Для балансування навантаження між кількома шляхами можуть використовуватися накладні додатки на рівні прикладного рівня або розширення маршрутизаційного протоколу. Це дозволяє більш ефективно розподіляти мережеві ресурси для сесій відеостримінгу [33]. Ця можливість розглядається як додатковий варіант у деяких алгоритмах вибору серверів.

Також, однією з проблем є необхідність масштабованості системи в режимі реального часу. Це актуально для багатокористувацьких сервісів, таких як Ant Media Server, який підтримує інтерактивні трансляції та низку сучасних потокових протоколів, включаючи WebRTC, HLS та DASH. Масштабування має бути достатньо швидким і точним, щоб уникнути недостатнього або надлишкового використання ресурсів. Недостатність ресурсів може спричинити буферизацію, погіршення якості стріму та незадоволення користувачів, тоді як надмірне виділення ресурсів призводить до непотрібних витрат.

Сучасні підходи до управління ресурсами в стрімінгових сервісах усе частіше включають методи машинного навчання. Вони дозволяють аналізувати історичні дані про навантаження, передбачати майбутні пікові періоди та формувати рекомендації щодо оптимального використання апаратних ресурсів. Наприклад, алгоритми можуть прогнозувати збільшення кількості користувачів під час певних подій і заздалегідь готувати систему до навантаження, що забезпечить стабільну роботу платформи та мінімізує витрати.

Використання віртуалізації обладнання дозволяє CSP запускати кілька віртуальних машин (VM) на одному фізичному сервері (PM). Однак навантаження на кожну VM змінюється з часом, що може призвести до

перевантаження PM, коли попит на ресурси перевищує доступну ємність. Такий дисбаланс негативно впливає на продуктивність усіх VM (а отже, і застосунків), що працюють на сервері. Недостатнє забезпечення ресурсів також порушує угоду про рівень обслуговування (Service Level Agreement, SLA) [34].

SLA — це договір між клієнтом і CSP, який гарантує певний рівень продуктивності застосунків клієнта. Щоб дотримуватися SLA, постачальник послуг має уникати перевантаження PM і забезпечувати, щоб VM отримували потрібні їм ресурси. Як показано на рисунку 2.3, модель прогнозування отримує історичні дані від менеджера ресурсів і надсилає йому передбачуване навантаження для кожного завдання. Це дозволяє менеджеру ефективно розподіляти VM між PM відповідно до прогнозованих результатів.

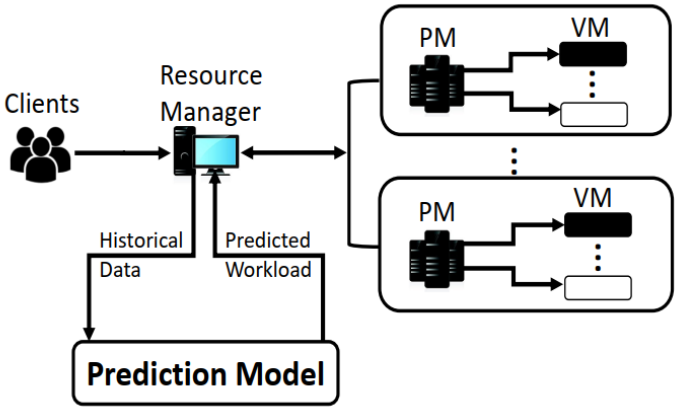


Рисунок 2.3 – Огляд процедури прогнозування робочого навантаження

Проблеми з ресурсами в хмарних дата-центрах. Часто хмарні дата-центри працюють із перевищенням допустимого завантаження. Ресурси, такі як ЦП та пропускна здатність, розподіляються між багатьма орендарями, що може призвести до конкуренції за ресурси між VM із високими потребами, що погіршує продуктивність застосунків.

Для вирішення цих проблем були розроблені стратегії управління ресурсами, зокрема для початкового розподілу VM та їхньої міграції. Останнім часом з’явилися методи прогнозування попиту на ресурси VM, які дозволяють забезпечувати ресурси заздалегідь або балансувати навантаження [35]. У рамках

проактивного балансування навантаження сервери прогнозують можливе перевантаження та заздалегідь переміщують VM.

Для кращого розуміння підходів до прогнозування навантажень можна порівняти репрезентативні методи на основі даних Google Cluster Trace [36]. У попередніх методах прогнозування використовувався підхід із нульовим часовим проміжком між точками вхідних даних про навантаження та прогнозованими значеннями (цей підхід називається 0-gap прогнозування). Однак такий підхід може залишати недостатньо часу для планування завдань на основі прогнозованого навантаження.

Запропоновано метод m-gap прогнозування, який зберігає проміжок у m часових точок між вхідними даними та прогнозованим значенням. Це забезпечує достатньо часу для планування завдань, не знижуючи точності прогнозування порівняно з 0-gap прогнозуванням.

Крім того, традиційні методи прогнозування створюють одну загальну модель для всіх завдань. Такий підхід не завжди може вловити специфічні закономірності для різнорідних завдань, що знижує точність прогнозів.

Для підвищення точності прогнозів запропоновано кластерний метод прогнозування, який класифікує завдання зі схожими патернами навантаження в групи для навчання. Для кожної групи створюється окрема модель прогнозування, яка потім використовується для прогнозування навантаження конкретного завдання, враховуючи його унікальні характеристики.

2.2 Розробка методу вибору апаратних ресурсів для стрімінгових сервісів

Основна мета цього дослідження — розробка методу вибору оптимальних апаратних ресурсів для стрімінгових сервісів, які працюють на основі Ant Media Server, із застосуванням машинного навчання.

Завданням дослідження є розв’язання кількох ключових аспектів:

- аналіз критеріїв вибору апаратних ресурсів;

- оптимізація розподілу навантаження;
- розробка підходу до інтеграції машинного навчання;
- адаптація підходу до умов стрімінгових сервісів;
- розробка моделі багатокритеріального прийняття рішень;
- експериментальна перевірка розробленого методу.

Першим завданням є визначення критеріїв, які впливають на вибір серверів і апаратних компонентів. До таких критеріїв належать продуктивність серверів, пропускна здатність мережі, енергоспоживання, вартість ресурсів та рівень затримок передачі даних. Врахування цих факторів є важливим для забезпечення ефективної роботи стрімінгових платформ.

Наступним етапом є оптимізація розподілу навантаження між серверами. Це завдання вимагає побудови прогнозних моделей, які здатні враховувати змінність навантаження у реальному часі. Оптимальний розподіл ресурсів дозволить забезпечити стабільну роботу систем навіть за умов пікового навантаження.

Для досягнення високої точності у виборі апаратних ресурсів дослідження орієнтоване на інтеграцію алгоритмів машинного навчання. Зокрема, це стосується розробки моделей прогнозування на основі сучасних методів, таких як LSTM або ARIMA, кластеризації серверних запитів і створення рекомендаційних систем для вибору серверів. Такий підхід дозволить автоматизувати процес прийняття рішень та враховувати специфіку різних типів навантаження.

Особливу увагу приділено адаптації запропонованого методу до умов роботи стрімінгових сервісів, зокрема Ant Media Server. Це передбачає врахування таких аспектів, як підтримка потокового передавання в реальному часі, мінімізація затримок та забезпечення якості обслуговування (QoS). Інтеграція запропонованого рішення в Ant Media Server дозволить значно покращити його ефективність.

Ще одним важливим завданням є розробка моделі багатокритеріального прийняття рішень. Поєднання методів машинного навчання з

багатокритеріальними підходами (MCDA) дозволить враховувати різноманітні параметри та їх вплив на кінцевий вибір. Це забезпечить збалансованість між продуктивністю системи, витратами та якістю обслуговування.

Фінальним етапом дослідження є експериментальна перевірка розробленого методу. Для цього передбачається створення прототипу системи, що дозволить провести тестування в умовах реального навантаження. Такий підхід забезпечить оцінку ефективності запропонованого рішення, а також дозволить визначити його вплив на якість роботи стрімінгових сервісів.

Результати дослідження дозволять створити адаптивний механізм управління ресурсами, який забезпечить стабільність роботи стрімінгових платформ, зменшення витрат на інфраструктуру та покращення користувацького досвіду. Це має важливе практичне значення для операторів цифрових платформ, які прагнуть надавати високоякісні послуги в умовах динамічно змінюваних вимог ринку.

2.3 Аналіз методів, та існуючого методологічного забезпечення

У даному розділі проведено аналіз методів, моделей і методик, що застосовуються для вирішення задач вибору серверів у системах доставки контенту. Розглянуто багатокритеріальні алгоритми прийняття рішень, процес вибору серверів, контроль допуску, а також статистичні й машинні методи прогнозування навантаження.

2.3.1 Багатокритеріальні алгоритми прийняття рішень

Вибір сервера і шляху в системах доставки контенту можна сформулювати як задачу оптимізації з кількома критеріями. Оскільки ці задачі є NP-складними (загалом), вони вирішуються на практиці за допомогою евристичних методів, таких як Мультікритеріальні алгоритми прийняття рішень (MCDA) [27]. Варіант MCDA, що використовується в цій статті, можна узагальнити наступним чином:

- Приймаючий рішення (постачальник послуг) визначає набір кандидатних рішень (простір рішень), $S = \{S_i\}_{i \in 1,\dots,n}$ на основі інформації про запит послуги та систему доставки контенту. Кандидатне рішення є вектором змінних рішень, $S_i = (v_{i,j})_{j \in 1,\dots,m}$ що представляють характеристики сервера і шляху, які можуть доставити запитуваний контент. Для кількох шляхів між сервером і клієнтом кожна пара, що складається із сервера та шляху, вважається окремим рішенням.

- Для кожного кандидатного рішення S_i приймаючий рішення обчислює ранг кожної змінної рішення $v_{i,j}$, $R_{i,j} = \frac{r_{i,j} - v_{i,j}}{r_{i,j} - a_{i,j}}$, де $v_{i,j}$ — поточне значення змінної, а— $r_{i,j}$ і $a_{i,j}$ - рівні резервування та прагнення для цієї змінної відповідно.

- Для кожного рішення S_i приймаючий рішення обчислює ранг рішення $R_i = \min_j R_{i,j}$ і потім визначає індекс найкращого рішення, . $\operatorname{argmin}_i R_i$.

Тут використовуються дві змінні рішення: доступну (невикористану) ємність сервера і доступну ємність (невикористану пропускну здатність) шляху. Для цих змінних рівень резервування є нижньою межею для відповідних рішень, а рівень прагнення є верхньою межею, за якою перевага рішень подібна. Як рівень прагнення встановлюється максимальна ємність сервера і шляху (дозволена для цієї послуги), а як рівень резервування — кількість ємності сервера та пропускної здатності шляху, що споживаються однією сесією, відповідно.

2.3.2 Процес вибору серверів

Використовуючи описаний алгоритм, постачальник відповідає на запит клієнта, вказуючи єдиний сервер [27]. Розглядається більш загальний підхід, за якого постачальник відповідає впорядкованим списком серверів. Клієнт може розпочати сесію, підключившись до першого сервера у списку, а інші використовувати як резерв. Крім того, надаючи список серверів, постачальник

дозволяє клієнту уточнити початковий вибір (наприклад, виконавши власний алгоритм вибору з використанням додаткової інформації) і/або застосовувати певні адаптивні техніки стримінгу (наприклад, перемикання серверів під час сесії).

Процес вибору складається з таких етапів:

- Постачальник послуг підтримує базу даних ресурсів із інформацією про мережу та сервери контенту, необхідну для вибору сервера. База даних містить статичну інформацію та частину динамічної інформації, наведеної в таблиці 2.1, залежно від використовуваного алгоритму.

Таблиця 2.1 – Інформація, що використовується для вибору серверів контенту

Статична або квазістатична інформація	**Динамічна інформація**
Списки серверів і відеофайлів	Контроль доступу до серверів.
Відображення відеофайлів на сервери	Доступна ємність серверів.
Відображення клієнтів і серверів на мережеві вузли	Контроль доступу до шляхів між серверами та клієнтами.
Довжина шляхів між серверами і клієнтами	Доступна ємність мережевих шляхів між серверами та клієнтами.

- Коли надходить запит на послугу, постачальник визначає впорядкований список відповідних серверів і надає його клієнту. Вибір базується на інформації, отриманій у запиті на послугу (наприклад, ідентифікатори клієнта та відео), а також на інформації з бази даних ресурсів. Кандидатні рішення складаються із сервера, який має копію запитуваного відеофайлу, і маршруту від цього сервера до клієнта. Алгоритм вибору отримує як вхідний параметр набір кандидатних рішень, які (за потреби) відповідають додатковим вимогам. Алгоритми, що використовуються в наших симуляційних експериментах, наведено в таблиці 2.2.

- Якщо постачальник не знаходить відповідний сервер (наприклад, відео недоступне або система перевантажена), запит відхиляється (тобто список серверів порожній).
- Якщо запит прийнято, клієнт розпочинає відеосесію, підключаючись до першого сервера в списку. У разі невдачі клієнт може спробувати підключитися до наступних серверів, не звертаючись повторно до постачальника для отримання рекомендацій щодо іншого сервера.

Таблиця 2.2 – Алгоритми вибору серверів

Алгоритм	**Вхідні дані**	**Вихідні дані (список серверів)**
Random servers	Набір допустимих рішень.	Випадковий сервер і найкоротший або випадковий шлях клієнт-сервер.
Closest servers	Набір допустимих рішень. Довжина шляхів від серверів до клієнта.	Сервери з найкоротшими шляхами до клієнта.
Best servers	Набір допустимих рішень. Доступна ємність серверів.	Сервери з найбільшою доступною ємністю і найкоротшим або випадковим шляхом.
MCDA	Набір допустимих рішень. Доступна ємність серверів і шляхів.	Сервери в рішеннях з найвищим рейтингом MCDA.

2.3.3 Контроль допуску

Процес вибору сервера має враховувати дві основні цілі: ефективно розподіляти ресурси системи та забезпечувати виконання вимог до продуктивності потокового відео. Безперервне відтворення відео потребує своєчасної доставки даних, а отже, визначення нижньої межі швидкості передачі даних.

Важливо уникати перевантаження ресурсів системи, впровадивши певну форму контролю допуску до процесу вибору сервера. Це можна досягти шляхом

виключення рішень із перевантаженими серверами або шляхами зі списку кандидатів, що подається на вхід алгоритмів вибору.

Контроль допуску простіше реалізувати для серверів контенту. Їхню доступну ємність можна оцінити за допомогою локального додатку моніторингу, який передає цю інформацію провайдерам або клієнтам. Перевантажений сервер може відхиляти нові запити клієнтів, що забезпечує базовий метод контролю допуску, незалежний від процесу вибору сервера.

З іншого боку, складніше визначити, чи може шлях від сервера до клієнта обробляти додаткові сеанси. Для цього провайдерам необхідна інфраструктура моніторингу мережі, яка вимірює пропускну здатність між певними вузлами мережі.

Процес вибору сервера можна розширити, додавши опціональний контроль допуску таким чином:

– знайти кандидатів, що включають сервер із копією запитуваного відеофайлу та шлях від цього сервера до клієнта;

– визначити допустимі рішення, виключивши з кандидатів ті, чиї сервери або шляхи перевантажені;

– сформувати список рекомендованих рішень, застосувавши алгоритм вибору до множини допустимих рішень;

– надати клієнту список серверів або відхилити запит, якщо список порожній.

2.3.4 Прогнозування навантаження

Методи прогнозування навантаження поділяються на три групи: статистичні підходи, методи машинного навчання та методи глибокого навчання. Для кожної категорії представлено найбільш ефективний метод на основі його продуктивності:

1. Auto Regression Integrated Moving Average (ARIMA). ARIMA є поширеним інструментом для аналізу часових рядів і прогнозування майбутніх значень на основі історичних даних [37]. Цей метод використовується для роботи

з нестабільними наборами даних шляхом трансформації їх через процес різницювання. Це дозволяє стабілізувати часоряд і підвищити точність прогнозу. ARIMA досягає середньої точності прогнозу понад 70%, що робить його ефективним представником статистичних підходів.

2. Байєсівська регресія хребта (Bayesian Ridge Regression, BRR). BRR ефективний при роботі з патологічними даними, які можуть мати великі відхилення від фактичних значень [38]. Метод базується на ймовірнісній моделі, яка враховує варіативність прогнозу та зменшує похибки за рахунок додавання деякого рівня упередження (bias). Це підвищує точність прогнозів навіть для складних наборів даних.

3. Регресія підтримувальних векторів (Support Vector Regression, SVR). SVR побудована на основі лінійної регресії у багатовимірному просторі [39]. Особливістю цього методу є те, що він приділяє більшу увагу точкам даних, які знаходяться далі від попередніх прогнозованих значень. Це дозволяє моделі краще знаходити закономірності у складних наборах даних. SVR часто використовується для нелінійних регресійних задач, забезпечуючи високу точність прогнозів.

4. Довготривала короткочасна пам'ять (Long Short-Term Memory, LSTM). LSTM — це модель нейронної мережі, яка використовується для аналізу часових рядів і забезпечує високу точність прогнозування [40]. Особливістю LSTM є здатність ігнорувати патологічні дані за допомогою "воріт забуття" (forget gate) і "воріт введення" (input gate). Це дозволяє ефективно працювати з нестабільними даними, забезпечуючи адаптивність та точність прогнозів.

2.4 Обґрунтування вибору методів, методик та інструментів дослідження

2.4.1 Моделювання процесів вибору серверів

Для моделювання процесів вибору серверів використано симуляційне програмне забезпечення, написане на мові C++ із базовим використанням

дискретного симулятора подій OMNeT++ (рисунок 2.4) [41]. Модуль OMNeT++ відповідає за інтеграцію всієї системи в симулятор OMNeT++, запуск експериментів моделювання та збереження результатів [42, 43]. Використання симулятора забезпечило зручність інтеграції ресурсів системи, алгоритмів та можливість збору детальної статистики під час моделювання.

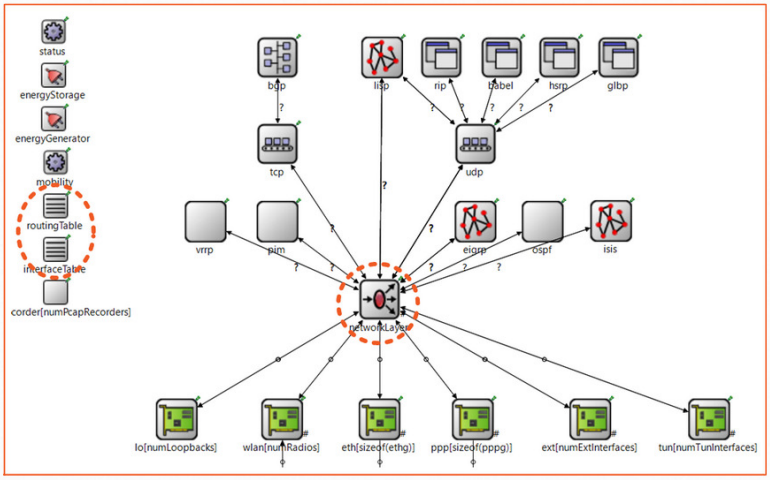


Рисунок 2.4 – Приклад роботи симулятора

На рисунку 2.5 показані основні компоненти симулятора та взаємодія між ними. Клас ResourceManager надає базу даних ресурсів із інформацією про всю інфраструктуру системи: топологію мережі (включаючи мережеві вузли, канали зв’язку, їхню пропускну здатність і навантаження), мережеві шляхи, список серверів контенту (зокрема їхню ємність і навантаження), відображення відеоконтенту на сервери, а також відображення клієнтів і серверів на вузли мережі.

Клас RequestProcess генерує запити на обслуговування, керує сеансами потокового відео та збирає відповідну статистику. Коли сеанс починається або закінчується, RequestProcess повідомляє ResourceManager, який відповідно оновлює навантаження на сервері та на каналах зв’язку на шляху між сервером і клієнтом.

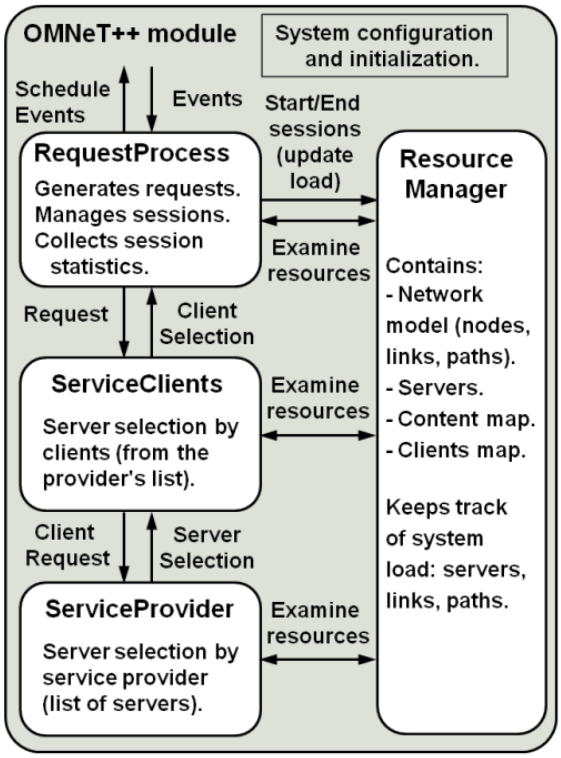


Рисунок 2.5 – Високорівнева структура симулятора

Процес вибору сервера реалізований класами ServiceProvider та ServiceClients. Клас ServiceProvider опрацьовує запити клієнтів на обслуговування та реалізує алгоритми вибору серверів, які виконує провайдер. Клас ServiceClients реалізує алгоритми, які використовують клієнти для вибору сервера зі списку, наданого провайдером, для потокового відеосеансу.

2.4.1.1 Модель мережі

Модель мережі використовує випадкові топології, згенеровані за допомогою інструмента aSHIIP [44]. Вузли мережі представляють автономні системи (AS) провайдерів мережевих послуг (див. рисунок 2.2). Така високорівнева топологія підвищує масштабованість моделі системи потокового відео і є достатньою для наших цілей, припускаючи, що основними вузькими місцями є між-AS з’єднання.

Мережеві топології мають багаторівневу структуру, схожу на Інтернет. Ми припускаємо, що ця структура відповідає звичайним бізнес-відносинам між провайдерами мережевих послуг: пірингові зв’язки між вузлами одного рівня та

відносини клієнт-провайдер між вузлами різних рівнів, де клієнт розташований на нижчому рівні. Програмне забезпечення для моделювання надає інструменти для обчислення кількох найкращих шляхів для будь-якої пари вузлів у порядку метрики шляху, починаючи з найкоротшого.

Шляхи, використані в моделюванні, схожі на ті, що визначаються протоколом BGP відповідно до відносин між провайдерами мережі («без долин») [6]. Клієнти та сервери контенту в системі потокового відео підключаються до вузлів мережі. Сервери розгортаються по всій мережі у «добре підключених» вузлах мережі, «близько до клієнтів», враховуючи кількість з'єднань і їхню пропускну здатність.

2.4.1.2 Ресурси системи

Ресурси, які споживаються під час сеансів потокового відео, вимірюються за допомогою метрики ємності серверів та метрики пропускної здатності каналів зв'язку. Загалом, сеанси можуть споживати різну кількість ресурсів залежно від кодування відео. Для спрощення аналізу припускаємо, що один сеанс вимагає 1 одиницю ємності сервера та 1 одиницю пропускної здатності каналу для забезпечення необхідної якості досвіду (QoE).

Система потокового відео забезпечується шляхом призначення фіксованої максимальної ємності для серверів контенту та каналів зв'язку. Ресурси серверів і мережі зазвичай спільно використовуються з іншими додатками чи послугами. Припускаємо, що призначена ємність відповідає ресурсам, які сервіс потокового відео може використовувати протягом експерименту моделювання.

Сервери мають однакову ємність і розгортаються у кластерах; вузли з кращими з'єднаннями мають більші кластери. Мережа забезпечується шляхом розподілу пропускної здатності каналів відповідно до ієрархічної структури мережі, причому у верхніх рівнях передбачено більшу пропускну здатність. Ємність вузлів (автономних систем) не обмежується, оскільки вузькими місцями є з'єднання між ними.

Також припускаємо, що відеофайли реплікуються залежно від їхньої популярності, щоб максимально збільшити кількість запитів, які сервери можуть обробляти. Це означає, що при повному навантаженні частка загальної ємності сервера, що використовується для певного відео, приблизно дорівнює ймовірності його запиту.

2.4.1.3 Метрики продуктивності

Основною метрикою продуктивності, що використовується для аналізу процесу вибору сервера, є коефіцієнт успішності: відношення кількості успішних запитів до загальної кількості запитів. Запит вважається успішним, якщо система має достатньо серверних і мережевих ресурсів для доставки відеоконтенту клієнту з потрібною якістю протягом усього часу його тривалості.

Збої можуть виникати як на етапі вибору сервера, так і під час сеансу потокового відео. Якщо доступний механізм контролю доступу, запит відхиляється у випадку, коли в процесі вибору не вдається знайти сервер і шлях із достатнім невикористаним ресурсом; у такому разі навантаження на систему залишається незмінним. В іншому випадку створюється новий сеанс, який може перевантажити сервер і/або деякі канали зв’язку на шляху від сервера до клієнта.

Процес RequestProcess відстежує поточні сеанси й оновлює їхній стан. Якщо до цього моменту сеанс був успішним, але більше не має достатніх ресурсів, його стан змінюється з "успішний" на "неуспішний".

2.4.2 Методи прогнозування

Прогнозування без розриву (0-gap Prediction)

Цей підхід базується на використанні w точок даних перед n-ю точкою часу для прогнозування значення Vn у тій самій точці часу. Розмір вікна w визначає кількість вхідних даних. Розподіл даних базується на схемі де перші 80% даних використовуються для навчання, решта 20% — для тестування. Особливістю є те, що вхідні дані безпосередньо пов’язані з прогнозованим моментом часу, що дозволяє оцінювати точність прогнозу.

На рисунку 2.6 зображено приклад процедури навчання та тестування для прогнозування без розриву (0-gap prediction). Числа у квадратах позначають послідовність часу. Наприклад, у процесі тестування червоні квадрати представляють вхідні дані, а їх кількість відповідає розміру вікна (w). У вікні розміру w =3 три точки часу (1-3) використовуються як вхідні дані для прогнозування значення на 4-му кроці. Результат прогнозу порівнюється з реальним значенням, щоб обчислити точність.

У 0-gap методі передбачуване значення Vn може бути відоме лише після реального виконання Vn, що залишає мало часу для планування. Для вирішення цієї проблеми запропоновано m-gap метод, де використовуються w точок даних перед точкою $n - m$ для прогнозування значення Vn. Перші 60% часу використовуються для навчання, а решта 40% — для тестування. Особливістю є те, що прогноз виконується з певним зазором (gap), що дозволяє більш ефективно використовувати результати для планування.

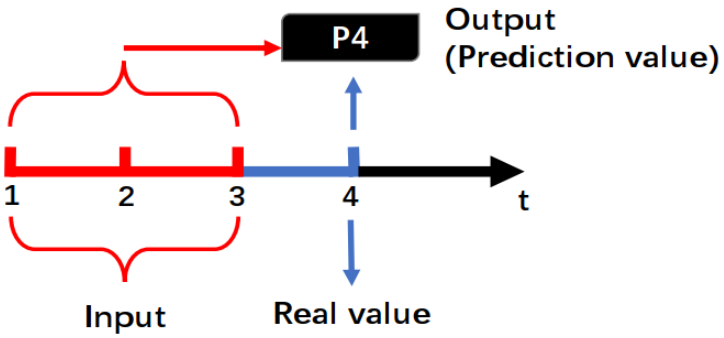


Рисунок 2.6 – Приклад процедуру навчання та тестування для прогнозування без розриву (0-gap prediction)

На рисунку 2.7 зображено процедуру навчання та тестування для прогнозування із часовим розривом (mm-gap prediction). На відміну від прогнозування без розриву (0-gap), у методі з часовим розривом між останньою точкою часу вхідних даних та точкою часу прогнозованого значення існує інтервал mm, який позначено сірими квадратами. При $m = 3$ значення на точках

часу 2, 3, і 4 використовуються для прогнозу значення на точці 8 (із зазором у 3 кроки). Це створює більше часу для прийняття рішень до настання реального навантаження.

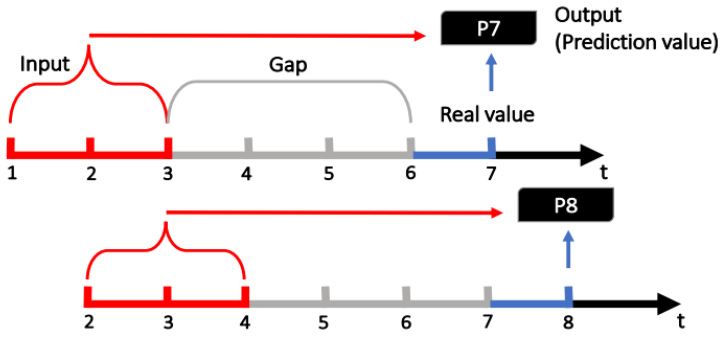


Рисунок 2.7 – Приклад процедури навчання та тестування для прогнозування із часовим розривом (mm-gap prediction).

Для оцінки продуктивності описаних методів використовуються три основні метрики:

1. Кумулятивна функція розподілу (CDF) точності: Для аналізу точності прогнозування використовується CDF. Точність прогнозування обчислюється за формулою:

$$A_n = 1 - \frac{|P_n - R_n|}{R_n}$$

де An — точність для n-го прогнозу,

Pn— прогнозоване значення,

Rn— реальне значення.

2. CDF точності з різними розмірами вікна: Щоб оцінити вплив розміру вікна (w) на точність, використовується CDF для моделі, яка показала найкращі результати.

2.4.3 Методи прогнозування на основі кластеризації

У попередньому розділі було створено одну загальну модель для прогнозування навантаження всіх завдань. Однак, враховуючи, що різні завдання мають різні характеристики навантаження, виникла потреба перевірити, чи можна покращити точність прогнозування шляхом побудови окремих моделей для груп схожих завдань. Для цього спочатку виконується кластеризація завдань за їх схожістю, після чого для кожного кластеру створюється окрема модель прогнозування, використовуючи однаковий алгоритм машинного навчання. Кожне завдання асоціюється з відповідною моделлю своєї категорії для виконання прогнозування навантаження. Для кластеризації завдань у групи було обрано два методи, які описані нижче:

Метод кластеризації на основі прототипів (Prototype-based Clustering Method, PBC) полягає в знаходженні найкоротшої відстані між кожним завданням і центром кластера. Кількість кластерів *N* означає, що завдання (які описуються даними про використання пам’яті) будуть поділені на *N* частин, і для кожної частини відстань між кожним описом завдання (або точкою даних) і центром буде найменшою. Існує два основні методи для PBC — це K-середніх (K-means) [45] та кластеризація за допомогою гауссівських змішаних моделей (Gaussian Mixture Clustering) [46].

K-середніх (K-means) — це ітеративний алгоритм, заснований на відстані. Він кластеризує всі спостереження в набір з *N* кластерів так, щоб кожне спостереження знаходилося ближче до центру свого кластеру, ніж до центрів інших кластерів. Алгоритм використовує ітеративну оптимізацію для мінімізації квадратичної помилки та наближення до цільової функції.

Gaussian Mixture Clustering використовує ймовірнісну модель для виразу прототипу кластеру. Однак завдання можуть мати різну довжину, тоді як методи PBC вимагають однакової довжини для всіх точок даних. Для вирішення цієї проблеми ми додаємо нулі в кінець завдання, щоб зробити всі завдання однакової довжини.

Експерименти показали, що метод K-середніх дає кращі результати, ніж Gaussian Mixture Clustering, тому для прогнозувальних методів буде використовуватись метод K-середніх.

Вхідними даними для методу PBC є всі завдання з набору даних. Завдання кластеризуються в різні підмножини. Для кожної підмножини ми використовуємо один з трьох алгоритмів прогнозування (ARIMA, BRR, LSTM), щоб побудувати модель. У результаті кількість підмножин N дає N моделей. Збільшення N призводить до більших обчислювальних витрат, але зменшення N знижує точність прогнозування, оскільки менша кількість підмножин може не захопити всі особливості навантаження. За замовчуванням ми використовуємо N=5, оскільки це забезпечує кращий баланс між точністю прогнозування та обчислювальними витратами згідно з нашими експериментами.

Метод кластеризації на основі щільності (Density-based Clustering Method, DBC) [47] є непараметричним методом кластеризації, що орієнтується на щільність. Він спочатку знаходить точки в районах з високою щільністю, а потім об'єднує близькі точки в один кластер. Водночас для викидів, які знаходяться в низькощільних областях, кожну таку точку призначають до найближчого кластеру. Наприкінці всі точки кластеризуються.

Однією з основних відмінностей між K-середніх та DBC є те, що в K-середніх кількість кластерів NNN задається вручну, тоді як у DBC кількість кластерів визначається самим методом на основі щільності. В даному випадку DBC кластеризує всі завдання в п’ять груп.

Оскільки DBC виявляє кластери шляхом безперервного з'єднання точок високої щільності в околицях, для його роботи потрібно лише визначити розмір околиці та поріг щільності. Завдяки цьому DBC здатний знаходити кластери різних форм і розмірів, що є його перевагою перед іншими методами кластеризації.

Використовуємо DBC в наших методах прогнозування. Вхідними даними для DBC є всі завдання з набору даних. Завдання кластеризуються в різні підмножини, і для кожної підмножини застосовуються попередньо описані

алгоритми прогнозування (ARIMA, BRR, LSTM). За допомогою цього методу отримуємо кластеризовані завдання зі схожими патернами, що дозволяє тренувати кожну підмножину окремо для покращення точності прогнозу.

2.5 Розробка методів та засобів реалізації поставлених завдань

За допомогою OMNeT++ проведено Симуляційний аналіз продуктивності алгоритмів, наведених у таблиці 2.3, з різними налаштуваннями для алгоритмів вибору, контролю доступу, управління сеансами, топології мережі, її розміру, розміщення серверів тощо.

Експерименти проводилися на мережі з 1000 вузлами, структурованими в 4 рівні. Алгоритми вибору можуть обирати серед 3 попередньо обчислених найкоротших шляхів для будь-якої пари вузлів (додаткові шляхи не значно покращують продуктивність).

Таблиця 2.3 – Конфігурації використані для вибору серверів

Назва	**Алгоритм вибору серверів та шляхів**	**Контроль доступу**
RandRand	Випадкові сервери, випадкові шляхи	Відмова серверів
RandShrt	Випадкові сервери, найкоротші шляхи	Відмова серверів
NearSrv	Сервери з найкоротшими шляхами до клієнта	Відмова серверів
BestRand	Сервери з найбільшою доступною потужністю, випадкові шляхи	АС для серверів
BestShrt	Сервери з найбільшою доступною потужністю, найкоротші шляхи	АС для серверів
MCDA	MCDA для доступної потужності серверів і шляхів	АС для серверів і шляхів

Загальна потужність серверів становить близько 61500 сеансів. Запити на сервіс генеруються випадковими інтервалами часу з експоненційним розподілом. Середня тривалість сеансу — 400 секунд. Мережа налаштована таким чином, що кількість одночасних сеансів може досягати загальної потужності серверів для ефективного вибору сервера. Ми вимірюємо коефіцієнт успішності сеансів як функцію швидкості запитів.

Таблиця 2.4 надає список конфігурацій вибору серверів, що використовувалися в цих експериментах. Ці конфігурації відповідають ряду можливостей постачальників послуг і природних комбінацій алгоритмів та налаштувань контролю доступу.

Таблиця 2.4 – Вплив вибору випадкових шляхів у експерименті з 5 шляхами

Шлях	**Метрика доступної потужності**	**Кількість переходів**	**Вузли**	**Рівні**
45->18	41	100	45:5:18	2:1:2
45->18	-342	150	45:2:1:5:18	2:1:1:2
45->18	-250	175	45:2:0:1:5:18	2:1:1:1:2
45->18	-180	175	45:2:0:6:5:18	2:1:1:1:1:2
45->18	-250	175	45:2:3:1:5:18	2:1:1:1:1:2

Постачальники послуг, які мають можливість моніторингу серверів та мереж, можуть використовувати MCDA (Мультикритеріальний аналіз рішень) з контролем доступу для серверів і шляхів. Ця конфігурація забезпечує найефективніше використання системних ресурсів. Коефіцієнт успішності залишається рівним 1 для швидкостей запитів до 150 запитів на секунду та близько 60000 одночасних сесій, що наближається до загальної потужності серверів. За межами цього порогу коефіцієнт успішності починає знижуватись, оскільки система повністю завантажена, і деякі запити відхиляються.

Постачальники послуг, які мають лише статичну інформацію про систему (див. таблиця 2.4), можуть вибирати випадкові сервери або найближчі сервери без контролю доступу (наприклад, конфігурації RandRand, RandShrt, NearSrv). Коефіцієнт успішності різко знижується, починаючи з нижчих швидкостей запитів, оскільки сесії можуть зазнати збоїв через перевантажені мережеві лінії або відхилення запитів через повністю завантажені сервери.

Рисунки 2.8 та 2.9 показують коефіцієнт успішності для початкового розміщення серверів та двох варіантів управління сеансами.

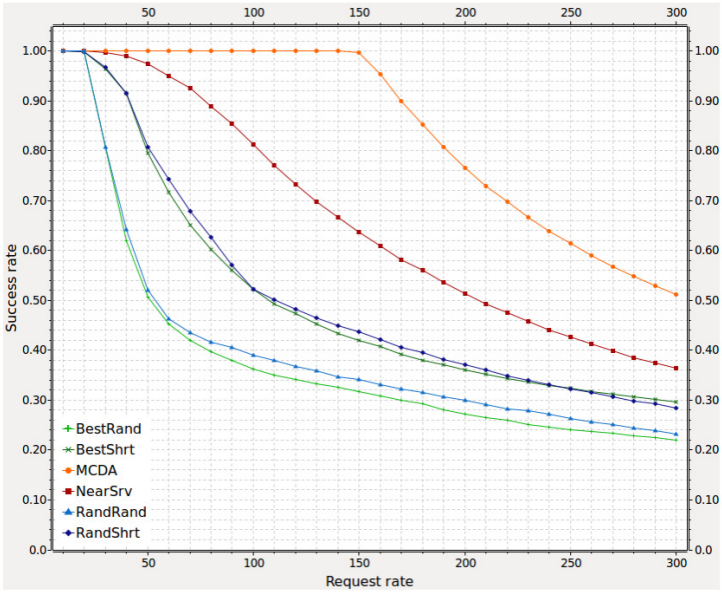


Рисунок 2.8 - Початкове розташування серверів із завершенням сесії

Вибір найменш завантаженого сервера (BestRand, BestShrt) не покращує продуктивність у цих експериментах, оскільки відеофайли реплікуються відповідно до їхньої популярності, і тому випадковий вибір сервера добре розподіляє навантаження між серверами.

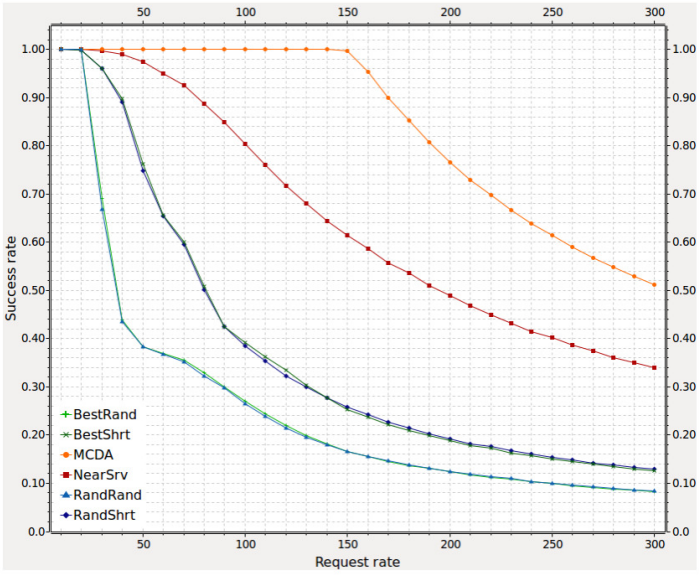


Рисунок 2.9 - Початкове розташування серверів без завершення сесії

Перевантажені сесії можуть бути припинені клієнтами або серверами. Звільнені ресурси можуть бути використані для запуску інших сесій, які можуть

бути успішними, наприклад, через те, що їхні шляхи не включають перевантажені лінії. Рисунки 2.8 та 2.9 показують коефіцієнт успішності в двох крайніх випадках: на рисунку 2.8 перевантажені сесії припиняються, а на рисунку 2.9 сесії продовжуються (знижена якість обслуговування, наприклад, знижена роздільна здатність і швидкість передачі даних, якщо доступна адаптація швидкості). Припинення перевантажених сесій значно покращує продуктивність для випадково вибраних серверів і/або шляхів (RandRand, RandShrt, BestRand), тоді як NearSrv меншою мірою постраждав, оскільки сесії завжди використовують найближчий сервер та найкоротший шлях.

Інтуїтивно, продуктивність повинна покращуватись, коли навантаження розподіляється на кілька мережевих шляхів. Однак спостерігаємо, що використання найкоротшого шляху дає кращу продуктивність, ніж випадковий вибір з 3-5 кращих шляхів (наприклад, RandShrt проти RandRand на рисунку 2.8). Це пов'язано з ієрархічною структурою мережі та побудовою шляхів. У багаторівневій мережі з безвальними шляхами найкоротший шлях може уникати ядра мережі, тоді як інші шляхи, ймовірно, перетинають його Таким чином, випадковий вибір шляху концентрує трафік у ядрі мережі. Таблиця 2.4 показує приклад експерименту з 5 шляхами: найкоротший шлях від вузла 45 до вузла 18 не перетинає ядро, і його доступна потужність складає 41, тоді як інші шляхи перетинають ядро (лінії першого рівня) і є сильно перевантаженими. Тому, коли доступно кілька шляхів, алгоритм вибору має враховувати навантаження на шлях (наприклад, використовуючи MCDA).

Для подібних причин продуктивність покращується, якщо постачальник послуг вибирає сервери з найкоротшими шляхами до клієнта (NearSrv), замість випадкового вибору серверів (наприклад, RandRand, RandShrt): сесії використовують менше мережевих ресурсів, і шляхи мають більшу ймовірність уникати ядра мережі. Таким чином, значне покращення продуктивності можна досягти, використовуючи лише квазістатичну інформацію.

Результати прогнозування без розриву (0-gap prediction) оцінюють ефективність методу, який здійснює прогнозування без часових розривів. На

рисунку 2.10 демонструється кумулятивна функція розподілу (CDF) точності прогнозування для використання процесора серед чотирьох порівнюваних методів: SVR, ARIMA, LSTM та BRR (Додаток А). Результати показали наступний порядок точності: SVR < ARIMA ≈ LSTM < BRR.

SVR (Support Vector Regression) показав гірші результати порівняно з іншими методами. Оскільки розмір Google Cluster Trace великий, SVR не зміг досягти високої точності прогнозування, як зазначено в [48]. Це пов'язано з тим, що SVR погано справляється з великими і складними наборами даних.

ARIMA та LSTM показали кращі результати, ніж SVR, але їх точність була все ще гіршою, ніж у BRR. ARIMA створює лінійну модель прогнозування, що не є оптимальним для обробки патологічних даних, таких як пропущені значення або нестабільні точки, які зустрічаються в Google Cluster Trace. Цей недолік обмежує її ефективність у таких випадках.

LSTM (Long Short-Term Memory) може працювати з нелінійними даними, що дозволяє йому підлаштовуватися під складніші закономірності. Однак для короткострокових задач, де є обмежена кількість точок для навчання, LSTM не показує гарних результатів, як зазначено в [49]. Через це його ефективність для таких задач є гіршою порівняно з BRR.

BRR (Bayesian Ridge Regression) виявився найбільш точним методом серед усіх. Завдяки використанню алгоритму Левенберга-Марквардта [50], який працює з нелінійними наборами даних, BRR зміг досягти найкращих результатів навіть за наявності патологічних даних або обмежених точок для навчання.

На рисунках 2.10 та 2.11 зображено кумулятивну функцію розподілу (CDF) точності прогнозування для використання пам’яті при різних розмірах вікна (w=1, w=10, w=50). Результати показали наступне співвідношення точності:

- Розмір вікна = 1 показав найгіршу точність.
- Розмір вікна = 10 і розмір вікна = 50 продемонстрували схожі результати з високою точністю.

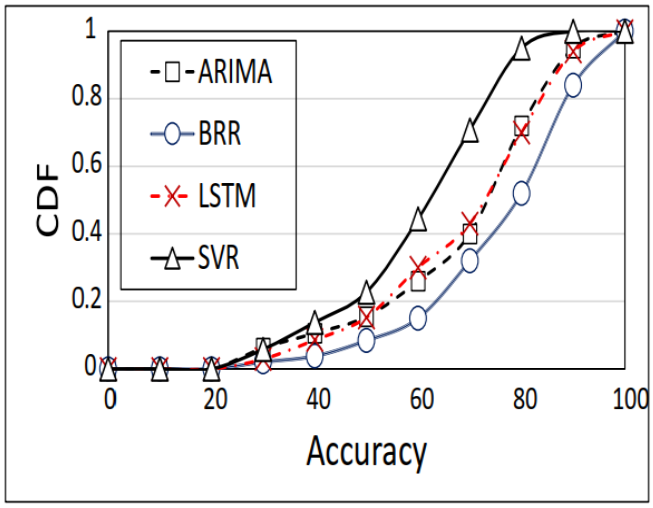
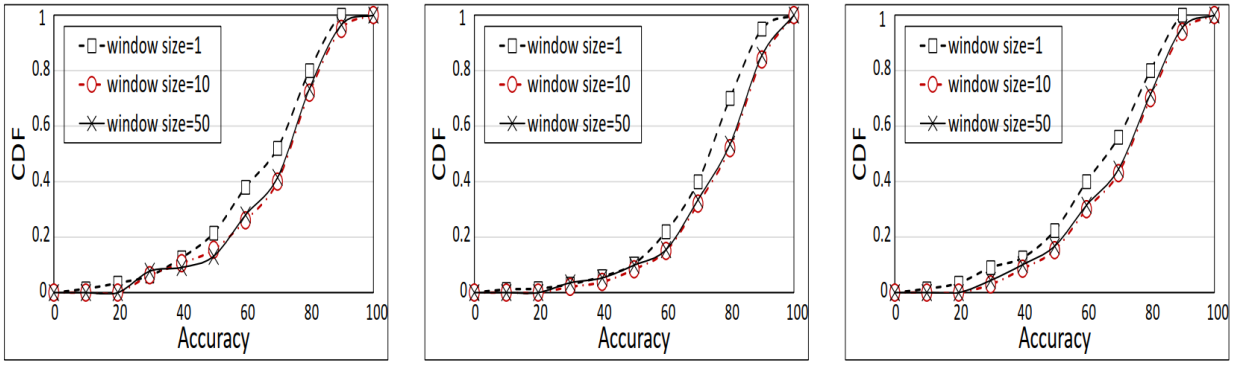


Рисунок 2.10 – Кумулятивна функція розподілу (CDF) точності



(a) ARIMA (b) BRR (c) LSTM

Рисунок 2.11 – Кумулятивна функція розподілу (CDF) точності процесора з різними розмірами вікна для прогнозування без розриву (0-gap)

Збільшення розміру вікна дозволяє вводити більше вхідних значень у модель, що покращує точність прогнозу. Однак більший розмір вікна також призводить до значного збільшення часу на навчання та тестування моделі. Оскільки розмір вікна w=10 має подібну точність прогнозування, але менший витрат часу, ми вибираємо цей параметр як основний, якщо не зазначено інше.

Подібно до рисунків 2.10 і 2.11, рисунки 2.12 і 2.13 показують результати прогнозування пам'яті, де спостерігається та сама тенденція і порядок для всіх порівнюваних методів та різних розмірів вікна через ті ж самі причини.

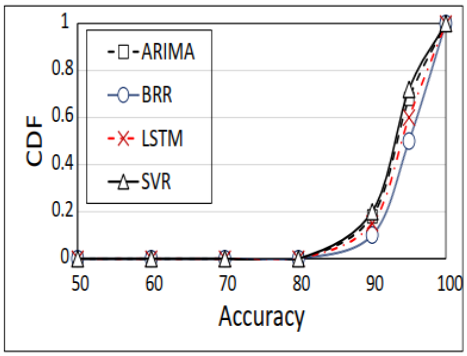
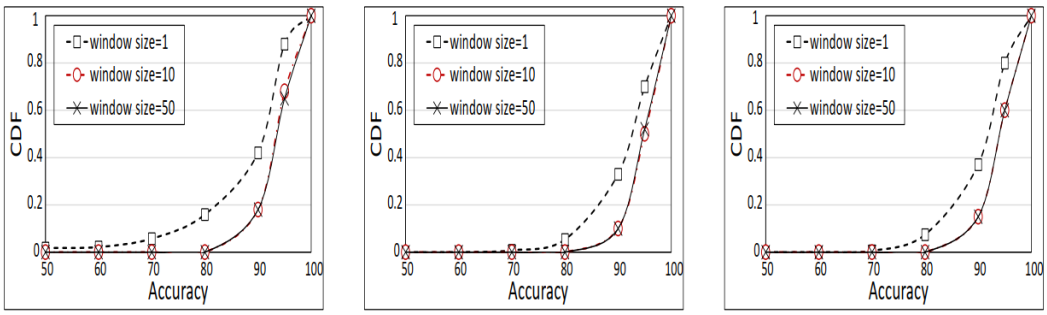


Рисунок 2.12 – Кумулятивна функція розподілу (CDF) точності пам'яті для прогнозування без розриву (0-gap)



(a) ARIMA (b) BRR (c) LSTM

Рисунок 2.13 – Кумулятивна функція розподілу (CDF) точності пам'яті з різними розмірами вікна для прогнозування без розриву (0-gap)

На рисунках 2.14 і 2.15 зображено кумулятивну функцію розподілу (CDF) точності прогнозування для використання пам’яті при прогнозуванні з часовим розривом (mmm-gap prediction). Як і в методі прогнозування без розриву (0-gap), результати показали такий самий порядок точності: ARIMA ≈ LSTM < BRR.

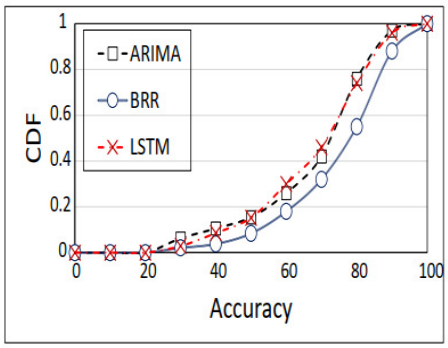
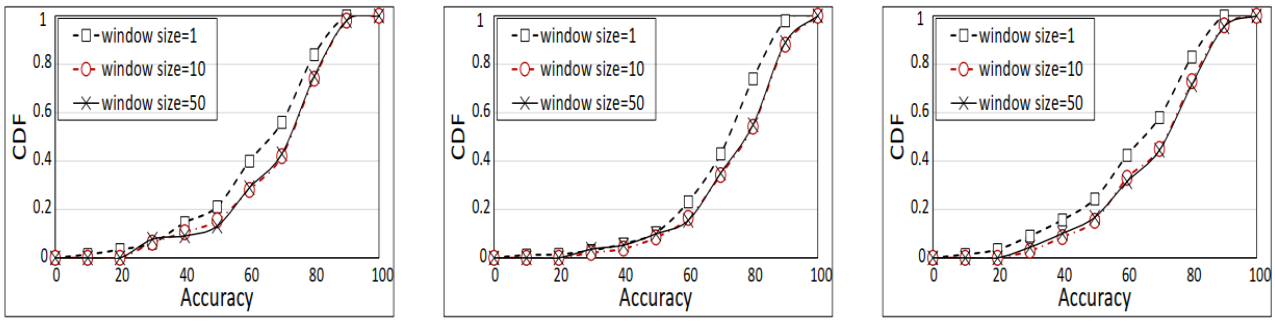


Рисунок 2.14 – Кумулятивна функція розподілу (CDF) точності процесора для прогнозування з розривом (m-gap)



(a) ARIMA (b) BRR (c) LSTM

Рисунок 2.15 – Кумулятивна функція розподілу (CDF) для різних розмірів вікна в точності процесора для прогнозування з розривом (m-gap)

- ARIMA не справляється добре з обробкою патологічних даних, що є причиною погіршення її результатів. Як зазначено раніше, ARIMA створює лінійне прогнозування, що обмежує її ефективність при наявності нестабільних або пропущених даних.
- LSTM здатний добре працювати з нелінійними даними завдяки своїй нейронній мережі. Однак для короткострокових задач, де є недостатньо точок для навчання, його продуктивність виявляється гіршою, ніж у BRR.
- BRR показує найвищу точність, оскільки цей метод може працювати з патологічними наборами даних і обмеженими точками для навчання, завдяки використанню алгоритму Левенберга-Марквардта, який добре підходить для нелінійних даних.

Також проводилась оцінка ефективності методів прогнозування, заснованих на PBC та DBC. Оскільки різниця в точності прогнозування між методами 0-gap (без часової паузи) та m-gap (із часовою паузою) незначна, у цих експериментах застосовувався метод 0-gap. Результати показали, що кластеризація задач перед прогнозуванням суттєво підвищує точність оцінки робочого навантаження.

Прогнозування на основі PBC: рисунок 2.16 показує кумулятивну функцію розподілу (CDF) точності прогнозування для використання CPU за допомогою методів прогнозування на основі PBC. Результати мають такий порядок: PBC-

ARIMA ≈ PBC-LSTM < PBC-BRR. Щодо PBC-ARIMA, як зазначалося раніше, ARIMA не може досягти високої точності, коли дані нестабільні. Оскільки патологічні дані випадково зустрічаються серед усіх задач [28], навіть після кластеризації PBC, яка групує подібні задачі для моделювання, продуктивність ARIMA залишається низькою та схожою на PBC-LSTM.

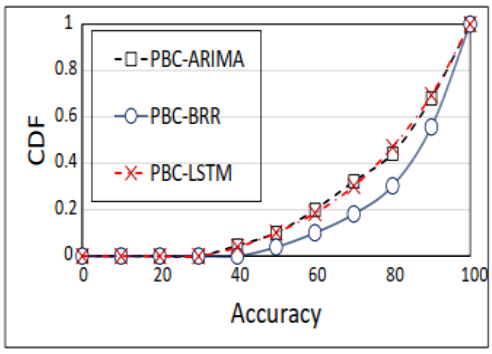
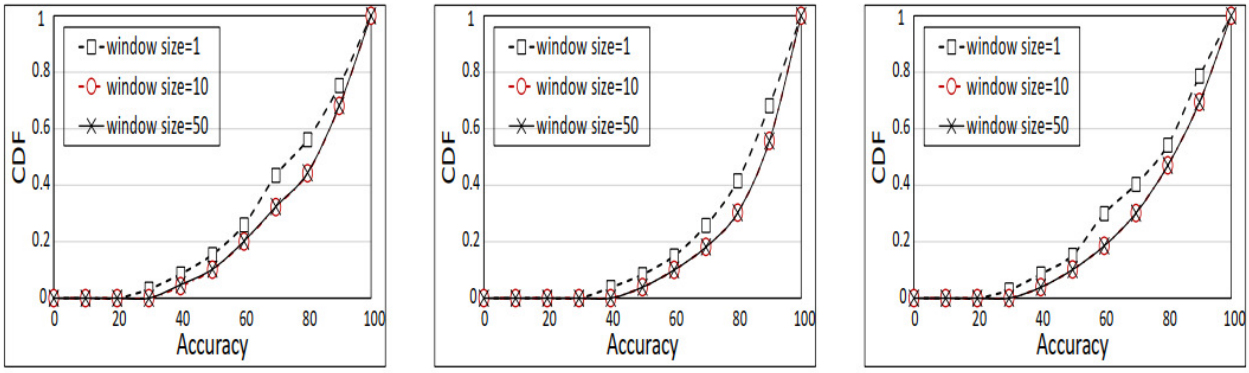


Рисунок 2.16 – Кумулятивна функція розподілу (CDF) точності процесора для прогнозування на основі PBC.

Перевага PBC-LSTM полягає у прогнозуванні робочого навантаження для довгострокових задач (з великою кількістю точок даних для навчання) завдяки використанню нейронної мережі LSTM. Однак метод менш ефективний для короткострокових задач із малою кількістю точок даних для тренування. Навіть після PBC PBC-LSTM демонструє нижчу продуктивність, ніж PBC-BRR.

PBC-BRR здатний працювати з нелінійними наборами даних і малими обсягами тренувальних даних. Після PBC PBC-BRR досягає вищої точності прогнозування, ніж два інших методи. Таким чином, PBC-BRR має найкращу продуктивність серед трьох методів.

Рисунок 2.17 показує CDF точності прогнозування для використання CPU із різними розмірами вікон для всіх трьох методів. Результати мають ту саму тенденцію і порядок, що й на рисунку 2.11, з тих самих причин.



(a) PBC-ARIMA (b) PBC-BRR (c) PBC-LSTM

Рисунок 2.17 – Кумулятивна функція розподілу (CDF) точності процесора з різними розмірами вікна для прогнозування на основі PBC.

Ці графіки показують, що всі методи досягають подібної точності прогнозування з різними розмірами вікон. Причина цього полягає в тому, що після кластеризації всі методи можуть забезпечувати схожу точність навіть із меншим розміром вікна. Порівнюючи ці результати з Рисунком 7, можна помітити значне покращення точності при розмірі вікна 1. Тому PBC значно покращує точність прогнозування.

2.6 Результати дослідження методів

За результатами оцінки продуктивності методів кластеризації з трьома сучасними методами прогнозування у задачах 0-gap прогнозування, також включено для порівняння результати методів прогнозування з m-gap. Отримані результати відповідають такому порядку: 0-gap прогнозування ≈ m-gap прогнозування < PBC ≈ DBC, де покращення сягає від 75% до понад 90% у прогнозуванні використання CPU і від 92% до 95% у прогнозуванні використання пам’яті.

Метод ARIMA виявився менш ефективним для обробки патологічних даних, таких як пропущені значення або дані з різкими змінами трендів. Оскільки ARIMA використовує лінійне моделювання, він не справляється з нелінійними

залежностями, що обмежує його ефективність в умовах нестабільних чи аномальних даних. Це впливає на якість прогнозів, коли вхідні дані мають значні коливання або пропуски.

LSTM (Long Short-Term Memory), як метод на основі нейронних мереж, показав значну ефективність у роботі з нелінійними даними і здатний враховувати довгострокові залежності. Однак для короткострокових завдань, де є обмежена кількість точок для навчання, LSTM працює гірше, ніж BRR. Це пов'язано з тим, що нейронні мережі потребують великих обсягів даних для успішного навчання, і у випадку невеликої кількості тренувальних даних LSTM не може показати свої сильні сторони.

BRR (Bayesian Ridge Regression) продемонстрував найкращі результати в контексті прогнозування при обмеженій кількості точок для навчання, зокрема при наявності патологічних наборів даних. Завдяки використанню алгоритму Левенберга-Марквардта, який добре підходить для нелінійних даних, BRR зміг забезпечити високу точність навіть при малих обсягах даних. Цей метод виявився стійким до нестабільностей у даних і продемонстрував високу ефективність у багатьох сценаріях.

Метод PBC-LSTM, який комбінує попередню обробку даних через кластеризацію з LSTM, показав хороші результати при прогнозуванні для довгострокових задач, де є достатня кількість точок для навчання. Однак для короткострокових задач, де дані обмежені, цей метод виявився менш ефективним порівняно з PBC-BRR. Після застосування PBC, PBC-BRR показав найкращу продуктивність серед усіх методів завдяки своїй здатності працювати з малими обсягами тренувальних даних і нестабільними наборами даних.

Таким чином, PBC-BRR продемонстрував найкращу точність прогнозування в умовах обмежених даних та високої нестабільності, на відміну від інших методів, що підтверджує його ефективність для застосування у стрімінгових сервісах з динамічними навантаженнями.

У трейсах, де різні задачі мають значно відмінні шаблони, 0-gap і m-gap прогнозування не можуть досягти високої продуктивності, використовуючи

лише одну модель. PBC і DBC покращують продуктивність методів прогнозування, оскільки кластеризація групує задачі зі схожими шаблонами для навчання. Потім модель, що відповідає групі задач, може легко захопити ці шаблони та точніше прогнозувати продуктивність. Результати показують, що методи кластеризації можуть значно підвищити точність прогнозування, аж до 15% у порівнянні з попередніми методами.

Висновки до розділу 2

1. Описано ключові аспекти вибору та управління апаратними ресурсами для стрімінгових сервісів, зокрема на основі методів машинного навчання. Проведений аналіз різних методів і моделей, що використовуються для оптимізації ресурсів, включав багатокритеріальні алгоритми прийняття рішень, вибір серверів, прогнозування навантаження та адаптацію до змінних умов навантаження.

2. Аналіз потреб в апаратних ресурсах для стрімінгових сервісів показав, що ефективне управління інфраструктурою вимагає врахування багатьох факторів, зокрема, обчислювальної потужності серверів, пропускної здатності мережі та обсягу оперативної пам'яті. Зокрема, важливим є вибір оптимальної конфігурації ресурсів для забезпечення стабільної роботи стрімінгових сервісів під час високих навантажень. Для цього необхідно постійно моніторити й прогнозувати зміни в навантаженні для забезпечення безперервного й ефективного надання послуг.

3. Аналіз методів, моделей та методик, що використовуються для вирішення завдання вибору апаратних ресурсів, продемонстрував, що застосування багатокритеріальних алгоритмів прийняття рішень може бути ефективним для забезпечення балансу між вартістю та якістю. Багатокритеріальні методи допомагають врахувати різні аспекти, такі як економічні обмеження, технічні характеристики серверів і вимоги до швидкості

обробки даних. Для вибору серверів критично важливим є забезпечення максимальної продуктивності при мінімальних витратах на інфраструктуру, з урахуванням змінних умов експлуатації.

4. Процес вибору серверів та контроль допуску є важливими етапами в управлінні інфраструктурою стрімінгових сервісів. Вибір серверів повинен ґрунтуватися на детальному аналізі доступних потужностей, прогнозуванні пікових навантажень та забезпеченні можливості масштабування в залежності від реального попиту на ресурси. Контроль допуску забезпечує захист від перегрузок системи, гарантуючи її стабільну роботу.

5. Розробка методів та засобів для реалізації поставлених завдань підтвердила доцільність використання запропонованих методів, таких як багатокритеріальне прийняття рішень у поєднанні з прогнозуванням навантаження. Вони дозволяють адаптувати інфраструктуру стрімінгових сервісів до реальних потреб, знижуючи витрати на хмарні ресурси та забезпечуючи стабільну якість обслуговування користувачів.

6. Результати дослідження методів показали, що комбіновані методи, такі як PBC-BRR, демонструють найвищу точність і здатність працювати з нелінійними та обмеженими даними, що дозволяє їм бути ефективними в умовах постійно змінного навантаження. Це забезпечує оптимальне управління ресурсами стрімінгових сервісів при мінімальних витратах і високій продуктивності.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МЕТОДУ ВИБОРУ АПАРАТНИХ РЕСУРСІВ

3.1 Опис архітектури системи

У даному розділі детально описується архітектура системи вибору апаратних ресурсів для стрімінгових сервісів на базі Ant Media Server з використанням машинного навчання, що дозволяє автоматизувати процес прогнозування навантаження та адаптації інфраструктури. Архітектура складається з кількох основних компонентів, які працюють разом для забезпечення стабільної роботи сервісу, ефективного управління ресурсами та забезпечення високої якості стрімінгу при змінному навантаженні.

Відомі системи Hetzner пропонують на своїх серверах наступні конфігурації з відповідними цінами [51], що наведено на рисунку 3.1. Аналогічно, сервіси хмарних обчислень, такі як Microsoft Azure та Amazon Web Services (AWS), також надають різноманітні серверні конфігурації для обробки та зберігання даних з різними рівнями продуктивності та цінами. Детальніше інформацію про доступні конфігурації серверів представлено у файлі cloud_price.csv, що знаходиться в додатку Б.

Microsoft Azure пропонує кілька варіантів серверних конфігурацій з адаптивними планами оплати, що дозволяє користувачам вибирати ресурси залежно від потреб у пропускній здатності, обчислювальних потужностях та зберіганні даних. Ціни залежать від вибору обчислювальних ресурсів (VM, бази даних, хмарні сервіси), а також тривалості використання та масштабу обробки даних.

Amazon Web Services (AWS) має схожу модель пропозиції, надаючи користувачам доступ до широкого спектра серверних конфігурацій через EC2 (Elastic Compute Cloud), а також інших хмарних сервісів для зберігання даних (S3), обробки великих даних, а також аналізу і машинного навчання. Ціни в AWS залежатимуть від вибору конкретних інстанцій, таких як загального призначення або спеціалізовані інстанції для високих навантажень.

	VCPU	RAM	NVME SSD	TRAFFIC INCL.	IPV4	HOURLY	MONTHLY
CX22	2	4 GB	40 GB	20 TB	✓	€ 0.0071	€ 4.51 max.
CX32	4	8 GB	80 GB	20 TB	✓	€ 0.0134	€ 8.09 max.
CX42	8	16 GB	160 GB	20 TB	✓	€ 0.0325	€ 19.52 max.
CX52	16	32 GB	320 GB	20 TB	✓	€ 0.0643	€ 38.56 max.

Рисунок 3.1 – Цінові пропозиції та конфігурації серверів Hetzner.

Оптимальні апаратні ресурси для стрімінгових сервісів, що працюють на основі Ant Media Server, є важливим аспектом для забезпечення стабільної роботи та ефективного масштабування системи. Для досягнення цієї мети необхідно зібрати та проаналізувати дані про використання ресурсів серверів, що дозволить створити ефективні моделі прогнозування та рекомендації щодо масштабування. Одним із ключових етапів цього процесу є збір точних та актуальних даних про навантаження на ресурси, що забезпечує високу якість прогнозів і дозволить оптимізувати використання апаратних засобів.

3.2 Збір даних та побудова моделі

Для збору даних про використання ресурсів серверу було використано утиліту sar (System Activity Report). Ця утиліта дозволяє збирати та аналізувати статистику роботи системи, включаючи використання процесора (CPU), пам'яті (RAM), диска (Disk) та мережі (Network).

```
sar -u 5 > cpu_usage.txt & sar -r 5 > memory_usage.txt & sar -
d 5 > disk_usage.txt & sar -n DEV 5 > network_usage.txt &
```

Ця команда дозволяє збирати дані про використання процесора, пам'яті, диска та мережевих ресурсів кожні 5 секунд. Інформація про процесор

зберігається у файл cpu_usage.txt, про пам'ять — у файл memory_usage.txt, про диск — у файл disk_usage.txt, а дані про мережеве використання — у файл network_usage.txt. Це дає можливість здійснювати моніторинг системних ресурсів у реальному часі та зберігати їх для подальшої обробки і аналізу.

Після збору даних з сервера необхідно перетворити їх у зручний формат для подальшої обробки, наприклад, у формат CSV. Ось приклад Python коду, який виконує конвертацію текстових логів у CSV формат для зручнішого аналізу (Додаток В).

Для побудови надійної моделі необхідно поділити набір даних на дві частини: тренувальну (для навчання моделі) та тестову (для оцінки її загальної продуктивності). Процес розділення виконується за допомогою функції train_test_split з бібліотеки sklearn.model_selection, що дозволяє отримати пропорційну вибірку для подальшого аналізу.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.optimizers import Adam

df = pd.read_csv('output.csv', usecols=['CPU', 'RAM', 'Disk',
'Network'])

# Перетворюємо відсотки на числові значення
df['CPU'] = df['CPU'].astype(float)
df['RAM'] = df['RAM'].astype(float)
df['Disk'] = df['Disk'].astype(float)
```

```
df['Network'] = df['Network'].astype(float)

# Масштабування даних (для LSTM зазвичай використовують нормалізацію)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df)
```

LSTM добре працює з нормалізованими даними. Ми будемо використовувати MinMaxScaler для масштабування значень. MinMaxScaler здійснює масштабування кожного елемента вектора в межах певного діапазону, зазвичай [0, 1], що зменшує вплив великого значення деяких ознак на модель. Це також сприяє кращій конвергенції під час навчання нейронної мережі, оскільки всі ознаки мають однаковий масштаб і не переважатимуть одна одну.

Нормалізація є важливою, особливо для рекурентних нейронних мереж, таких як LSTM, оскільки великі значення можуть призвести до проблеми "затухаючого градієнта", яка ускладнює навчання. Тому використання MinMaxScaler є корисною практикою для стабільності та продуктивності LSTM-моделей.

```
# Функція для створення послідовностей даних для LSTM
def create_sequences(data, time_steps):
    X, y = [], []
    for i in range(len(data) - time_steps):
    X.append(data[i:i + time_steps])
    y.append(data[i + time_steps])
    return np.array(X), np.array(y)

# Вибираємо кількість часових кроків (time_steps)
time_steps = 5   # кількість попередніх спостережень для передбачення

# Створюємо послідовності для LSTM
X, y = create_sequences(scaled_data, time_steps)

# Розбиваємо на тренувальну та тестову вибірки
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

LSTM працює з часовими рядами, тому потрібно підготувати дані в формат, який буде зручний для моделі. Для цього потрібно створити вхідні та вихідні дані у вигляді послідовностей.

Модель складається з одного LSTM шару, що обробляє послідовності даних, та вихідного Dense шару для прогнозування чотирьох характеристик. Модель була навчена з використанням оптимізатора Adam і функції втрат середньоквадратичної похибки (MSE), що дозволило досягти високої точності прогнозів.

```
#Створення LSTM моделі
model = Sequential()

# Додаємо перший LSTM шар
model.add(LSTM(units=50,                return_sequences=False,
input_shape=(X_train.shape[1], X_train.shape[2])))

# Додаємо Dense шар для виведення
model.add(Dense(units=4))  # 4 характеристики (CPU, RAM, Disk,
Network)
model.compile(optimizer=Adam(), loss='mean_squared_error')

# Навчаємо модель
model.fit(X_train,    y_train,    epochs=20,    batch_size=32,
validation_data=(X_test, y_test))
```

Тепер можна використовувати модель для прогнозування майбутніх значень для тестових даних і порівняти ці прогнози з реальними значеннями.

```
# Передбачення для тестових даних
y_pred = model.predict(X_test)
# Зворотнє масштабування результатів для виведення
y_pred_rescaled = scaler.inverse_transform(y_pred)
y_test_rescaled = scaler.inverse_transform(y_test)
# Виведення результатів
```

```
print("Передбачення:", y_pred_rescaled[:5])     # перші 5
передбачених значень
print("Реальні значення:", y_test_rescaled[:5])   # перші 5
реальних значень
```

Для реалізації програми створення графіка для візуалізації порівняння реальних та передбачених значень можна використовувати мову програмування Python разом з бібліотеками для обробки даних та побудови графіків, такими як matplotlib та pandas.

```
features = ['CPU', 'RAM', 'Disk', 'Network']
for i, feature in enumerate(features):
     plt.figure(figsize=(10, 6))
     plt.plot(y_test_rescaled[:, i], label='Real values',
color='blue')
     plt.plot(y_pred_rescaled[:, i], label='Predicted values',
color='red', linestyle='--')
     plt.title(f'{feature} - Real vs Predicted', fontsize=14)
     plt.xlabel('Samples', fontsize=12)
     plt.ylabel(f'{feature} value', fontsize=12)
     plt.legend()
     plt.show()
```

3.3 Аналіз та оцінка результатів реалізації

Оцінка результатів прогнозування здійснюється за допомогою середньоквадратичної похибки (MSE, Mean Squared Error), яка вимірює середню квадратичну різницю між фактичними та прогнозованими значеннями. Чим менше значення MSE, тим точніші прогнози дає модель, що свідчить про її кращу ефективність.

MSE можна обчислити для кожної з характеристик, таких як процесор (CPU), оперативна пам'ять (RAM), диск (Disk) і мережа (Network). Це дозволяє

нам оцінити, наскільки добре модель прогнозує значення для кожного з параметрів.

```
# Обчислюємо MSE
for i, feature in enumerate(features):
    mse = mean_squared_error(y_test_rescaled[:, i], y_pred_rescaled[:, i])
    print(f'MSE for {feature}: {mse}')
```

Результат оцінки:

```
MSE for CPU: 0.19139648810259097
MSE for RAM: 0.008586408627735166
MSE for Disk: 0.016622315317366944
MSE for Network: 0.01722371895480092
```

Ці значення MSE знаходяться в діапазоні від 0 до 1. Чим ближче MSE до 0, тим менше похибки в прогнозах, що свідчить про хорошу точність моделі. Якщо цільові значення (наприклад, відсотки використання процесора, або використання оперативної пам'яті в гігабайтах) знаходяться в подібному діапазоні, то MSE можна вважати відносно низьким.

Зокрема, для таких характеристик, як оперативна пам'ять і процесор, значення MSE є досить низькими, що означає хорошу точність моделі для цих параметрів. Водночас, для мережі та диска значення MSE трохи вищі, але все одно не вказують на суттєві похибки.

Загалом, наведені значення MSE є відносно невеликими, що вказує на те, що модель працює добре, принаймні для даного діапазону значень. Однак, для досягнення ще кращих результатів можна досліджувати можливість покращення моделі, наприклад, шляхом використання інших методів навчання або налаштування параметрів моделі.

На рисунку 3.2 зображено хмарні конфігурації, де x – ціна на хмарні послуги, у це системні ресурси. Рекомендовані конфігурації для Ant Media Server, позначені червоним прямокутником для зручності ідентифікації. Вибір оптимальних параметрів базується на результатах тестування та порівняння продуктивності різних конфігурацій. Рекомендовані варіанти серверів

забезпечують баланс між високою продуктивністю та економічною ефективністю, що дозволяє досягти найкращих результатів при використанні Ant Media Server для потокової трансляції відео.

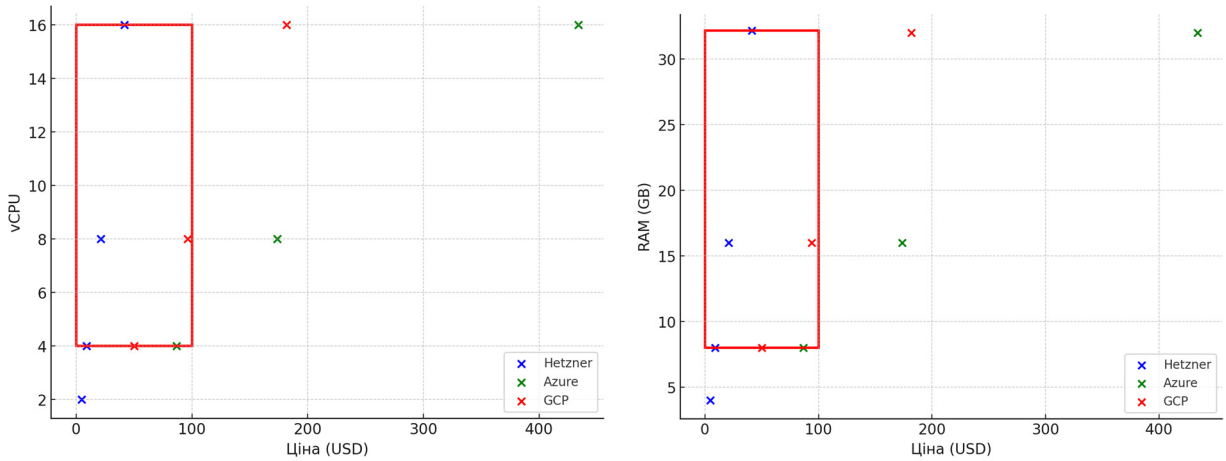


Рисунок 3.2 – Цінові пропозиції та конфігурації серверів.

Прогнозування та вибір відповідних серверних конфігурацій є важливим етапом у забезпеченні ефективної роботи будь-якої хмарної інфраструктури. Для цього необхідно враховувати не лише поточні вимоги до ресурсів, але й майбутнє масштабування системи, зокрема, зростання навантаження та обсягів даних. Використання правильних конфігурацій допомагає забезпечити стабільну та швидку обробку запитів, знижує ризики збоїв і перевантажень, а також оптимізує витрати на ресурси. Рекомендації щодо конфігурацій, надані в даному дослідженні, є результатом ретельного аналізу та тестування різних варіантів серверів, що дозволяє зробити обґрунтований вибір для використання у реальних умовах.

Висновки до розділу 3

1. Розглянуто структуру системи, яка включає основні компоненти, що взаємодіють для забезпечення стабільної роботи сервісу, ефективного

управління ресурсами та підтримки високої якості стрімінгу за змінного навантаження.

2. Описано процес збору даних і побудови моделі для прогнозування навантаження. Модель складається з LSTM шару для обробки послідовностей даних та Dense шару для прогнозування ключових характеристик. Також було зазначено використання оптимізатора Adam і функції втрат MSE для досягнення високої точності.

3. Використання мови програмування Python та таких бібліотек, як pandas, scikit-learn, matplotlib та TensorFlow, забезпечило зручність розробки, високу гнучкість у реалізації методів прогнозування навантаження та вибору апаратних ресурсів для стрімінгових сервісів. Завдяки цим інструментам було побудовано ефективну модель машинного навчання, яка дозволяє прогнозувати навантаження на систему та здійснювати оптимальний вибір апаратних ресурсів у реальному часі.

4. Експериментальне тестування моделі продемонструвало її стабільність у роботі та високу точність прогнозів. Зокрема, використання моделі LSTM для прогнозування навантаження дозволило досягти низьких значень середньоквадратичної помилки (MSE), що вказує на високу точність.

5. У процесі тестування моделі були отримані значення MSE, що знаходяться в діапазоні від 0 до 1, де значення, ближчі до 0, свідчать про високу точність прогнозів. Це підтверджує ефективність моделі в контексті вибору апаратних ресурсів і прогнозування навантаження на систем.

6. Аналіз результатів виявив, що точність прогнозів знижується при обробці даних з високою варіативністю або при прогнозуванні на довші періоди, що вказує на необхідність удосконалення алгоритмів для врахування додаткових факторів, таких як зміни в інфраструктурі або непередбачувані піки навантаження.

ВИСНОВКИ

1. Стрімінгові сервіси є складними системами, що потребують ефективного управління апаратними ресурсами для забезпечення безперебійної роботи та високої якості обслуговування користувачів, особливо в умовах змінного навантаження.

2. Сучасні технології машинного навчання надають можливості для автоматизації вибору апаратних ресурсів, що дозволяє точно прогнозувати навантаження на стрімінгові платформи та адаптувати їх інфраструктуру відповідно до змінюваного попиту.

3. Алгоритми машинного навчання, такі як ансамблеві методи, нейронні мережі та методи регресії, показують різний рівень ефективності у прогнозуванні навантаження на сервери та визначенні оптимальних ресурсів для підтримки стабільної роботи стрімінгових сервісів.

4. Автоматизація процесу вибору та масштабування апаратних ресурсів за допомогою алгоритмів машинного навчання дозволяє значно підвищити ефективність управління інфраструктурою, зменшуючи витрати на інфраструктуру та покращуючи якість обслуговування користувачів.

5. Інтеграція машинного навчання для прогнозування навантаження та автоматизація масштабування ресурсів відкриває нові можливості для створення гнучких і ефективних рішень, що можуть адаптуватися до змінних умов попиту та забезпечувати безперебійну роботу стрімінгових платформ.

6. Описано ключові аспекти вибору та управління апаратними ресурсами для стрімінгових сервісів, зокрема на основі методів машинного навчання. Проведений аналіз різних методів і моделей, що використовуються для оптимізації ресурсів, включав багатокритеріальні алгоритми прийняття рішень, вибір серверів, прогнозування навантаження та адаптацію до змінних умов навантаження.

7. Аналіз потреб в апаратних ресурсах для стрімінгових сервісів показав, що ефективне управління інфраструктурою вимагає врахування багатьох факторів, зокрема, обчислювальної потужності серверів, пропускної

здатності мережі та обсягу оперативної пам'яті. Зокрема, важливим є вибір оптимальної конфігурації ресурсів для забезпечення стабільної роботи стрімінгових сервісів під час високих навантажень. Для цього необхідно постійно моніторити й прогнозувати зміни в навантаженні для забезпечення безперервного й ефективного надання послуг.

8. Аналіз методів, моделей та методик, що використовуються для вирішення завдання вибору апаратних ресурсів, продемонстрував, що застосування багатокритеріальних алгоритмів прийняття рішень може бути ефективним для забезпечення балансу між вартістю та якістю. Багатокритеріальні методи допомагають врахувати різні аспекти, такі як економічні обмеження, технічні характеристики серверів і вимоги до швидкості обробки даних. Для вибору серверів критично важливим є забезпечення максимальної продуктивності при мінімальних витратах на інфраструктуру, з урахуванням змінних умов експлуатації.

9. Процес вибору серверів та контроль допуску є важливими етапами в управлінні інфраструктурою стрімінгових сервісів. Вибір серверів повинен ґрунтуватися на детальному аналізі доступних потужностей, прогнозуванні пікових навантажень та забезпеченні можливості масштабування в залежності від реального попиту на ресурси. Контроль допуску забезпечує захист від перегрузок системи, гарантуючи її стабільну роботу.

10. Розробка методів та засобів для реалізації поставлених завдань підтвердила доцільність використання запропонованих методів, таких як багатокритеріальне прийняття рішень у поєднанні з прогнозуванням навантаження. Вони дозволяють адаптувати інфраструктуру стрімінгових сервісів до реальних потреб, знижуючи витрати на хмарні ресурси та забезпечуючи стабільну якість обслуговування користувачів.

11. Результати дослідження методів показали, що комбіновані методи, такі як PBC-BRR, демонструють найвищу точність і здатність працювати з нелінійними та обмеженими даними, що дозволяє їм бути ефективними в умовах постійно змінного навантаження. Це забезпечує оптимальне управління

ресурсами стрімінгових сервісів при мінімальних витратах і високій продуктивності.

12. Розглянуто структуру системи, яка включає основні компоненти, що взаємодіють для забезпечення стабільної роботи сервісу, ефективного управління ресурсами та підтримки високої якості стрімінгу за змінного навантаження.

13. Описано процес збору даних і побудови моделі для прогнозування навантаження. Модель складається з LSTM шару для обробки послідовностей даних та Dense шару для прогнозування ключових характеристик. Також було зазначено використання оптимізатора Adam і функції втрат MSE для досягнення високої точності.

14. Використання мови програмування Python та таких бібліотек, як pandas, scikit-learn, matplotlib та TensorFlow, забезпечило зручність розробки, високу гнучкість у реалізації методів прогнозування навантаження та вибору апаратних ресурсів для стрімінгових сервісів.

15. Експериментальне тестування моделі продемонструвало її стабільність у роботі та високу точність прогнозів. Зокрема, використання моделі LSTM для прогнозування навантаження дозволило досягти низьких значень середньоквадратичної помилки (MSE), що вказує на високу точність.

16. У процесі тестування моделі були отримані значення MSE, що знаходяться в діапазоні від 0 до 1, де значення, ближчі до 0, свідчать про високу точність прогнозів. Це підтверджує ефективність моделі в контексті вибору апаратних ресурсів і прогнозування навантаження на систем.

17. Аналіз результатів виявив, що точність прогнозів знижується при обробці даних з високою варіативністю або при прогнозуванні на довші періоди, що вказує на необхідність удосконалення алгоритмів для врахування додаткових факторів, таких як зміни в інфраструктурі або непередбачувані піки навантаження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ant Media Server. URL: https://antmedia.io
2. YouTube. URL: https://www.youtube.com
3. Netflix. URL: https://www.netflix.com
4. Twitch. URL: https://www.twitch.tv
5. Facebook Live. URL: https://www.facebook.com/live/
6. Beben A., Batalla J. M., Chai W., Sliwinski J. Multi-criteria decision algorithms for efficient content delivery in content networks. Annals of Telecommunications, 2013, vol. 68, no. 3, pp. 153-165. Springer.
7. Kher, R., & Patel, M. (2019). Data imputation techniques in machine learning: A survey. Journal of Big Data.
8. Vaquero L., Rodero-Merino L., Caceres J., Lindner M. A break in the clouds: towards a cloud definition // Proc. of SIGCOMM. 2008.
9. Shen H., Chen L. Distributed autonomous virtual resource management in datacenters using finite-markov decision process // Trans. on TON. 2017.
10. Josep A., Katz R., Konwinski A., Gunho L., Patterson D., Rabkin A. A view of cloud computing // Communications of the ACM. 2010.
11. Khan A., Yan X., Tao S., Anerousis N. Workload characterization and prediction in the cloud: A multiple time series approach // Proc. of NOMS. 2012.
12. Jiang Y., Perng C., Li T., Chang R. Asap: A self-adaptive prediction system for instant cloud resource demand provisioning // Proc. of ICDM. 2011.
13. Morais A., Brasileiro V., Lopes V., Santos A., Satterfield W., Rosa L. Autoflex: Service agnostic auto-scaling framework for iaas deployment models // Proc. of CCGrid. 2013.
14. Gong Z., Gu X., Wilkes J. Press: Predictive elastic resource scaling for cloud systems // Proc. of ICNSM. 2010.
15. Imam T., Miskhat F., Rahman R., Amin A. Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources // Proc. of ICCIT. 2011.

16. Farahnakian F., Liljeberg P., Plosila J. Lircup: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers // Proc. of EuroSEAA. 2013.

17. Bankole A., Ajila S. Cloud client prediction models for cloud resource provisioning in a multitier web application environment // Proc. of SOSE. 2013.

18. Islam S., Keung J., Lee K., Liu A. Empirical prediction models for adaptive resource provisioning in the cloud // Trans. on FGCS. 2012.

19. Roy N., Dubey A., Gokhale A. Efficient autoscaling in the cloud using predictive models for workload forecasting // Proc. of ICC. 2011.

20. Nikravesh A., Ajila S., Lung C. Measuring prediction sensitivity of a cloud auto-scaling system // Proc. of CSAC. 2014.

21. Shyam G., Manvi S. Virtual resource prediction in cloud environment: a bayesian approach // Journal of Network and Computer Applications. 2016.

22. Qiu F., Zhang B., Guo J. A deep learning approach for vm workload prediction in the cloud // Proc. of SNPD. 2016.

23. Zhang W., Duan P., Yang L., Xia F., Li Z., Lu Q., Gong W., Yang S. Resource requests prediction in the cloud computing environment with a deep belief network // Software: Practice and Experience. 2017.

24. Zhang Q., Yang L., Yan Z., Chen Z., Li P. An efficient deep learning model to predict cloud workload for industry informatics // Trans. on TII. 2018.

25. Kumar J., Goomer R., Singh K. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters // Trans. on Computer Science. 2018.

26. Song B., Yu Y., Zhou Y., Wang Z., Du S. Host load prediction with long short-term memory in cloud computing // Journal of Supercomputing. 2018.

27. O. Catrina, "Multi-criteria Server Selection for Over-the-top Video Streaming," 2016 International Conference on Communications (ICComm), 2016, pp. 303–308, doi: 10.1109/ICComm.2016.7528293.

28. Sodagar I. The MPEG-DASH standard for multimedia streaming over the Internet. IEEE Multimedia, 2011, vol. 18, no. 4, pp. 62-67. IEEE.

29. Hofmann M., Beaumont L. Content Networking. Architecture, Protocols, and Practice. Morgan Kaufmann, 2005.

30. Adhikari V. K., Jain S., Chen Y., Zhang Z. Vivisecting YouTube: An Active Measurement Study. Technical Report TR 11-012, Department of Computer Science and Engineering, University of Minnesota, 2011.

31. Adhikari V. K., Guo Y., Hao F., Varvello M., Hilt V., Steiner M., Zhang Z.-L. Unreeling Netflix: Understanding and improving multi-CDN movie delivery. IEEE INFOCOM 2012, pp. 1620-1628. IEEE.

32. Gao L. On inferring autonomous system relationships in the Internet. IEEE/ACM Trans. on Networking, 2001, vol. 9, no. 6, pp. 733-745. IEEE.

33. He J., Rexford J. Toward Internet-wide multipath routing. IEEE Network, 2008, vol. 22, no. 2, pp. 16-21. IEEE.

34. Qiu C., Shen H., Chen L. Probabilistic demand allocation for cloud service brokerage // Proc. of INFOCOM. 2016.

35. Yu Y., Jindal F., Bastani V., Li F., Yen I. Improving the smartness of cloud management via machine learning based workload prediction // Proc. of COMPSAC. 2018.

36. Reiss C., Wilkes J., Hellerstein J. Google cluster-usage traces: format+ schema // Google Inc., White Paper. 2011.

37. Calheiros R., Masoumi E., Ranjan R., Buyya R. Workload prediction using arima model and its impact on cloud applications' qos // Trans. on CC. 2015.

38. Shyam G., Manvi S. Virtual resource prediction in cloud environment: a bayesian approach // Journal of Network and Computer Applications. 2016.

39. Chen L., Shen H., Sapra K. Rial: Resource intensity aware load balancing in clouds // Proc. of INFOCOM. 2014.

40. Song B., Yu Y., Zhou Y., Wang Z., Du S. Host load prediction with long short-term memory in cloud computing // Journal of Supercomputing. 2018.

41. Gao, Jiechao, Wang, Haoyu, Shen, Haiying. Machine Learning Based Workload Prediction in Cloud Computing [Text] / J. Gao, H. Wang, H. Shen //

Proceedings of the International Conference on Computer Communications and Networks (ICCCN). – 2020. – P. 1–9. – DOI: 10.1109/ICCCN49398.2020.9209730.

42. Wehrle K., Gross J., Gnes M. Modeling and Tools for Network Simulation. Springer, 2010.

43. Varga A. OMNeT++ Discrete Event Simulator. http://www.omnetpp.org.

44. Tommasik J., Weisse M.-A. The inter-domain hierarchy in measured and randomly generated AS-level topologies. In IEEE International Conference on Communications (ICC2012). IEEE, 2012.

45. Krishna K., Murty N. Genetic k-means algorithm // Trans. on SMC. 1999.

46. Celeux G., Govaert G. Gaussian parsimonious clustering models. 1995.

47. Sander J., Ester M., Kriegel H., Xu X. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. 1998.

48. Joachims T. Training linear svms in linear time // Proc. of SIGKDD. 2006.

49. Hochreiter S., Schmidhuber J. Long short-term memory // Neural computation. 1997.

50. More J. The levenberg-marquardt algorithm: implementation and theory // Numerical analysis. 1978.

51. Hetzner. [Електронний ресурс] // Офіційний сайт компанії. – Режим доступу: https://www.hetzner.com/

52. Гадевич В.Ю. Аналіз підходів до вибору апаратних ресурсів серверів для стрімінгових платформ // Матеріали Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Інтелектуальні Комп'ютерні Системи та Мережі» - Тернопіль: Західноукраїнський національний університет, кафедра комп'ютерної інженерії, 2024 р., С. 154-158.

53. Готяш Ю.П., Гадевич В.Ю. Аналіз мережевих параметрів для оптимізації вибору стрімінгового протоколу на базі ant media server // Міжнародна науково-практична інтернет-конференція «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» [Електронний ресурс] – Режим доступу: http://www.konferenciaonline.org.ua/ua/article/id-2018/

54. Комар М.П., Саченко А.О., Васильків Н.М., Загородня Д.І. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за другим (магістерським) рівнем вищої освіти. – Тернопіль: ЗУНУ, 2024. – 32 с.

Додаток А
Програмна реалізація методів прогнозування навантаження

1. ARIMA (AutoRegressive Integrated Moving Average)

```
from statsmodels.tsa.arima.model import ARIMA
import numpy as np

# Приклад даних (симуляція часового ряду)
data = np.random.randn(100)

# Налаштування ARIMA моделі
model = ARIMA(data, order=(5, 1, 0))  # (p, d, q) - порядок
ARIMA
model_fit = model.fit()

# Прогноз на наступні 10 кроків
forecast = model_fit.forecast(steps=10)
print("ARIMA Forecast:", forecast)
```

1. SVR (Support Vector Regression)

```
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
import numpy as np

# Генерація випадкових даних
X = np.arange(0, 100).reshape(-1, 1)  # Час як ознака
y = np.sin(X).ravel() + np.random.normal(0, 0.1, 100)  #
Нелінійні тренди

# Розподіл на тренувальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Налаштування SVR моделі
svr_model = SVR(kernel='rbf', C=1.0, epsilon=0.1)
```

```
svr_model.fit(X_train, y_train)

# Прогноз
predictions = svr_model.predict(X_test)
print("SVR Predictions:", predictions)
```

3. Bayesian Ridge Regression (BRR)

```
from sklearn.linear_model import BayesianRidge
from sklearn.model_selection import train_test_split
import numpy as np

# Генерація випадкових даних
X = np.arange(0, 100).reshape(-1, 1)
y = np.cos(X).ravel() + np.random.normal(0, 0.1, 100)

# Розподіл на тренувальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Налаштування BRR моделі
brr_model = BayesianRidge()
brr_model.fit(X_train, y_train)

# Прогноз
predictions = brr_model.predict(X_test)
print("BRR Predictions:", predictions)
```

4. LSTM (Long Short-Term Memory)

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

```
# Генерація випадкових даних
data = np.sin(np.linspace(0, 100, 500))
X, y = [], []
window_size = 10

# Створення вхідних даних
for i in range(len(data) - window_size):
    X.append(data[i:i + window_size])
    y.append(data[i + window_size])

X = np.array(X)
y = np.array(y)

# Розподіл на тренувальні та тестові набори
X_train, X_test = X[:400], X[400:]
y_train, y_test = y[:400], y[400:]

# Формування LSTM моделі
model = Sequential([
    LSTM(50, activation='relu', input_shape=(window_size, 1)),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
model.fit(X_train[..., np.newaxis], y_train, epochs=20,
batch_size=16)

# Прогноз
predictions = model.predict(X_test[..., np.newaxis])
print("LSTM Predictions:", predictions)
```

5. Прогнозування з часовим розривом (m-gap)

```
import numpy as np

# Симуляція даних
```

```
    data = np.sin(np.linspace(0, 100, 500))
    window_size = 10
    m_gap = 3

    # Створення вхідних даних для m-gap прогнозування
    X, y = [], []
    for i in range(len(data) - window_size - m_gap):
        X.append(data[i:i + window_size])
        y.append(data[i + window_size + m_gap])

    X = np.array(X)
    y = np.array(y)

    # Розподіл на тренувальні та тестові набори
    X_train, X_test = X[:400], X[400:]
    y_train, y_test = y[:400], y[400:]

    # Використання LSTM для m-gap прогнозування
    model = Sequential([
        LSTM(50, activation='relu', input_shape=(window_size, 1)),
        Dense(1)
    ])

    model.compile(optimizer='adam', loss='mse')
    model.fit(X_train[..., np.newaxis], y_train, epochs=20,
batch_size=16)

    # Прогноз
    predictions = model.predict(X_test[..., np.newaxis])
    print("m-gap Predictions:", predictions)
```

Додаток Б

Технічні конфігурації хмарних серверів

```
Cloud,vCPU,RAM (GB),SSD (GB),Monthly Rate (USD)
Hetzner,2,4,40,4.87
Hetzner,4,8,80,8.74
Hetzner,8,16,160,21.06
Hetzner,16,32,320,41.62
Azure,4,8,100,86.87
Azure,8,16,100,173.74
Azure,16,32,100,434.08
GCP,4,8,100,50.26
GCP,8,16,100,96.25
GCP,16,32,100,181.77
```

Додаток В

Програмна реалізація обробки текстових файлів

```
import pandas as pd
import io

cpuUsageFile = 'cpu_usage.txt'
memoryUsageFile = 'memory_usage.txt'
diskUsageFile = 'disk_usage.txt'
networkUsageFile = 'network_usage.txt'
outputFile = 'output.csv'

with open(cpuUsageFile, 'r') as file:
cpuFileContent = file.read()
cpuSarOutput = io.StringIO(cpuFileContent)

cpuDataFrame = pd.read_csv(cpuSarOutput, sep="\s+", header=1)
cpuColumn = cpuDataFrame.iloc[:, 3]

with open(memoryUsageFile, 'r') as file:
memoryFileContent = file.read()
memorySarOutput = io.StringIO(memoryFileContent)

memoryDataFrame = pd.read_csv(memorySarOutput, sep="\s+",
header=1)

memoryColumn = memoryDataFrame.iloc[:, 5]

with open(diskUsageFile, 'r') as file:
diskFileContent = file.read()
diskSarOutput = io.StringIO(diskFileContent)

# Skip the first few lines (header)
diskDataFrame = pd.read_csv(diskSarOutput, sep="\s+",
header=1)
```

```
    diskDataFrameFiltered = diskDataFrame[diskDataFrame.iloc[:, 2]
== 'sda']

    # Get the last column
    diskColumn = diskDataFrameFiltered.iloc[:, -1]

    with open(networkUsageFile, 'r') as file:
    networkFileContent = file.read()
    networkSarOutput = io.StringIO(networkFileContent)

    networkDataFrame = pd.read_csv(networkSarOutput, sep="\s+",
header=1)

    networkDataFrameFiltered =
networkDataFrame[networkDataFrame.iloc[:, 2] == 'eno2']

    networkColumn = networkDataFrameFiltered.iloc[:, -1]

    resultDataFrame = pd.DataFrame(columns=['CPU', 'RAM', 'Disk',
'Network'])

    resultDataFrame['CPU'] = cpuColumn
    resultDataFrame['RAM'] = memoryColumn

    diskColumnResetIndex = diskColumn.reset_index(drop=True)
    resultDataFrame['Disk'] = diskColumnResetIndex

    networkColumnResetIndex = diskColumn.reset_index(drop=True)
    resultDataFrame['Network'] = networkColumnResetIndex
    resultDataFrame.to_csv(outputFile)
```

Додаток Г

Копії публікацій

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

ВСЕУКРАЇНСЬКА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ СТУДЕНТІВ, АСПІРАНТІВ ТА МОЛОДИХ ВЧЕНИХ «ІНТЕЛЕКТУАЛЬНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

5 ЛИСТОПАДА 2024

KI.WUNU.EDU.UA/CONFERENCE/

ТЕРНОПІЛЬ

2024

Гадевич В. Ю.
магістрант 2 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Биковий П.Є., кафедра ІОСУ ЗУНУ

АНАЛІЗ ПІДХОДІВ ДО ВИБОРУ АПАРАТНИХ РЕСУРСІВ СЕРВЕРІВ ДЛЯ СТРІМІНГОВИХ ПЛАТФОРМ

Вступ. Стрімінгові сервіси, такі як YouTube, Netflix, Twitch і Facebook Live, відіграють важливу роль у сучасному цифровому середовищі, забезпечуючи користувачів доступом до аудіо- та відеоконтенту в реальному часі. Їх популярність продовжує зростати завдяки інтерактивним можливостям і високій якості обслуговування. Однак зі зростанням попиту виникають нові виклики: як ефективно управляти ресурсами, адаптуватися до пікових навантажень і забезпечувати оптимальну якість обслуговування для користувачів, які бажають проводити власні стріми.

Постановка задачі. Існують різні конфігурації апаратного забезпечення для обробки та зберігання даних, що відрізняються рівнями продуктивності та вартістю. Оптимальний вибір апаратного забезпечення для систем потокового відео, який дозволить забезпечити необхідну якість обслуговування (QoE) при мінімальних витратах на мережеві та серверні ресурси, є актуальним завданням.

Хмарні рішення. Відомі системи, як Hetzner, Microsoft Azure, Amazon Web Services (AWS) пропонують різні конфігурації на своїх серверах для обробки та зберігання даних з різними рівнями продуктивності та відповідними цінами (Рис. 1) [1].

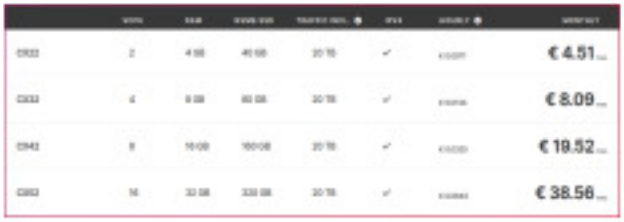


Рисунок 1 – Цінові пропозиції та конфігурації серверів Hetzner (станом на листопад 2024)

Microsoft Azure пропонує кілька варіантів серверних конфігурацій з адаптивними планами оплати, що дозволяє користувачам вибирати ресурси залежно від потреб у пропускній здатності, обчислювальних потужностях та зберіганні даних. Ціни залежать від вибору обчислювальних ресурсів (VM, бази даних, хмарні сервіси), а також тривалості використання та масштабу обробки даних.

AWS має схожу модель пропозиції, надаючи користувачам доступ до широкого спектра серверних конфігурацій через EC2 (Elastic Compute Cloud), а також інших хмарних сервісів для зберігання даних (S3), обробки великих даних, а також аналізу і машинного навчання. Ціни в AWS залежатимуть від вибору конкретних інстанцій, таких як загального призначення або спеціалізовані інстанції для високих навантажень.

Підходи до вибору апаратних ресурсів. Для вибору відповідного сервера в системах стрімінгового відео існує кілька підходів, кожен з яких має свої переваги і використовується залежно від потреб системи та доступних ресурсів.

Прості алгоритми на основі статичної інформації є найбільш підходящими для провайдерів, які не мають змоги проводити динамічний моніторинг [2] стану мережі та серверів. Ці алгоритми працюють з фіксованими параметрами, такими як відображення контенту на певні сервери, розподіл серверів і клієнтів по мережевих вузлах або оцінка довжини мережевих шляхів. Наприклад, вибір найближчого сервера, заснований на найкоротшому мережевому шляху, часто забезпечує найкращі результати з точки зору продуктивності та мінімізації витрат.

Такий підхід є простим у реалізації і добре підходить для умов із передбачуваним навантаженням.

Для більш складних сценаріїв, де необхідно враховувати велику кількість змінних і забезпечувати високий рівень продуктивності, використовуються багатокритеріальні методи прийняття рішень (MCDA) [3]. Цей підхід дозволяє аналізувати й оцінювати доступну потужність серверів, пропускну здатність мережевих шляхів і навіть додаткові параметри, такі як енергоспоживання чи затримки. Хоча впровадження MCDA потребує інфраструктури для моніторингу ресурсів, ефективність і точність цього методу виправдовують витрати, особливо для великих мереж. Цей підхід найкраще підходить для багатоярусних мереж і складних стрімінгових платформ, де важливо забезпечити стабільність навіть при великих навантаженнях.

Альтернативою попередніх рішень є комбінований підхід до вибору серверів, який розподіляє завдання між постачальником послуг і клієнтом. У цьому випадку постачальник формує короткий список серверів на основі статичної інформації, а клієнт застосовує власний алгоритм, наприклад MCDA, для остаточного вибору. Такий підхід дозволяє досягти результатів, подібних до MCDA, але при цьому значно скоротити витрати на моніторинг.

Динамічний моніторинг. Ще один підхід базується на динамічному моніторингу стану мережі та сервера. Даний підхід запропоновано виконати на базі Ant Media Server [4]. З його допомогою можна здійснити збір точних та актуальних даних про навантаження на ресурси. Дана інформація дозволить здійснювати прогноз навантаження і дозволить оптимізувати використання апаратних засобів сервера.

Процес збору даних про використання ресурсів серверу здійснюється за допомогою утиліти sar (System Activity Report). Ця утиліта дозволяє збирати та аналізувати статистику роботи системи, включаючи використання процесора (CPU), пам'яті (RAM), диска (Disk) та мережі (Network).

```
sar -u 5 > cpu_usage.txt & sar -r 5 > memory_usage.txt & sar -d 5 >
disk_usage.txt & sar -n DEV 5 > network_usage.txt &
```

Ця команда дозволяє збирати дані про використання процесора, пам'яті, диска та мережевих ресурсів кожні 5 секунд. Інформація про процесор зберігається у файл cpu_usage.txt, про пам'ять — у файл memory_usage.txt, про диск — у файл disk_usage.txt, а дані про мережеве використання — у файл network_usage.txt. Це дає можливість здійснювати моніторинг системних ресурсів у реальному часі та зберігати їх для подальшої обробки і аналізу.

У таблиці 1 представлений моніторинг мережевої активності різних інтерфейсів сервера у реальному часі. Таблиця містить дані про роботу таких інтерфейсів, як lo, eno1 і eno2, для кожного з яких зафіксовано кількість отриманих і переданих пакетів, обсяг переданих даних, а також інші показники мережевої активності. Локальний інтерфейс lo демонструє стабільну передачу невеликих обсягів даних, що характерно для внутрішньої взаємодії системи. Інтерфейс eno1 має мінімальну активність, вказуючи на обмежене використання цього каналу.

Таблиця 1 - Моніторинг мережевої активності різних інтерфейсів сервера

01:33:05 PM	IFACE	rxpck/s	txpck/s	rxkB/s	txkB/s	rxcmp/s	txcmp/s	rxmcst/s	%ifutil
01:33:10 PM	lo	4.40	4.40	1.64	1.64	0.00	0.00	0.00	0.00
01:33:10 PM	eno1	3.20	0.00	0.75	0.00	0.00	0.00	0.60	0.00
01:33:10 PM	eno2	17.80	10.40	2.69	1.90	0.00	0.00	0.60	0.00
01:33:10 PM	IFACE	rxpck/s	txpck/s	rxkB/s	txkB/s	rxcmp/s	txcmp/s	rxmcst/s	%ifutil
01:33:15 PM	lo	4.40	4.40	1.64	1.64	0.00	0.00	0.00	0.00
01:33:15 PM	eno1	2.80	0.00	0.84	0.00	0.00	0.00	0.40	0.00
01:33:15 PM	eno2	15.60	9.00	2.59	1.73	0.00	0.00	0.40	0.00
01:33:15 PM	IFACE	rxpck/s	txpck/s	rxkB/s	txkB/s	rxcmp/s	txcmp/s	rxmcst/s	%ifutil
01:33:20 PM	lo	4.40	4.40	1.64	1.64	0.00	0.00	0.00	0.00
01:33:20 PM	eno1	2.80	0.00	0.62	0.00	0.00	0.00	0.40	0.00
01:33:20 PM	eno2	16.00	9.00	2.50	1.72	0.00	0.00	0.40	0.00

Водночас eno2 показує значно більший рівень активності, з помітною кількістю отриманих і переданих пакетів, а також вищими показниками обсягу переданих даних. Такий моніторинг дозволяє оцінити продуктивність мережі та визначити, який інтерфейс є найбільш завантаженим.

В таблиці 2 показано інформацію про активність різних пристроїв, включаючи логічні томи (loop0-loop7), фізичні диски (sda, sdb), оптичний привід (sr0) та віртуальний диск (dm-0). Для кожного пристрою наведені ключові показники роботи. Більшість логічних томів (loop) не мають активності в цей час, що свідчить про їхнє невикористання для зчитування чи запису даних. Фізичний диск sda демонструє помірну активність із показниками 5.60 кілобайт на секунду для читання та 14.40 кілобайт на секунду для запису. Пристрій dm-0 також має подібні показники запису, але не задіяний для читання. Інші пристрої, такі як sdb та sr0, залишаються неактивними.

Таблиця 2 - Поточне навантаження на пристрої зберігання інформації сервера

01:33:05 PM	DEV	tps	rkB/s	wkB/s	dkB/s	areq-sz	aqu-sz	await	%util
01:33:10 PM	loop0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	loop1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	loop2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	loop3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	loop4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	loop5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	loop6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	loop7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	sdb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	sda	5.60	0.00	14.40	0.00	2.57	0.00	0.07	0.48
01:33:10 PM	sr0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
01:33:10 PM	dm-0	7.00	0.00	14.40	0.00	2.06	0.00	0.00	0.48

В таблиці 3 відображені ключові метрики, що дозволяють оцінити обсяг доступної, використаної та кешованої пам'яті на різних часових інтервалах. Колонка kbmemfree відображає обсяг вільної пам'яті в кілобайтах, який залишається стабільним на рівні близько 9,059,612 кілобайт протягом усього періоду спостережень. Значення в колонці kbavail показують доступну пам'ять, яка також зберігається на постійному рівні близько 10,112,000 кілобайт. У той же час колонка kbmemused демонструє використану пам'ять, яка складає приблизно 5,702,000 кілобайт, що відповідає рівню завантаження оперативної пам'яті в 35%. Крім того, значення в колонках kbbuffers і kbcached показують обсяги пам'яті, виділеної під буфери та кеш, які залишаються стабільними на рівні близько 144,560 і 1,107,476 кілобайт відповідно. Метрика kbcommit вказує на обсяг пам'яті, зарезервованої для виконання поточних завдань, яка становить близько 2,949,580 кілобайт. Додатково колонки kbactive і kbinact відображають активну та неактивну пам'ять, значення яких становлять приблизно 5,919,000 і 6,094,000 кілобайт відповідно. Значення у колонці kbdirty, яке показує обсяг «брудної» пам'яті (що очікує запису на диск), залишаються дуже низькими, коливаючись між 44 і 64 кілобайтами.

Таблиця 3 - Динаміка використання оперативної пам'яті сервера в реальному часі

01:33:05 PM	kbmemfree	kbavail	kbmemused	%memused	kbbuffers	kbcached	kbcommit	%commit	kbactive	kbinact	kbdirty
01:33:10 PM	9059664	10112388	5702744	35.00	144560	1107476	2949580	14.40	591928	6094208	64
01:33:15 PM	9059664	10112388	5702736	35.00	144568	1107476	2949580	14.40	591928	6094276	40
01:33:20 PM	9059664	10112404	5702720	35.00	144576	1107476	2949580	14.40	591944	6094280	40
01:33:25 PM	9059664	10112412	5702720	35.00	144576	1107476	2949564	14.40	591952	6094208	12
01:33:30 PM	9059664	10112420	5702712	35.00	144584	1107484	2949532	14.40	591952	6093960	44
01:33:35 PM	9059664	10112424	5702704	35.00	144592	1107484	2949516	14.40	591952	6093980	8
01:33:40 PM	9059664	10112428	5702700	35.00	144600	1107484	2941308	14.36	591952	6093752	56
01:33:45 PM	9059664	10112428	5702688	35.00	144608	1107488	2941308	14.36	591952	6093752	8
01:33:50 PM	9059664	10112468	5702680	35.00	144616	1107488	2941308	14.36	591992	6093756	44
01:33:55 PM	9059664	10112480	5702672	35.00	144624	1107488	2941308	14.36	592000	6093760	4
01:34:00 PM	9059664	10112480	5702652	35.00	144632	1107500	2941308	14.36	592000	6093760	48
01:34:05 PM	9059664	10112480	5702644	35.00	144640	1107500	2941292	14.36	592000	6093624	8

В таблиці 4 представлено інформацію про використання процесора в різних режимах, включаючи виконання користувацьких і системних процесів, а також про час простою. Значення в колонці `%user` показують частку завантаження, пов'язаного з виконанням користувацьких програм. Вона поступово зростає, досягаючи пікових значень близько 12.16% на одному з етапів. Колонка `%system` демонструє мінімальні значення, які не перевищують 2.32%, вказуючи на низьке завантаження системних процесів. Час очікування вводу-виводу (`%iowait`) залишається майже на нульовому рівні, що свідчить про відсутність значних затримок у роботі із зовнішніми пристроями. Колонка `%idle`, яка відображає відсоток простою процесора, поступово знижується зі зростанням навантаження, досягаючи мінімуму на рівні 84.42%.

Таблиця 4 - Динаміка завантаження центрального процесора сервера в реальному часі

01:33:05 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
01:42:10 PM	all	5.20	0.00	0.44	0.00	0.00	94.36
01:42:15 PM	all	6.71	0.00	0.27	0.01	0.00	93.01
01:42:20 PM	all	9.76	0.00	0.32	0.01	0.00	89.91
01:42:25 PM	all	7.51	0.00	0.22	0.01	0.00	92.26
01:42:30 PM	all	7.86	0.00	0.19	0.00	0.00	91.95
01:42:35 PM	all	10.01	0.00	0.42	0.00	0.00	89.57
01:42:40 PM	all	12.16	0.00	3.42	0.01	0.00	84.42
01:42:45 PM	all	11.30	0.00	1.64	0.01	0.00	87.05
01:42:50 PM	all	9.65	0.00	1.90	0.00	0.00	88.44
01:42:55 PM	all	10.11	0.00	1.52	0.01	0.00	88.35
01:43:00 PM	all	10.51	0.00	2.32	0.00	0.00	87.17
01:43:05 PM	all	10.03	0.00	1.52	0.00	0.00	88.45
01:43:10 PM	all	8.48	0.00	1.49	0.00	0.00	90.04
01:43:15 PM	all	10.74	0.00	1.59	0.01	0.00	87.67
01:43:20 PM	all	8.79	0.00	1.38	0.00	0.00	89.83

Висновки. Загалом вибір методу залежить від потреб системи, доступних ресурсів і вимог до продуктивності. Використання різних підходів або їх поєднання дозволяє оптимізувати роботу серверів, забезпечуючи стабільність і високу якість обслуговування користувачів. В даній роботі запропоновано порівняння різних підходів до вибору сервера та описано метод динамічного збору інформації про навантаження сервера на базі Ant Media Server. Дана інформація є основою для методу прогнозування навантаження на сервер [5], що дозволяє значно покращити продуктивність систем потокового відео та забезпечує ефективне використання ресурсів і підтримку високого рівня якості обслуговування для кінцевих користувачів.

Список літератури

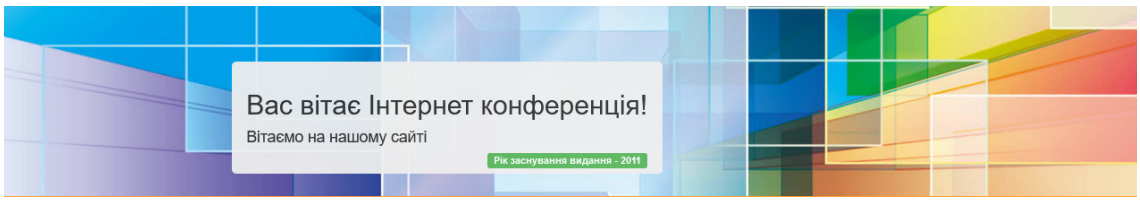
1. Hetzner. "Official website of the company." [Online]. Available: https://www.hetzner.com/.

2. A. Gohar and S. Lee, "Multipath Dynamic Adaptive Streaming over HTTP Using Scalable Video Coding in Software Defined Networking," *Applied Sciences*, vol. 10, no. 10, p. 16, Oct. 2020. DOI: 10.3390/app10217691.

3. O. Catrina, "Multi-Criteria Server Selection for Over-the-Top Video Streaming," 2016 *International Conference on Communications (COMM)*, Bucharest, Romania, 2016, pp. 303–308. DOI: 10.1109/ICComm.2016.7528293.

4. Ant Media. "Ant Media Server Documentation." [Online]. Available: https://antmedia.io/.

5. R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS," *IEEE Transactions on Cloud Computing (TCC)*, vol. 3, no. 4, pp. 449–458, Oct.–Dec. 2015. DOI: 10.1109/TCC.2015.2459802.



ІМПУТАЦІЯ ПРОПУЩЕНИХ ДАНИХ ЗА ДОПОМОГОЮ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ ТА НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ

[1. Інформаційні системи і технології]

Автор: **Гончар Ярослав Андрійович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

АНАЛІЗ МЕРЕЖЕВИХ ПАРАМЕТРІВ ДЛЯ ОПТИМІЗАЦІЇ ВИБОРУ СТРІМІНГОВОГО ПРОТОКОЛУ НА БАЗІ ANT MEDIA SERVER

[1. Інформаційні системи і технології]

Автор: **Готяш Юрій Павлович, магістрант, Західноукраїнський національний університет, м. Тернопіль; Гадевич Володимир Юрійович, магістрант, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

АВТОМАТИЗОВАНА СИСТЕМА ПРОГНОЗУВАННЯ ЕФЕКТИВНОСТІ ПРИВАТНОЇ СОНЯЧНОЇ ЕЛЕКТРОСТАНЦІЇ

[1. Інформаційні системи і технології]

Автор: **Григоренко Олег Ігорович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

РОЗПОДІЛЕНЕ ГЛИБОКЕ НАВЧАННЯ ДЛЯ ОБРОБКИ ДАНИХ ІНТЕРНЕТУ РЕЧЕЙ

[1. Інформаційні системи і технології]

Автор: **Демчишин Владислав Володимирович, магістр, Західноукраїнський національний університет, м. Тернопіль**

Відкрити тези доповіді »

ІНТЕГРАЦІЯ GOOGLE SHEETS З PYTHON ТА MATLAB ДЛЯ СКЛАДНИХ ОБЧИСЛЕНЬ

[1. Інформаційні системи і технології]

Автор: **Замуруєва О.В., кандидат фізико-математичних наук, доцент кафедри теоретичної та комп'ютерної фізики імені А. В. Свідзинського Навчально-наукового фізико-технологічного інституту Волинського національного університету імені Лесі Українки, м. Луцьк; Шваліковський А., студент 1-ого курсу Навчально-наукового фізико-технологічного інституту Волинського національного університету імені Лесі Українки, м. Луцьк; Мельничук А., студент 1-ого курсу Навчально-наукового фізико-технологічного інституту Волинського національного університету імені Лесі Українки,**

***Готяш Юрій Павлович**, магістрант,*
Західноукраїнський національний університет, м. Тернопіль
***Гадевич Володимир Юрійович**, магістрант,*
Західноукраїнський національний університет, м. Тернопіль
Науковий керівник: Биковий Павло Євгенович,
кандидат технічних наук, доцент,
Західноукраїнський національний університет, м. Тернопіль

АНАЛІЗ МЕРЕЖЕВИХ ПАРАМЕТРІВ ДЛЯ ОПТИМІЗАЦІЇ ВИБОРУ СТРІМІНГОВОГО ПРОТОКОЛУ НА БАЗІ ANT MEDIA SERVER

Анотація

У статті описано методики вимірювання ключових параметрів комп’ютерної мережі, зокрема затримки, пропускної здатності та втрат пакетів. Детально розглянуто реалізацію цих вимірювань на базі Ant Media Server і проведено аналіз отриманих даних. Отримані результати є вхідними даними для системи автоматичного вибору протоколу, залежно від вимог до якості, затримки та стабільності з'єднання.

Постановка задачі

Пряма (Live) трансляція полягає в процесі передачі аудіо- та відеосигналів від джерела на сервер, де вони обробляються та передаються кінцевим користувачам. Для передачі сигналів від джерела до сервера зазвичай використовується протокол RTMP. Протокол WebRTC забезпечує мінімальну затримку при передачі даних безпосередньо кінцевим користувачам у режимі реального часу. Протокол DASH дозволяє адаптувати якість відео до змінних мережевих умов, що забезпечує стабільність відтворення, але має більшу затримку. У таких умовах автоматичний вибір протоколу, враховуючи

різноманітність мережевих підключень та пристроїв кінцевих користувачів, є актуальною задачею [1].

Вимірювання затримки

Вимірювання затримки здійснюється за допомогою утиліти ping, яка є одним із найефективніших способів оцінити час, необхідний для передачі пакету даних між двома точками в мережі. Утиліта ping надсилає ICMP-запит до вказаного хоста та вимірює час, що проходить від моменту відправлення запиту до отримання відповіді. Це дозволяє визначити затримку у мілісекундах, оцінити якість мережевого з'єднання та виявити можливі проблеми, такі як висока затримка або втрати пакетів. Затримка до 50 мс вважається гарним показником для більшості стрімінгових застосунків, тоді як значення понад 100 мс можуть свідчити про потенційні проблеми з мережевим з'єднанням.

Вимірювання пропускної здатності

Iperf3 — це потужний та гнучкий інструмент для тестування пропускної здатності мережі, який дозволяє вимірювати швидкість передачі даних між двома пристроями. Для проведення тесту необхідно запустити iperf3 у режимі сервера на одному комп’ютері та в режимі клієнта на іншому. Сервер, зазвичай, працює на стандартному порту 5201 (або іншому, якщо вказано), очікуючи підключення клієнта, який ініціює тестування.

Інструмент підтримує як TCP, так і UDP протоколи, що дозволяє виконувати різні типи тестів, включаючи оцінку затримки та втрат пакетів для UDP-з’єднань. Використовуючи iperf3, можна налаштовувати різні параметри тесту, такі як тривалість вимірювання, кількість потоків передачі даних, пропускну здатність для UDP-з’єднань.

Результати тесту відображаються у вигляді статистики, яка включає середню швидкість передачі даних, затримки та, для UDP, втрати пакетів. Ця інформація дозволяє точно оцінити продуктивність мережі та виявити її слабкі місця. Завдяки своїй простоті та широким можливостям налаштування, iperf3 є

одним із найпопулярніших інструментів для мережевих адміністраторів і інженерів.

Для використання iperf3 спочатку потрібно встановити програму на серверній машині. Після інсталяції сервер запускається в режимі очікування з'єднань клієнта. Як тільки сервер активується, він готовий до тестування.

Вимірювання втрат пакетів

Вимірювання втрат пакетів здійснюється за допомогою утиліти ping з параметром -c 100 дозволяє оцінити якість з'єднання та визначити, чи є проблеми з доставкою даних через мережу. Параметр -c 100 вказує утиліті відправити 100 ICMP запитів до вказаного хоста, що дає змогу отримати статистику про можливі втрати пакетів протягом серії перевірок. Це дозволяє оцінити стабільність з'єднання в умовах більше тривалого тестування, що знижує ймовірність випадкових коливань і дає точніше уявлення про ефективність мережі.

Результати

Для вимірювання був використаний Ant Media Server, як платформа для трансляції відео в реальному часі, яка підтримує популярні срімінг протоколи RTMP, HLS, DASH, WebRTC [2]. Цей сервер оптимізовано для забезпечення високоякісного відео стрімінгу з низькою затримкою, що робить його ідеальним для таких додатків, як відеоконференції, ігрові стріми, навчальні платформи та інші інтерактивні трансляції.

Після вимірювання затримки, пропускної здатності та втрат пакетів всі дані були записані у файл data.csv для подальшого аналізу. Частина даного файлу представлена нижче:

```
2024-11-10 T15:43:51.894072597Z,32,1,15
2024-11-10 T15:48:51.894036837Z,41,1,21
2024-11-10 T15:53:51.894124444Z,28,0,23
2024-11-10 T15:58:51.894158470Z,291,1,22
2024-11-10 T16:03:51.894053477Z,166,1,9
2024-11-10 T16:08:51.894161324Z,42,2,4
2024-11-10 T16:13:51.894109856Z,38,1,11
2024-11-10 T16:18:51.894125589Z,25,2,4
2024-11-10 T16:23:51.894146960Z,32,2,17
2024-11-10 T16:28:51.894116286Z,48,1,6
2024-11-10 T16:33:51.894153890Z,24,1,5
2024-11-10 T16:38:51.894178865Z,36,1,7
2024-11-10 T16:43:51.894116818Z,27,1,10
```

В ході дослідження показано, що WebRTC забезпечує мінімальну затримку, що особливо важливо для інтерактивних трансляцій і відеозв'язку. Водночас DASH дозволяє адаптувати якість відео до змінних мережевих умов, забезпечуючи стабільне відтворення навіть за нестабільного з'єднання, проте він є повільніший. На відміну від HLS, який через буферизацію має вищу затримку, ці протоколи значно покращують досвід користувачів, пропонуючи більш гнучкі та ефективні рішення для різних сценаріїв використання.

Висновок

У даній статті проведено аналіз якості мережевих з'єднань для забезпечення прямої трансляції з використанням Ant Media Server, який забезпечує який високу продуктивність і гнучкість у роботі з стрімінг протоколами. Дослідження охопило вимірювання затримки, пропускної здатності та втрат пакетів із використанням утиліт ping та iperf3, що дозволило оцінити стабільність і ефективність мережевої інфраструктури. Результати дослідження дозволи оцінити реальні умови експлуатації, проаналізувати ключові показники мережі для подальшого вибору протоколу для різних сценаріїв використання. Результати тестувань підтвердили доцільність автоматичного вибору протоколу залежно від потреб конкретної трансляції, що дозволяє забезпечити оптимальну якість обслуговування для кінцевих користувачів.

Література:

1. Letzgro. How to tackle frequent live video streaming challenges [Електронний ресурс]. – Режим доступу: https://letzgro.net/how-to-tackle-frequent-live-video-streaming-challenges/
2. Ant Media. Ant Media Server Documentation [Електронний ресурс]. URL: https://antmedia.io.