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

Грицай Ростислав Ігорович

Модель хмарної передачі відео в реальному часі / Cloud-based real-time video transmission model

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

Кваліфікаційна робота

Виконав студент групи КНм-21
Р. І. Грицай
_____

Науковий керівник:
к.т.н., доцент Осолінський О.Р.
_____

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
Грицаю Ростиславу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Модель хмарної передачі відео в реальному часі / Cloud-based real-time video transmission model.
керівник роботи к.т.н., доцент Осолінський О.Р.
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;
- дослідити методи хмарних обчислень для потокового відео;
- проаналізувати проблеми та рішення потокового відео;
- дослідити особливості хмарного потокового відео;
- вибір перспективного шляху і постановка задачі дослідження;
- розробити метод хмарної трансляції відео;
- удосконалити метод планування на базі якості обслуговування;
- розробити архітектуру системи трансляції відео в реальному часі за допомогою хмарних сервісів;
- розробити модель потоку клієнта;
- провести аналіз відео за допомогою Kafka та Spark;
- провести налаштування конфігурації Kafka об'єкта JSON для аналізу відео;
- провести оцінку результатів тестування

5. Перелік графічного матеріалу у роботі
- загальна архітектура системи.

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 _____ Р. І. Грицай
підпис

Керівник роботи _____ к.т.н., доцент Осолінський О.Р.
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Модель хмарної передачі відео в реальному часі» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 71 сторінки і містить 15 ілюстрації, 4 додатки та 41 використаних джерел.

Метою даної кваліфікаційної роботи є розробка моделі хмарної передачі відео в реальному часі яка об'єднує інноваційні підходи до організації потокового відео, що включають динамічне пакування, хмарне перекодування та використання адаптивних методів для забезпечення високої якості обслуговування.

Методи досліджень: включають аналіз наукових досліджень існуючих методів формування потокового відео за допомогою хмарних технологій, мережевих моделей передачі відео, алгоритмів пакування та перекодування відео.

Результати дослідження: вдосконалено моделі хмарної передачі відео в реальному часі та реалізації паралельного програмування та моделі «виробник-споживач», що оптимізує процеси обробки й передачі даних, знижуючи затримки у складних умовах мережі.

Результати роботи можуть успішно застосовуватися для передачі відео високої якісті обслуговування зі можливісттю зниження вартості оренди послуг.

Ключові слова: ПОТОКОВЕ ВІДЕО В РЕАЛЬНОМУ ЧАСІ, ПОТОКОВЕ ВІДЕО НА ВИМОГУ, KAFKA, RABBITMQ, OPENCV, SPARK, СЛУЖБИ AWS, ХМАРНІ ОБЧИСЛЕННЯ

ABSTRACT

The qualification work on the topic "Cloud-based real-time video transmission model" for obtaining the educational degree "Master" in specialty 122 "Computer Science" of the educational program "Computer Science" is written in a volume of 71 pages and contains 15 illustrations, 4 appendices and 41 used references.

The purpose of this qualification work is to develop a model of real-time cloud video transmission that combines innovative approaches to organizing streaming video, including dynamic packaging, cloud transcoding and the use of adaptive methods to ensure high quality of service.

Research methods: include an analysis of scientific research on existing methods of forming streaming video using cloud technologies, network video transmission models, video packaging and transcoding algorithms.

Research results: the model of real-time cloud video transmission and the implementation of parallel programming and the "producer-consumer" model, which optimizes data processing and transmission processes, reducing delays in difficult network conditions, have been improved.

The results of the work can be successfully applied to the transmission of high-quality video services with the possibility of reducing the cost of renting services.

Keywords: LIVE VIDEO STREAMING, VIDEO ON-DEMAND STREAMING, KAFKA, RABBITMQ, OPENCV, SPARK, AWS SERVICES, CLOUD COMPUTING

ЗМІСТ

ВСТУП

Потокове відео у різних формах відео на вимогу (VOD-video on demand), у прямому ефірі та потоковому ефірі різко зросло за останні кілька років. У порівнянні з традиційними кабельними каналами, вміст яких можна дивитися лише на телевізорах, потокове відео є повсюдним, і глядачі можуть гнучко переглядати відеоконтент на різних пристроях, починаючи від смартфонів і закінчуючи ноутбуками, а також на смарт телевізорах. Така повсюдність і гнучкість забезпечуються завдяки застосуванню багатьох технологій, таких як стиснення відео, хмарні обчислення, мережі доставки контенту та інші технології.

Ідея прийому потокового відеовмісту бере свій початок з винаходу телебачення на початку 20 століття. Однак середовище, на якому користувачі отримують і переглядають відеоконтент, суттєво змінилося протягом останнього десятиліття — від звичайних телевізорів до потокової трансляції на різноманітних пристроях (наприклад, ноутбуках, настільних ПК та планшетах) через Інтернет. Прийняття потокового відео в Інтернеті стрімко зросло до такої міри, що воно домінує у всьому Інтернет-трафіку. Звіт Global Internet Phenomena показує, що потокове відео вже становить понад 60% усього Інтернет-трафіку [1]. Кількість абонентів Netflix вже перевищила кількість абонентів кабельного телебачення в США [2].

Сьогодні багато Інтернет-додатків працюють на основі потокового відео. Такі додатки включають відеоконтент, створений користувачами (наприклад, на YouTube, Vimeo), прямі трансляції та особисті трансляції через соціальні мережі (наприклад, UStreamd і Facebook Livee), потокове передавання (OTT- over the top) (наприклад, Netflix і Amazon Primef).), системи електронного навчання, наприклад, Udemyg, [3], платформа потокової трансляції ігор (наприклад, Twitchh), системи відеочату та конференцій [4], системи боротьби зі стихійними лихами та системи безпеки,

які працюють на основі відеоспостереження [5], а також мережеві широкі трансляції каналів (наприклад, новини та інші телеканали) [6].

Із зростанням популярності служб потокового відео, вони вимагають більше комп'ютерних послуг для потокового передавання. Зростаюча популярність і впровадження потокової передачі збіглися з поширеністю технологій хмарних обчислень. Хмарні постачальники пропонують широкий спектр обчислювальних послуг і дозволяють користувачам передавати свої обчислювальні вимоги на аутсорсинг. Хмарні провайдери звільняють провайдерів потокового відео від додаткового навантаження та наслідків підтримки та оновлення дорогої обчислювальної інфраструктури [7]. На даний момент постачальники потокового відео значною мірою покладаються на хмарні сервіси для більшості або всіх своїх обчислювальних вимог [8]. Поєднання потокового відео та хмарних сервісів породило низку нових невирішених завдань, методів і технологій у обчислювальній галузі.

Метою роботи є розробка моделі хмарної передачі відео в реальному часі. Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Дослідити методи хмарних обчислень для потокового відео.
2. Проаналізувати проблеми та рішення потокового відео.
3. Дослідити особливості хмарного потокового відео.
4. Розробити метод хмарної трансляції відео.
5. Удосконалити метод планування на базі якості обслуговування
6. Розробити архітектуру системи трансляції відео в реальному часі за допомогою хмарних сервісів.
7. Розробити модель потоку клієнта.
8. Провести аналіз відео за допомогою Kafka та Spark.
9. Провести налаштування конфігурації Kafka об'єкта JSON для аналізу відео.
10. Провести оцінку результатів тестування

Об'єктом дослідження є процес формування потокового відео для користувача.

Предметом дослідження є методи стиснення, сегментації, зберігання потокового відео.

Методи дослідження включають аналіз наукових досліджень існуючих методів формування потокового відео за допомогою хмарних технологій, мережевих моделей передачі відео, алгоритмів пакування та перекодування відео.

Практичне значення дослідження полягає в можливості застосування нових моделей відображення потокового відео з меншими часовими та фінансовими затратами.

Структура та обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися на IV Міжнародній науковій конференції «Період трансформаційних процесів в світовій науці: задачі та виклики» (м. Рівне, Україна) та VIII Міжнародній студентській науковій конференції «Актуальні питання та перспективи проведення наукових досліджень» (м. Дніпро, Україна).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Хмарні обчислення для потокового відео

Щоб забезпечити високу якість досвіду (QoE) для багатьох глядачів, розташованих по всьому світу з різними пристроями відображення та мережевими характеристиками, постачальники потокового відео зазвичай здійснюють попередню обробку (наприклад, попереднє кодування та розфасовку) і зберігають відеоконтент у кількох версіях [9]. Таким чином, глядачі з різними пристроями відображення можуть легко знайти версію, яка відповідає їхнім пристроям. Альтернативним підходом до попередньої обробки відео є обробка їх у відкладений (тобто на вимогу) спосіб [7, 9]. Зокрема, цей підхід може бути корисним для відео, до яких рідко звертаються. Нещодавні аналітичні дослідження статистичних закономірностей доступу до відеопотоків показують, що доступ до багатьох відеопотоків відбувається рідко, і лише до невеликої частини (приблизно 5%) відео (зазвичай їх називають гарячими відеопотоками) часто відкривають. І для попередньої обробки, і для підходів на вимогу постачальник потокового відео потребує надання величезних обчислювальних ресурсів.

Підтримка та модернізація внутрішньої інфраструктури для задоволення потреб постачальників відеопотоку в обчисленнях, сховищах і мережах коштує дорого. Крім того, технічно він далекий від основного бізнесу потокових провайдерів, яким є виробництво та публікація відеоконтенту. Крім того, постачальники хмарних послуг, такі як Amazon Cloud (AWS), Google Cloud і Microsoft Azure, можуть задовольнити ці вимоги, пропонуючи ефективні послуги з високою доступністю [10]. Постачальники потокового відео стали широко покладатися на хмарні служби для більшості або всіх своїх обчислювальних вимог. Наприклад, Netflix передала всі свої обчислювальні вимоги хмарі Amazon [11].

Оскільки хмарні провайдери стягують плату за свої послуги зі своїх користувачів за принципом оплати за використання [12], завдання для стрім-провайдерів полягає в тому, щоб мінімізувати свої хмарні витрати, пропонуючи при цьому певний рівень QoE для своїх глядачів.

Було проведено багато досліджень, щоб усунути або вирішити проблеми потокового відео за допомогою хмарних сервісів. Наприклад, дослідники вивчали вартість розгортання хмарних служб для транскодування відео [10, 13], економічні переваги різних моделей сегментації відео [9, 14], застосування спеціальних методів планування для завдань потокового відео [13, 15], і методи надання ресурсів (віртуальних машин) для потокового відео [9, 16]. Тим не менш, ці дослідження здебільшого зосереджені на одному аспекті обробки потокового відео та тому, як цей аспект можна ефективно виконувати в хмарі. Потрібні подальші дослідження, щоб позиціонувати ці роботи в загальній картині та забезпечити вищий рівень бачення ефективного використання хмарних служб для всього робочого процесу потокового відео.

1.1.1 Структура відеопотоку

Як показано на рисунку 1.1, відеопотік складається з послідовності кількох менших сегментів. Кожен сегмент містить декілька груп зображень (GOP- -group of pictures) із заголовком сегмента на початку кожної групи зображень. Заголовок сегмента містить таку інформацію, як кількість груп зображень у цьому сегменті та тип GOP. Група зображень складається з послідовності кадрів. Перший кадр є I (внутрішнім) кадром, за яким йдуть кілька P (прогнозований) і B (двонаправлений прогнозований) кадри. Кадр у груповій групі поділено на кілька фрагментів, і кожен фрагмент складається з кількох макроблоків (МБ). Макроблок розглядаються, як одиниця для операцій кодування та декодування відео.

Може існувати два типи груп зображень, а саме закрита GOP і відкрита GOP. Перша група зображень є незалежною, тобто немає зв’язку. Таким

чином, закриті GOP можна обробляти незалежно. Крім того, у відкритому GOP залежить від іншого GOP, отже, не може оброблятися незалежно.

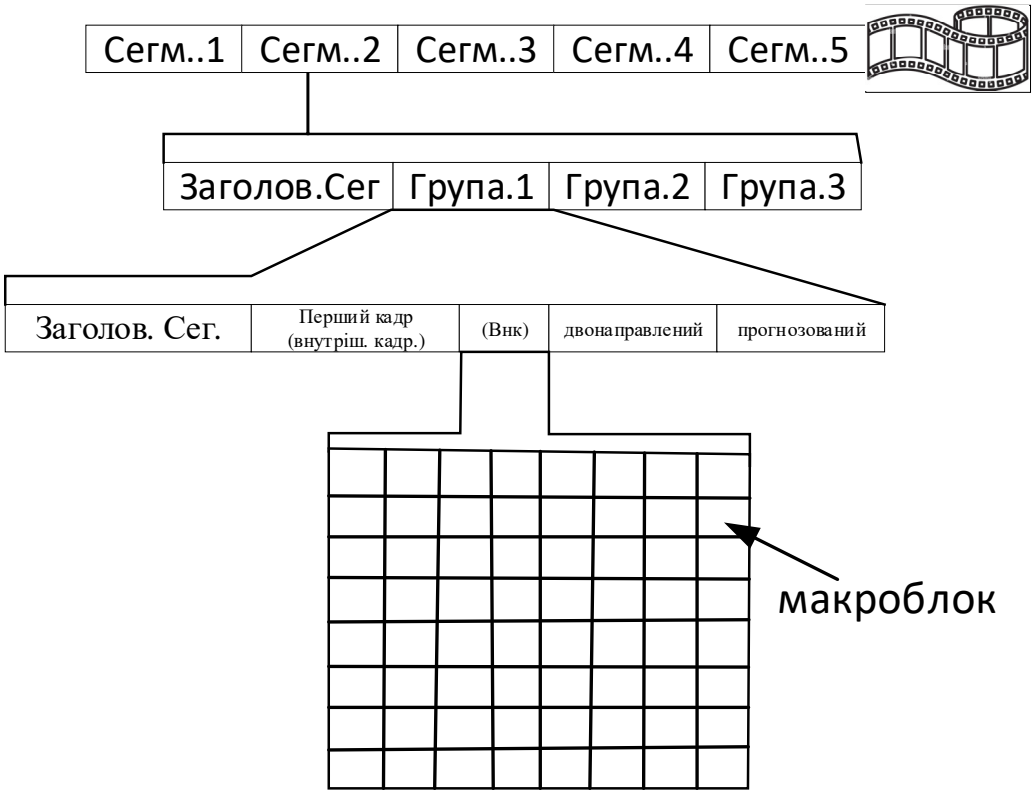


Рисунок 1.1 - Структура відеопотоку.

Як видно з рисунку відеопотік містить кілька сегментів, і кожен сегмент містить кілька груп зображень. Кадр у такій групі містить певну кількість макроблоків (МБ).

Щоб обробити відеопотоки, їх можна розділити на різні рівні, а саме рівень сегмента, рівень групи зображень, рівень кадру, рівень фрагмента та рівень макроблоку. Рівень послідовності містить кілька груп зображень, які можна обробляти незалежно. Однак через великий розмір послідовності її передача та час обробки стають вузьким місцем для всього процесу передачі [17]. Обробка на рівнях кадру, зрізу та макроблоку передбачає роботу з просторово-часовими залежностями, що робить обробку складною та повільною [17]. На практиці постачальники відеопотоку зазвичай виконують обробку на рівнях сегментів або групи зображень. Тобто вони визначають

сегмент або групу, як одиницю обробки (тобто завдання), яку можна обробляти незалежно [16].

1.1.2 Робочий процес потокового відео

Як у відео на вимогу (наприклад, Hulu, YouTube, Netflix), так і в прямому ефірі (наприклад, Livestreami), відеоконтент, створений камерами, має пройти через складний робочий процес перед відтворенням на пристроях глядачів.

На рисунку 1.2 представлено основні процеси, що виконуються для потокової передачі відео — від виробництва відео до відтворення відео на пристроях глядачів.

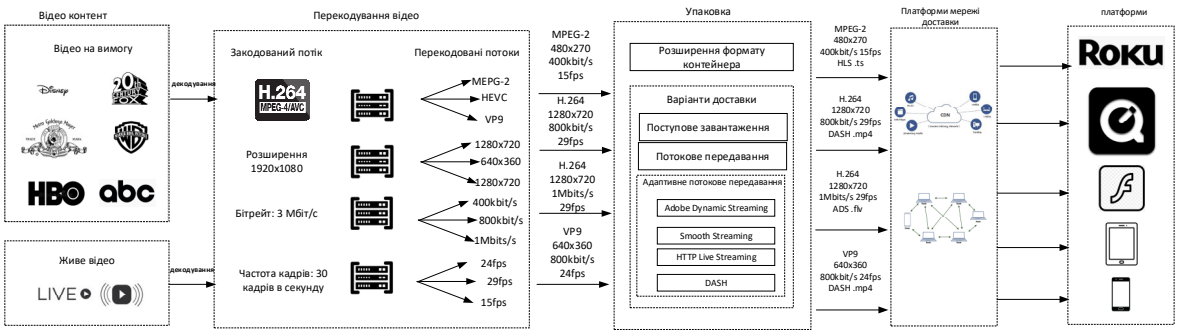


Рисунок 1.2 - Робочий процес, над відеопотоком від джерела до пристрою глядача.

Робочий процес включає процеси стиснення, перекодування, упаковки та доставки. Ці процеси разом дозволяють відтворювати необроблені та об’ємні відео, створені камерами, на різноманітних пристроях глядачів у режимі реального часу та з мінімальною затримкою. Варто зазначити, що на додаток до цих процесів, як правило, існують інші процеси для включення таких функцій, як захист відеовмісту та економічна ефективність потокового відео.

Першим кроком у потоковому відео є створення відеоконтенту. Необроблений відеоконтент, створений камерами, може займати значний

дисковий простір для зберігання, який неможливо передати через поточну швидкість Інтернету. Наприклад, одна секунда необробленого відео з роздільною здатністю 4K займає приблизно 1 ГБ пам’яті. Тому згенероване відео, по-перше, має бути стиснуте, що також відомо, як кодування відео.

Концепція відео полягає в безперервному показі великої кількості кадрів з певною частотою (вона ж частота кадрів), щоб створити рухливу сцену. Ця велика кількість кадрів зазвичай містить просторові та часові надмірності, як в середині кадру, так і між послідовними кадрами. У процесі стиснення відео ці надлишки видаляються за допомогою певного стандарту стиснення, наприклад, H.264/MPEG-4 AVC [18], VP9 [19] і H.265/HEVC [20]. Процес стиснення кодує необроблене відео на основі певного стандарту стиснення (відомого як кодек), роздільної здатності, бітрейту та частоти кадрів із значно меншим розміром.

Пристрій відображення глядача зазвичай може декодувати відео з певним форматом стиснення. Тому, щоб підтримувати гетерогенні пристрої відображення, відео, закодоване в певному форматі, має бути перетворено у різні формати. У процесі перекодування відео спочатку декодується, а потім кодується в інший формат стиснення. Таким чином, транскодування, як правило, є інтенсивним обчислювальним процесом.

Для потокової передачі перекодованого відео на пристрій глядача, відеофайл має бути структурований, щоб полегшити передачу та часову презентацію відеовмісту. Таким чином, перекодований відеофайл має бути упакований на основі певної структури, яка підтримується програвачем програми перегляду. Процес пакування є основою того, що широко відомо як формат відеофайлу (також званий форматом контейнера). Формат контейнера вводить заголовок, який містить інформацію про підтримувані потокові протоколи та правила надсилання сегментів відео. ISO Base Media File Format (ISOBMFF) [21], який є основою для формату MP4, і 3GPP TS [22], який є основою для формату ts, широко використовуються для процесу пакування.

Щоб доставити упаковані (тобто відформатовані) відеофайли через мережу, їм потрібно використовувати мережеві протоколи прикладного рівня (наприклад, HTTP [23], RTMP [24] і RTSP [25]). Існують різні методи доставки, які визначають спосіб отримання та відтворення відеопотоку. Прогресивне завантаження [26], HTTP Live Streaming (HLS) [27] і Dynamic Adaptive Streaming over HTTP (DASH) [28] є прикладами методів доставки. Кожен метод доставки базується на певному протоколі рівня мережевих програм. Наприклад, HLS і DASH працюють на основі HTTP, а Adobe Flash — на основі RTMP.

Після того, як відео запаковано, воно доставляється глядачам по всьому світу через мережу розповсюдження. Однак через затримку передачі в Інтернеті браузері, які розташовані далеко від відеосерверів і сховищ, страждають від тривалої затримки, щоб розпочати потокове передавання [29]. Щоб зменшити цю затримку, мережі доставки контенту (CDN) [30] використовуються для кешування відеоконтенту, який часто переглядають, у географічних місцях поблизу глядачів. Іншим варіантом зменшення затримки запуску є використання однорангових (тобто безсерверних) підходів, у яких кожен приймач (наприклад, комп’ютер глядача) діє як сервер і надсилає відеоконтент іншому одноранговому глядачу [31].

1.2 Проблеми та рішення потокового відео

На рисунку 1.3 представлено таксономію всіх проблем, які має вирішувати система потокового відео.

Таксономія показує різні типи потокового відео (наприклад, відео на вимогу і потокове передавання в прямому ефірі). Таксономія показує можливі шляхи розповсюдження відеоконтенту, такі як CDN та P2P .

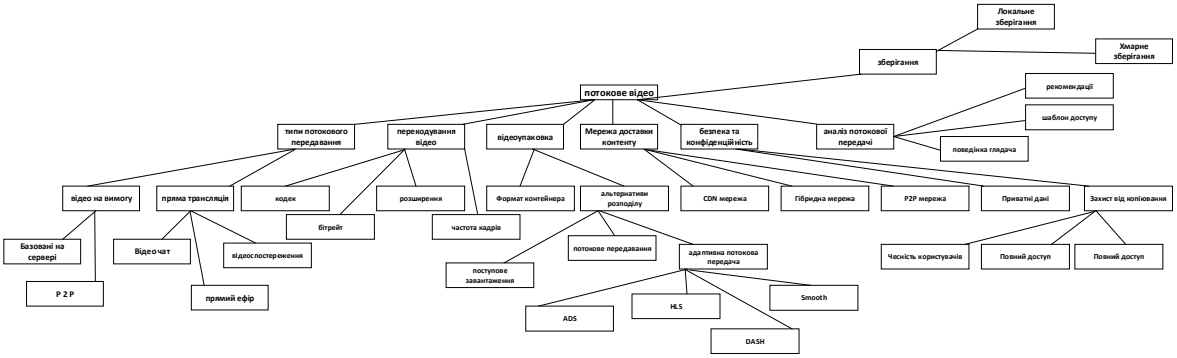


Рисунок 1.3 - Таксономія проблем в потоковому відео

Захист відеовмісту детально описується щодо конфіденційності відео для потокового відео та питань авторського права, відомого як керування цифровими правами (DRM). Таксономія охоплює аналітичні дослідження, які були проведені для розуміння поведінки глядачів і виявлення моделей їхнього доступу до відеопотоків. Також таксономія охоплює питання зберігання потокового відео та можливі стратегії керування репозиторієм потокового відео, як власне сховище та аутсорсингові рішення за допомогою хмарних служб зберігання.

1.2.1 Типи потокового відео

Таксономія, показана на рисунку 1.4, показує можливі способи надання послуг потокового відео глядачам.

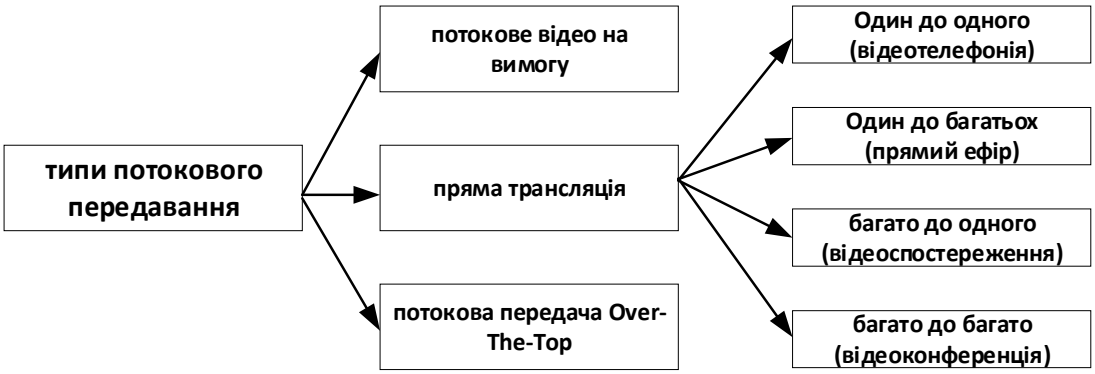


Рисунок 1.4 - Таксономія різних типів потокового відео.

Сервіс потокового відео можна реалізувати трьома основними способами: потокове відео за запитом, потокове передавання в прямому ефірі та потокове передавання з прямого ефіру на відео за запитом.

Транскодування відео є інтенсивним обчислювальним завданням і потребує значних обчислювальних ресурсів, в тому числі апаратних для його обробки. Тому перекодування відео зазвичай виконується в автономному режимі, що називається попереднім перекодуванням.

Швидкість передачі даних безпосередньо впливає на якість відео, оскільки вища швидкість забезпечує кращу якість, а вища швидкість споживає більше пропускної здатності мережі та обсягу пам’яті. Щоб передавати відео глядачам із різними умовами мережі, постачальники послуг потокового передавання зазвичай конвертують відео з кількома бітрейтами.

1.2.2 Упаковка відео

Щоб вибрати правильну технологію потокової передачі, потрібно розуміти переваги та недоліки протоколів потокової передачі та відеоупаковки.

Раніше, відео не можна було переглядати, доки воно не було повністю завантажено на пристрій. Поступове завантаження вирішило цю проблему, дозволивши відтворювати відео, щойно початковий буфер програвача заповнюється сегментами відео. Це скорочувало час очікування до 3–10 с для початку перегляду відео.

Недоліки прогресивного завантаження на основі HTTP викликали потребу в спеціальній технології потокового відео.

Щоб уникнути проблем прогресивного завантаження, був створений спеціальний протокол для потокової передачі в реальному часі (відомий як RTP). Цей протокол доставляв відеовміст із окремого потокового сервера. Хоча традиційні HTTP-сервери обробляють веб-запити, потокові сервери обробляють лише потокове передавання. У порівнянні з HTTP без збереження стану, протокол RTP враховує стан сеансу через це постійне з’єднання.

Завдяки постійному з’єднанню спеціальні протоколи потокової передачі дозволяють довільний доступ (наприклад, перемотування вперед) до потокового відео. Крім того, вони дозволяють адаптивну потокову передачу, за якої кілька закодованих відеопотоків можуть бути доставлені до різних програвачів на основі доступної ширини смуги та характеристик обробки. Потоковий сервер може контролювати вихідний потік, тому, якщо глядач припиняє перегляд відео, він припиняє надсилати відеопакет глядачеві

Щоб усунути обмеження попередніх потокових технологій, потокові рішення на основі HTTP повернулися на передній план адаптивної потокової передачі.

Адаптивне потокове передавання має такі переваги: По-перше, як і потоковий сервер, немає втрати мережі, оскільки відеоконтент доставляється на ходу. Тому один HTTP-сервер може ефективно обслуговувати кілька потоків. По-друге, потокове передавання на основі HTTP доставляється через протокол HTTP, що дозволяє уникнути проблем з брандмауером, з якими стикається RTMP. По-третє, це коштує дешевше, ніж використання потокового сервера. По-четверте, його можна швидко й ефективно масштабувати, щоб обслуговувати більше користувачів. По-п'яте, пошук всередині потоку більше не є проблемою. Коли глядач переміщує повзунок програвача, програвач просто отримує точні сегменти відео, а не все відео до потрібної точки.

1.2.3 Мережі доставки потокового відео

Метою технології CDN (Мережа розповсюдження контенту) є зменшення затримки мережі під час доступу до веб-вмісту. CDN мережа копіює вміст на територіально розподілені сервери, які знаходяться поблизу користувачів або глядачів. Враховуючи те, що для великого розміру відеовмісту зазвичай потрібно багато часу для передачі, використання Мережа доправлення контенту для кешування відеовмісту поблизу глядачів значно зменшує затримку. Netflix, як один із найбільших провайдерів потоку,

використовує трьох різних провайдерів CDN (Akamai, LimeLight і Level-3), щоб охопити всіх глядачів у різних регіонах. Перекодований і упакований відеоконтент відтворюється в усіх трьох CDN.

Передача потокового відеоконтенту через такі спеціальні мережі вивчалася в більш ранніх роботах. В роботі [32] запропонували архітектуру, яка мала абревіатуру PRISM для розповсюдження, зберігання та доставки високоякісного потокового контенту через IP-мережі. Автори в роботі [33] представили архітектуру для мобільного потокового CDN, яка була розроблена відповідно до вимог мобільності та масштабування.

1.2.4 Однорангові (P2P) мережі

Мережі P2P забезпечують прямий спільний доступ до обчислювальних ресурсів (наприклад, ресурсів ЦП, пам’яті та вмісту) між одноранговими вузлами в мережі. Мережі P2P розроблені для того, щоб і клієнти, і сервери діяли, як однорангові мережі. Вони можуть завантажувати дані з тих самих вузлів і завантажувати їх на інші вузли в мережі. Мережі P2P можуть поширювати файли даних серед користувачів протягом короткого періоду часу. Мережі P2P також широко використовуються для послуг потокового відео.

Потокове передавання P2P поділяється на два типи, а саме на основі дерева та на основі сітки. Структура P2P на основі дерева розподіляє відеопотоки, надсилаючи дані від однорангового вузла його дочірнім одноранговим вузлам. У структурі P2P на основі сітки однорангові вузли не дотримуються певної топології, натомість вони функціонують на основі вмісту та доступності пропускної здатності однорангових вузлів.

Основними перевагами P2P є низька вартість і гнучкість масштабованості, однак такий тип страждає від нестабільності якості обслуговування (QoS) . Тому багато розробників та науковців запропонували поєднати переваги як CDN, так і P2P в одній системі.

1.2.5 Безпека потокового відео

З повсюдним поширенням потокового відео на різноманітних пристроях відображення конфіденційність відеовмісту стала серйозною проблемою. Зокрема, живий вміст у формі відеоспостереження або потокової трансляції на основі користувача фіксує місця та записує багато небажаного/непов'язаного вмісту. Наприклад, людина, яка веде пряму трансляцію з вулиці, може ненавмисно зафіксувати номерний знак транспортних засобів, які проїжджають у цьому місці і можуть належати спецслужбам, чи взагалі громадянам які не давали згоди на відеозйому їхніх транспортних засобів. Таким чином, системи потокового відео можуть порушувати конфіденційність людей (наприклад, обличчя та теги транспортних засобів). Було розроблено різні технології для захисту конфіденційності прямої трансляції відеовмісту

В роботі [34] запропоновано дві методики приховування регіонів інтересів під час роботи систем відеоспостереження. Також в роботі [36] розроблено структуру для зберігання інформації про конфіденційність системи відеоспостереження у формі водяного знака.

Також вміст прямої трансляції відео зазвичай передається через бездротове середовище, яке можна легко перехопити та змінити. Крім того, DoS та DDoS -атаки можуть бути запущені на відеотрафік в режимі прямої трансляції.

1.3 Хмарне потокове відео

Після надходження запиту на потокове передавання потрібний відеопотік отримується «витягується» із серверів хмарного сховища, і над ними виконується робочий процес, описаний раніше, перед тим, як передати їх глядачам. Ці процеси зазвичай реалізуються у формі незалежних служб, відомих як мікросервіси, і, як правило, розгортаються у слабко зв'язаний спосіб на незалежних серверах у хмарних центрах обробки даних.

Прикладами мікросервісів, розгорнутих у центрах обробки даних для потокового відео, є такі служби, як веб-сервер, прийом відео, кодування, перекодування та упаковка. Задля надійності та відмовостійкості кожен із цих мікросервісів розгорнуто на кількох серверах що і утворює велику розподілену систему. Балансувальник навантаження розгортається для серверів кожного мікросервісу. Балансувальник навантаження призначає потокові завдання відповідному серверу з метою мінімізації затримки та покриття можливих збоїв на серверах.

Вищезазначені мікросервіси зазвичай реалізуються за допомогою контейнерних технологій, з використанням парадигми безсерверних обчислень. Контейнери Docker масштабуються набагато швидше, ніж віртуальні машини, і мають швидший час запуску (завантаження), що дає їм перевагу перед віртуальними машинами в обслуговуванні змінних вимог до потокового відео. Крім того, контейнери Docker використовуються для упаковки відео, обробки потокових запитів, що надходять і відомі, як «прийом запитів», і також для вставки реклами в відеопотоки або між ними.

Sprocket — це безсерверна система, реалізована на базі AWS Lambda і дозволяє розробникам програмувати серію операцій над відеовмістом у модульному та розширюваному порядку. Ще одна перевага розгортання безсерверної обчислювальної парадигми (функція/сервіс як послуга), такої як AWS Lambda, для потокового відео полягає в тому, щоб позбавити постачальників потоку від проблем з плануванням, балансуванням навантаження, наданням ресурсів і моніторингом ресурсів.

Окрім типу обчислювальної платформи (наприклад, на основі віртуальних машин і контейнерів) для потокового відео, тип машин, призначених для обробки потокових завдань, також впливає на затримку та понесені витрати на потокове передавання. Наприклад, хмари пропонують різні типи віртуальних машин із різноманітними конфігураціями (наприклад, GPU, CPU, IO та Memory) або різні типи резервування (наприклад, резервування на вимогу, сповільнене та попереднє резервування).

1.3.1 Хмарне перекодування відео

Щоб охопити глядачів гетерогенними пристроями відображення, які працюють із різними кодеками, роздільною здатністю, бітрейтом і частотою кадрів, відеоконтент зазвичай потрібно перекодувати в кілька форматів. Процес транскодування відео потребує значної обчислювальної потужності та часу в усьому робочому процесі потокового відео. Методи та проблеми транскодування відео були вивчені в роботах [35] [36].

У минулому постачальникам потокових послуг доводилося обслуговувати великі обчислювальні системи (тобто власні центри обробки даних), щоб досягти перекодування відео. Однак через оновлення та витрати на технічне обслуговування внутрішніх центрів обробки даних багато провайдерів потокових послуг вирішили передати свої операції перекодування хмарним серверам.

Для ефективного використання хмарних служб вкрай важливо усвідомлювати різницю в продуктивності операцій перекодування відео на різних типах віртуальних машин. В роботі [37] проведено оцінку продуктивності різних операцій транскодування на гетерогенних віртуальних машинах, щоб дослідити ключові фактори часу обробки транскодування.

Щоб ще більше скоротити витрати на використання хмарних служб для перекодування відео на вимогу, в роботах [7, 9] пропонують хмарний механізм потокового відео (CVSE), який працює на рівні відеосховища та попередньо транскодує лише гарячі відео, тоді як повторно транскодує рідко доступні відео за запитом. Механізм здатний забезпечити низьку затримку запуску та «тремтіння» відтворення за допомогою належного планування та політики надання ресурсів.

Хмарне перекодування відео також широко використовується в потоковому ефірі [12]. В дослідженні [38] запропоновано механізм транскодування в режимі реального часу з підтримкою хмари, який заснований на протоколі HLS.

1.3.2 Пакування відео на основі хмари

Пакування відео є біль легшим в обчислювальному плані, ніж транскодування відео, отже, це вигідно та більш реально обробляти таке відео на льоту, використовуючи хмарні служби (VM або контейнери). Для потокового передавання відео на вимогу, відеовміст зазвичай попередньо транскодується в різні типи відтворення (формати), а потім кожне відтворення упаковується в різні версії відповідно до різних вимог протоколу потокового передавання (наприклад, DASH, HLS, Smooth).

Замість статичного пакування транскодованих відео в різні відтворення протоколів і зберігання їх у сховищі (тобто попереднього пакування), динамічне пакування відео пакує відеосегменти на основі підтримуваних пристроєм протоколів. Весь процес займає лише мілісекунди, і користувачі/глядачі зазвичай цього не помічають, однак це економить значну вартість зберігання для постачальників потокового відео.

Через легкий характер процесу пакування контейнерні послуги зазвичай використовуються для їх реалізації в хмарних центрах обробки даних

1.3.3 Хмарне сховище для потокового відео

Швидке зростання використання потокового відео в різних програмах, таких як електронне навчання, відеоспостереження та ситуаційна обізнаність, а також на різних формах мобільних пристроїв (наприклад, смартфонах, планшетах, ноутбуках) створило проблему великих відеоданих. Швидкий ріст відеовмісту в Інтернеті вимагає величезних сховищ. Хмарні служби зберігання надають рішення для масштабованості, надійності та відмовостійкості. Таким чином, основні постачальники потокових послуг (наприклад, Netflix і Hulu) повністю покладаються на хмарні служби зберігання для своїх вимог до зберігання відео.

1.4 Вибір перспективного шляху і постановка задачі дослідження

Наразі, щоб підтримувати високоякісну пряму трансляцію на різних пристроях відображення, видавцям відео доводиться генерувати кілька версій (тобто форматів) того самого відео (наприклад, кілька кодувань відео) самостійно. Однак цей підхід страждає від апаратних обмежень і вузького місця пропускної здатності мережі. Крім того, цей підхід не враховує запити глядачів. Тобто він потенційно може генерувати версії, які не запитуються глядачами. Таким чином, цей підхід залишається занадто дорогим і неефективним.

Іншим підходом є перекодування потокового відео в прямому ефірі за запитом. У такому підході видавець створює лише одну версію відео. Відео в прямому ефірі можна транслювати глядачам, якщо їхні пристрої відображення сумісні з форматом потокової передачі. Перекодування відео в прямому ефірі на вимогу відбувається після приєднання нового засобу перегляду з несумісним форматом пристрою відображення.

Транскодування відео є обчислювально важким і трудомістким процесом, він потребує величезної пам’яті та обчислювальної інфраструктури. Внутрішня модернізація такої інфраструктури для задоволення швидко зростаючих глобальних потреб у транскодуванні відео є непомірно високою. Тому використання хмарних сервісів стає звичайною практикою серед провайдерів потокових послуг

Тому, розробка моделі хмарної передачі відео в реальному часі, що зменшує витрати і обчислювальну складність витрачену на потокове відео є актуальною задачею.

Метою роботи є модель хмарної передачі відео в реальному часі. Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Вивчення теоретичних основ побудови систем потокового відео.
2. Проаналізувати існуючі системи та підходи організації потокового відео.

3. Розробити метод для хмарної трансляції відео в реальному часі.
4. Розробити модель потоку джерела відео та потоку клієнта.
5. Розробити QOS-Aware метод планування
6. Розробити архітектуру VLSC
7. Розробити архітектуру клієнта системи
8. Провести аналіз відео та оцінку результатів

Висновки до розділу 1

1. Розглянуто комплексний підхід до організації потокового відео, що включає всі аспекти його обробки: стиснення, перекодування, упаковку, передачу та зберігання. Основна увага була зосереджена на можливості використання хмарних сервісів як інструменту для оптимізації цих процесів, що дає змогу зменшити витрати та підвищити якість обслуговування глядачів/користувачів.

2. Проаналізовано структуру відеопотоку та ключові аспекти його побудови, зокрема рівні обробки: сегменти, групи зображень, кадри, фрагменти та макроблоки. Це дозволяє ефективно визначити оптимальний рівень обробки та мінімізувати вузькі місця у процесі передачі відео. Крім того в розділі було приділено увагу такому питанню, як адаптивність технологій, яка дозволяє підлаштовувати якість відео до умов мережі, забезпечуючи доступність контенту на різних пристроях.

3. Під час аналізу робіт та використання технологій, виявилось, що однією з головних проблем є оптимізація витрат на хмарні обчислення. Методи динамічного пакування та хмарного перекодування дозволяють зменшити навантаження на інфраструктуру постачальників відео та скоротити час передачі. Використання контейнерних технологій, таких як Docker, підвищує ефективність масштабування сервісів. Крім того, впровадження однорангових мереж (P2P) у поєднанні з CDN забезпечує гнучкість і зниження затримок у передачі.

4. Крім було розглянуто питання безпеки потокового відео також є важливим аспектом. Використання технологій захисту, таких як керування цифровими правами (DRM) та шифрування, дозволяє зберегти конфіденційність контенту.

5. Тому розробка хмарної моделі передачі відео в реальному часі є актуальним завданням, адже воно спрямоване на підвищення ефективності потокових сервісів, забезпечення їхньої масштабованості та економічної доцільності. Запропонована модель має стати основою для впровадження нових інноваційних рішень у сфері мультимедіа та кіберінфраструктури, що відкриває широкі перспективи для розвитку галузі.

2 МОДЕЛЬ ХМАРНОЇ ПЕРЕДАЧІ ВІДЕО

2.1 Хмарна трансляція відео в реальному часі

Cloud Live Video Streaming (CLVS), або іншими словами це метод коли трансляція відео використовує мережу серверів, які розміщують і передають відео. Такий підхід дуже ефективний для потокового відео в реальному часі, який створює окрему модель ціноутворення від сучасних сервісів потокового відео. Ключовим компонентом CLVS є служба Amazon Simple Storage Service (S3), яка використовується для зберігання сегментів відео та метаданих. Використовуючи S3, CLVS використовує те, що називається «безсерверним» дизайном, усуваючи необхідність передавати відео через проміжний сервер. CLVS також усуває потребу в облікових записах третіх сторін і ліцензійних угодах. Далі буде описано прототип CLVS, оскільки потокове відео в прямому ефірі стає все більш поширеним, альтернативні та економічно ефективні рішення є важливими.

Вартість використання CLVS використовує аналогічну модель ціноутворення, але полегшує ліцензійні угоди. У багатьох випадках ціна на CLVS набагато нижча, ніж на використання сучасних служб потокового відео в прямому ефірі, але забезпечує порівнянну продуктивність. Перспективи CLVS дуже багатообіцяючі на сучасному ринку.

CLVS використовує деякі основні методи потокового відео, але має деякі ключові відмінності. Основною розробкою CLVS є безсерверна архітектура. Замість використання проміжних серверів CLVS використовує S3 як спільний простір для передачі даних. Щоб отримати доступ до потоку, глядач використовує відповідну назву сегмента та ключ, а також дійсні облікові дані, після чого відображається останній сегмент відео. Ця техніка має багато переваг, включаючи масштабованість, вартість, доступ до додаткових хмарних ресурсів, таких як мережі доставки контенту (CDN), і безпеку. Такий дизайн спрощує сучасну практику потокового відео, усуває

потребу у виділених серверах і є чудовою альтернативою потоковому відео в реальному часі.

2.2 Пропонований підхід для хмарної трансляції відео в реальному часі

Реалізація CLVS дотримується подібних принципів, розглянутих раніше. Джерело відео записує подію, кожен кадр стискається, фрагмент стиснутих кадрів надсилається в хмарне сховище, клієнт отримує кожен фрагмент із хмарного сховища та переглядає відео. Рисунок 2.1 ілюструє подібність між моделлю CLVS і сучасним потоковим відео, тоді як рисунок 2.2 показує ключові операції в CLVS.

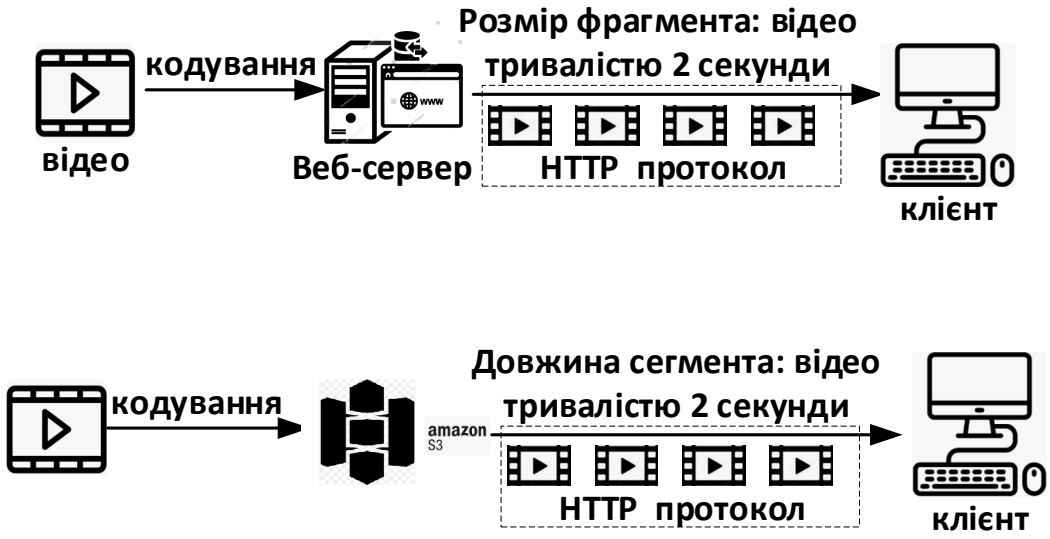


Рисунок 2.1- Основи CLVS

Це рішення також відрізняється від деяких звичайних застосувань. CLVS використовує дуже ефективний дизайн із застосуванням методів паралельного програмування, попередньої вибірки та споживчої моделі виробника.

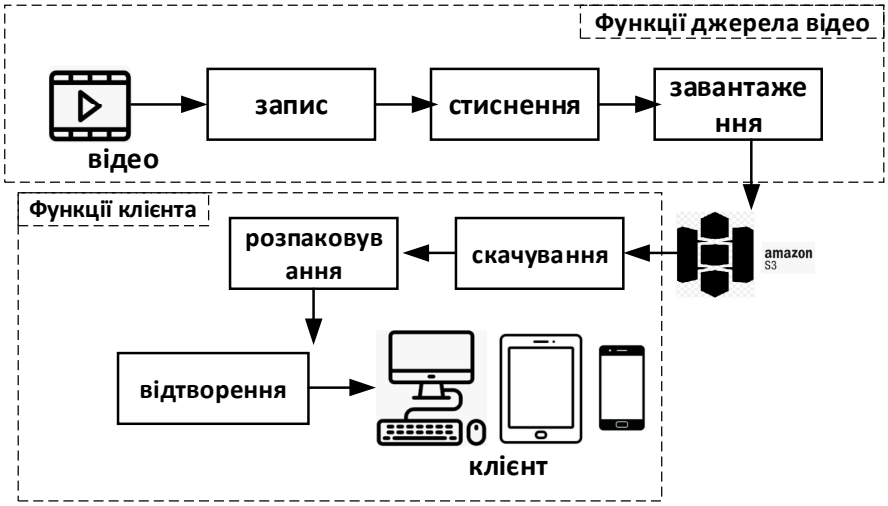


Рисунок 2.2-Загальна концепція CLVS

2.3 Метод планування на базі якості обслуговування

Архітектура планувальника перекодування проілюстрована на рисунку 2.3. Для планування групи запитаних відеопотоків вони групуються в чергу після надходження. Щоб мінімізувати затримку запуску відеопотоків, було вибрано інший підхід до побудови черги, яка називається чергою запуску. Перші кілька груп зображень кожного нового відеопотоку розміщуються в черзі запуску, яка має вищий пріоритет порівняно з пакетною чергою. Щоб уникнути будь-якої затримки виконання, кожній віртуальній машині виділяється локальна черга, у яку попередньо завантажуються необхідні дані для GOP перед початком виконання транскодування групи зображень.

Для кожного GOP j із відеопотоку i, позначеного G_{ij}, доступні час прибуття та кінцевий термін (позначений δ_{ij}).

У потоковому відео в реальному часі планувальник не знає про час виконання G_{ij} завдань перекодування. Однак час виконання групи зображень можна передбачити, припускається, що доступна оцінка часу виконання кожної групи зображень (GOP) G_{ij} який називається часом перекодування та позначається τ_{ij}.

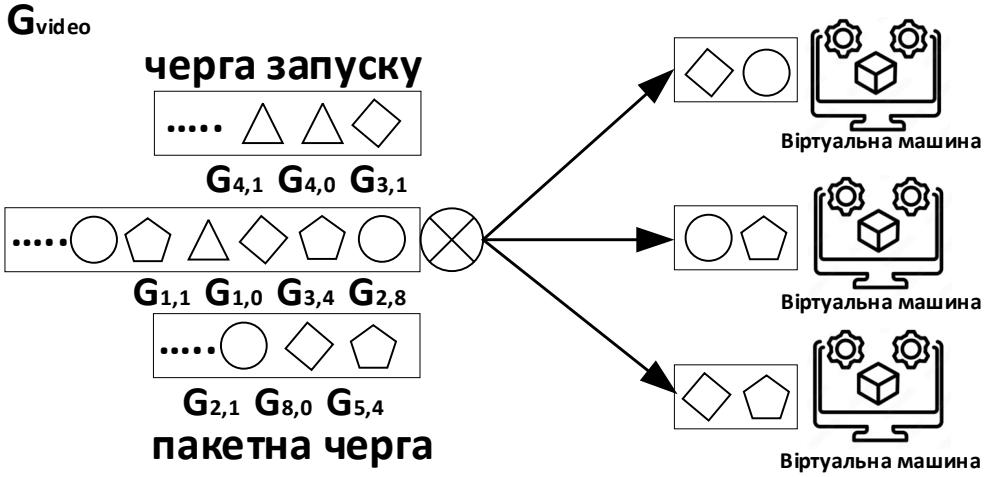


Рисунок 2.3 - Архітектура планування з урахуванням якості обслуговування у VLSC.

Перші кілька груп зображень кожного потоку ставляться в чергу запуску, а решта розміщуються в пакеті після надходження. Кожній віртуальній машині для перекодування виділяється локальна черга для попереднього завантаження GOP перед початком їх виконання.

Щойно вільне місце з'являється в локальній черзі віртуальної машини, виконується планувальник, щоб зіставити GOP із вільним місцем. Планувальник зіставляє GOP з віртуальною машиною, яка забезпечує найкоротший час завершення.

Загалом, щоб оцінити час завершення прибулої групи зображень G_x на VM_j, можна визначити очікуваний час виконання поточного GOP у VM_j, що залишився, з розрахунковим часом виконання всіх завдань попереду G_x у локальній черзі VM_j. Після чого додається розрахунковий час виконання G_x (тобто τ_x). Тому нехай t_r — очікуваний час виконання поточного завдання на VM_j, що залишився, і t_c — поточний час. Тоді можна оцінити час виконання завдання для G_x (позначається як φ_x) наступним чином:

$$\varphi_x = t_c + t_r + \sum_{p=1}^{n} \tau_p + \tau_x \quad (1)$$

де, τ_p позначає розрахунковий час виконання будь-якого завдання, що очікує перед G_x у локальній черзі VM_j,;

n - кількість завдань, що очікують у локальній черзі VM_j.

Слід також враховувати, що в прямому ефірі завдання транскодування мають жорсткий термін. Тобто, якщо розрахунковий мінімальний час завершення GOP φ_x перевищує кінцевий термін подання δ_{ij}, тоді GOP відкидається, а планувальник переходить до наступного GOP у черзі.

У запропонованому методі планування позначається вищий пріоритет завданням групи зображень у черзі запуску. Однак пріоритет не повинен спричиняти пропуски кінцевих термінів завдань, що очікують у пакетній черзі.

Нехай G_b, перша група зображень у пакетній черзі, і G_s, перша GOP у черзі запуску. Під час кожної події планування G_s можна запланувати перед G_b, лише якщо це не призведе до пропуску G_b свого кінцевого терміну. Для цього потрібно обчислити мінімальний час завершення G для всіх віртуальних машин. Потім потрібно обчислити мінімальний час завершення G_b, припускаючи, що G_s уже зіставлено з віртуальною машиною, і нарешті перевіряється, чи G_b пропустить свій кінцевий термін чи ні. Якщо ні, то G_s можна запланувати перед G_b.

Ефективність запропонованого методу планування також залежить від політики черги пакетної черги. Можна використовувати будь-яку звичайну політику черги (наприклад, FCFS, SDF або SJF), щоб визначити пріоритет завдань у пакетній черзі.

2.2.1 Модель потоку джерела відео

Записуючий пристрій, який використовується для захоплення та надсилання відеоданих, називається джерелом відео. CLVS використовує Raspberry Pi 3 із 5-мегапіксельним модулем камери, який забезпечує помірно потужну обчислювальну потужність, високу мережеву здатність, широкі можливості налаштування та економічну ефективність. Незважаючи на вищі характеристики, записане відео використовує мінімальні вимоги – без

кольору, без звуку та низьких кадрів за секунду (FPS). Підтримка примітивної якості відео забезпечує просте впровадження та тестування. Джерело відео має багато кроків, зокрема: підготовка, стиснення, сегментація, транспортування сегментів до S3, видалення старих сегментів із S3 та запис даних продуктивності.

Джерело відео використовує три основні потоки для виконання всіх вищезазначених операцій, а саме основний потік, потік завантаження та потік видалення (рисунок 2.4).

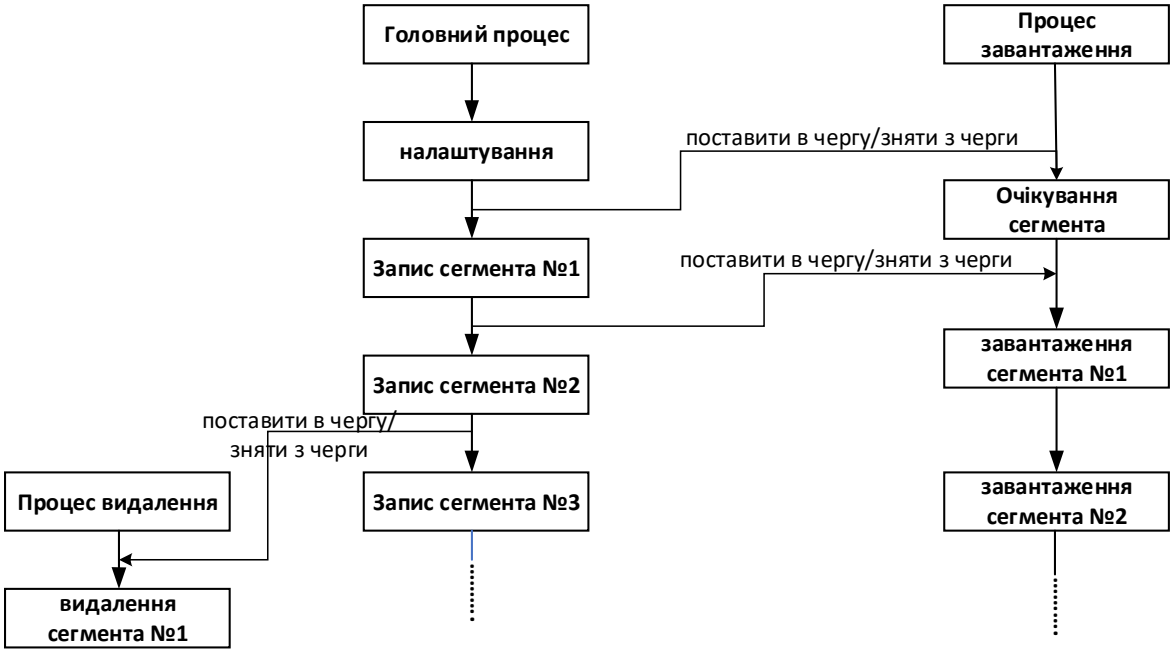


Рисунок 2.4 - Модель потоку джерела відео

Основний потік ініціалізує параметри налаштування, такі як FPS, довжина сегмента та час початку. Потім два інших потоки запускаються зі спільними ресурсами. Після налаштування основний потік продовжує записувати відео, передавати сегменти в потік завантаження та сигналізувати про видалення. Потік завантаження використовується для завантаження всіх записаних сегментів до Amazon S3. Потік завантаження підтримує пряме з'єднання з сегментом Amazon S3 і спілкується з потоком видалення для надсилання запитів на видалення до Amazon S3. Нарешті, потік видалення

використовується для видалення старих сегментів на основі параметра max segments. Якщо сегмент Amazon S3 містить максимальну кількість сегментів, потік видалення видаляє найстаріший сегмент під час завантаження найновішого. Усі потоки спілкуються через кілька черг.

2.2.2 Стиснення

Ефективним способом стиснення відео є міжкадрове стиснення. Однак недоліком використання бібліотек з відкритим кодом є наявність різних алгоритмів стиснення. Однак більш ефективно буде зберігати стислий сегмент в основну пам’ять, а потім безпосередньо передавати його на Amazon S3.

Крім того, кожен відеосегмент повинен бути записаний на диск, перш ніж його можна буде прочитати з відеопрогравача. Тому CLVS не використовує розширене міжкадрове стиснення. CLVS використовує внутрішньокадрове стиснення, щоб уникнути кількох операцій введення-виведення. Використовується вбудована бібліотека стиснення Java, де кожен кадр стискається за допомогою кодування JPEG. Хоча кодування JPEG не таке ефективне, як міжкадрове стиснення, все ж забезпечує середнє скорочення даних 11,5:1.

Використовуючи внутрішньокадрове стиснення, можна уникнути накладних витрат і затримки файлового введення-виведення. Зосереджуючись на швидкості потоку, а не на якості зображення, потенціал CLVS можна оцінити без використання оптимального алгоритму стиснення.

2.2.3 Сегментація

Кожен записаний відеосегмент базується на параметрах налаштування, таких як: час початку, довжина сегмента та FPS. Час початку – це позначка часу, коли джерело відео почало записувати. Довжина сегмента – це кількість секунд, протягом яких відтворюватиметься кожен сегмент. FPS – це кількість кадрів, які з’являться протягом секунди перегляду. Таким чином, кожен сегмент міститиме кадри довжиною сегмента⇽ FPS.

Для кожного сегмента відео існує об’єкт заголовка та об’єкт даних, як показано на рисунку 2.5.

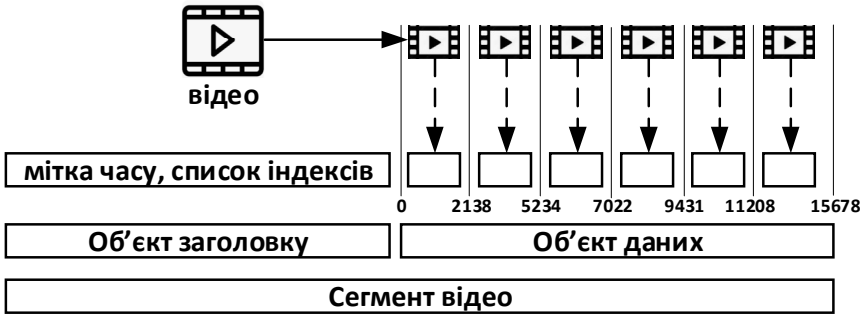


Рисунок 2.5 - Структура сегмента відео

Об'єкт даних містить буфер, який містить усі записані стиснуті кадри. Об’єкт заголовка містить як буфер індексів, який називається списком індексів, так і мітку часу. Список індексів містить позиції індексів кожного кадру в об’єкті даних. Мітка часу використовується для отримання поточної тривалості записаної події. Він записується до того, як перший кадр буде збережено в буфері даних, і віднімає час початку.

Максимальний індекс відео та максимальні сегменти використовуються для визначення метаданих сегментів. Коли відеосегмент записується, йому дається ім’я на основі індексу (тобто myvideo0, myvideo1, myvideo2 тощо). Максимальний індекс відео визначає максимальний ліміт індексу, після досягнення якого поточний індекс скидається до нуля. Максимальна кількість сегментів визначає обмеження сегментів, які можна зберігати в Amazon S3 одночасно. Це контролює простір, який займає відеоконтент у S3.

2.2.4 Хмарне зберігання

Хмарне зберігання, використовується для зберігання всіх записаних відеосегментів. Для використання Amazon S3 як відправник, так і одержувач повинні надати дійсні облікові дані. Ці облікові дані надає Amazon у вигляді файлу, що містить як ім'я користувача, так і секретний ключ. Перш ніж

отримати доступ до бакета, користувач і ключ повинні бути перевірені через Amazon. Це додає рівень безпеки, щоб запобігти несанкціонованому доступу до файлів користувача.

Будь-який файл, збережений у S3, називається об'єктом, а назва об'єкта називається ключем. Перед завантаженням сегмента в бакет S3 йому потрібно призначити ключ. У цьому випадку ключем є назва сегмента. Після завершення всіх записів всі сегменти видаляються з S3, хоча вони можуть бути збережені для подальшої обробки. Єдиний інший файл, який завантажується в бакет, це файл налаштувань, який додається перед завантаженням будь-яких сегментів. Кількість сегментів, які можуть перебувати в бакеті одночасно, залежить від параметра максимальної кількості сегментів.

CLVS використовує клас TransferManager з API S3, який керує декількома потоками і з'єднаннями для прискорення передачі великих файлів при наявності високої пропускної здатності. Основою S3 є використання HTTP, TCP та TLS 1.2 для передачі даних. HTTP і TCP розроблені для надійної передачі файлів. Крім того, ці протоколи наразі використовуються для потокового відео. Хоча UDP є швидшим, TCP надає переваги в контролі перевантаження, а також кращі можливості попереднього завантаження і буферизації.

Багато платформ потокового відтворення налаштовані на підтримку вебсервісів, включаючи мережі доставки контенту (CDN). Тому HTTP часто використовується для доставки відео. Ще одна перевага HTTP і TCP полягає в тому, що вони широко використовуються в Інтернеті, тому не блокуються міжмережевими екранами. Протокол TLS використовується для шифрування даних і запобігання доступу з боку зловмисників.

Висновки до розділу 2

1. Запропонована модель хмарної передачі відео в реальному часі, Cloud Live Video Streaming (CLVS), демонструє покращення у підходах до трансляції мультимедійного контенту. Основна ідея використання безсерверної архітектури дозволяє ефективно усунути потребу у виділених проміжних серверах, спрощуючи процес і знижуючи витрати. Використання технології Amazon S3 як центрального елемента зберігання та передачі даних підвищує масштабованість і безпеку системи, а також дозволяє адаптувати потоки до потреб користувачів у реальному часі.

2. Описаний підхід включає в себе реалізацію паралельного програмування, попереднього завантаження та моделі «виробник-споживач», що значно оптимізує процеси обробки та доставки відео. Використання ефективного стиснення, сегментації та зберігання сегментів у хмарі створює умови для швидкої та надійної передачі контенту з мінімальними затримками. Зокрема, застосування внутрішньокадрового стиснення знижує затримки, спричинені файловими операціями, зберігаючи при цьому прийнятну якість зображення.

3. Інтеграція хмарного зберігання з API Amazon S3 забезпечує високий рівень безпеки передачі даних завдяки використанню протоколів TLS, HTTP і TCP. Це дозволяє уникнути блокування міжмережевими екранами та забезпечує надійність передачі навіть у складних мережевих умовах. Крім того, технології, що базуються на вебсервісах, підтримують інтеграцію з CDN, знижують затримки доступу до контенту для кінцевих користувачів.

4. Реалізація планування задач на основі якості обслуговування (QoS) дозволяє оптимізувати розподіл обчислювальних ресурсів, що особливо важливо для реального часу. Запропонований метод пріоритетів для груп зображень у черзі запуску гарантує мінімальні затримки при виконанні найважливіших задач, забезпечуючи при цьому баланс між продуктивністю системи та дотриманням кінцевих термінів.

5. Модель CLVS пропонує новий підхід до організації потокового відео, який є економічно вигідним, технологічно простим і є високо адаптивним. Така модель відповідає сучасним тенденціям мультимедійних технологій та забезпечує перспективи для подальшого розвитку потокового відео, інтегрованого з хмарними сервісами, задовольняючи запити як користувачів, так і постачальників контенту.

3 РЕАЛІЗАЦІЯ МОДЕЛІ ХМАРНОЇ ПЕРЕДАЧІ ВІДЕО В РЕАЛЬНОМУ ЧАСІ

3.1 Архітектура системи трансляції відео в реальному часі за допомогою хмарних сервісів

На основі другого та першого розділів пропонується наступна архітектура для прямої трансляції з використанням хмарних сервісів. Огляд архітектури представлено на рисунку 3.1. Дана архітектура показує послідовність дій, які виконуються, коли глядачі запитують відео від постачальника послуг потокового передавання. Архітектура включає шість основних компонентів, а саме відеорозподільник, оцінювач часу, планувальник завдань (тобто групи зображень), віртуальні машини транскодування (VM), диспетчер віртуальних кластерів (VCM) і злиття відео. Комбінація цих компонентів веде до економічно ефективного перекодування потоків в реальному часі у хмарі з урахуванням якості обслуговування.

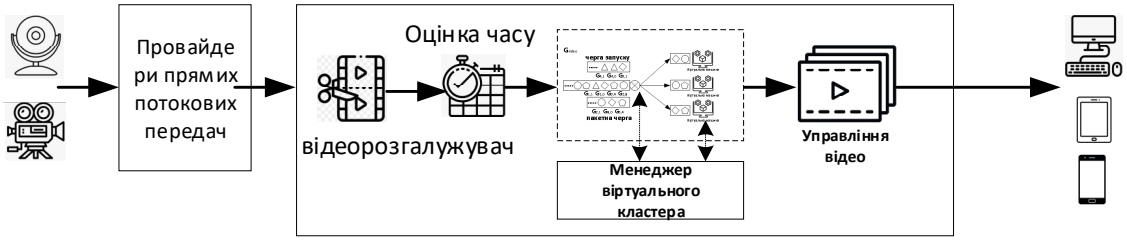


Рисунок 3.1 - Хмарна архітектура транскодування потокової трансляції

3.1.1 Відеоспліттер

У компоненті розділювача відео відеопотікd в реальному часі після надходження розбивається на кілька груп зображень, які потім можна перекодувати незалежно. Кожна група зображень має окремий жорсткий термін на основі часу представлення першого кадру в цій групі. Оскільки в даній роботі розглядається тільки випадок прямої трансляції, якщо група

зображень пропускає свій крайній термін, перекодування такої групи не має значення, і його потрібно видалити.

3.1.2 Оцінка часу виконання

У прямому ефірі час виконання завдань групи зображень, що надходять, невідомий. Варто зазначити, що це основна відмінність прямого ефіру від відео на вимогу (VOD), де відеопотік обробляється кілька разів. Таким чином, час перекодування (тобто виконання) кожного завдання групи зображень можна оцінити на основі попередньої інформації про виконання. Така інформація про попереднє виконання не існує в потоковому відео в реальному часі, оскільки це перший раз, коли відео транслюється.

Хоча попередня інформація про виконання групи зображень не існує в потоковому ефірі, результати експерименту демонстрували, що час виконання GOP корелює з часом виконання інших груп зображень у тому самому відео. Фактично, згенеровані групи зображень у відео включають подібну кількість кадрів, отже, подібний час виконання. Таким чином, у потоковому відео в реальному часі час виконання певного завдання GOP можна оцінити на основі інформації про час виконання попередніх перекодованих GOP у тому самому відеопотоці.

Моделювання часу перекодування попередніх груп зображень у відеопотоці має нормальний розподіл $N_{(\mu,\sigma)}$, а час транскодування групи зображень також корелює з розміром даних, тобто кадрів. Отже, для заданого GOPn із розміром S_n оцінений час транскодування, позначений τ_n, обчислюється наступним чином.

$$\tau_n = \frac{S_n}{S\text{ср}} \cdot \mu + \sigma \qquad (2)$$

Варто зазначити, що додавши σ, в рівняння забезпечить найгіршу оцінку для часу перекодування GOPn. Також слід зазначити, що S_n відомий до виконання GOPn.

3.1.3 Планувальник завдань перекодування групи зображень.

Планувальник завдань перекодування (коротко називається планувальником перекодування) відповідає за відображення завдань групи зображень на серверах перекодування. Основна задача планувальника полягає в тому, щоб задовольнити вимоги QoS клієнтів (щодо мінімальної затримки запуску та швидкості подання відеопотоків) без додаткових витрат для постачальника потоків.

3.1.4 Перекодування на рівні віртуальної машини (VM)

Віртуальні машини виділяються від хмарного постачальника для обробки завдань групи зображень. Припускається, що виділені віртуальні машини є однорідними. Кожна віртуальна машина має локальну чергу, де необхідні дані для груп зображень попередньо завантажуються перед виконанням. Коли в локальній черзі віртуальної машини з'являється вільне місце, планувальник отримує сповіщення про зіставлення групи зображень з віртуальною машиною. Також за замовчуванням, завдання групи зображень у локальній черзі плануються за допомогою методу FCFS.

3.1.5 Менеджер віртуального кластера (VCM)

Менеджер віртуального кластера відстежує роботу транскодування віртуальних машин в архітектурі VLSC і змінює розмір кластера віртуальної машини, щоб відповідати вимогам клієнта щодо QoS і мінімізувати понесені витрати постачальника послуг потокового передавання. В даній роботі розглядався випадок використання політики надання статичних ресурсів (тобто фіксовану кількість віртуальних машин). Реалізація динамічного надання ресурсів не розглядалась.

3.1.6 Злиття відео

Метою об'єднання відео є розміщення всіх перекодованих груп зображень у правильному порядку та створення прямого потоку. Об'єднання відео надсилає перекодовані прямі трансляції глядачам.

3.2 Модель потоку клієнта

Клієнт (рисунок 3.2) витягує сегменти з S3 і переглядає відеопотік. У цьому процесі використовуються три потоки: основний потік, потік завантажувача та дисплей. Головний потік ініціалізує інші два потоки та комунікує із завантажувачем через дві черги. Черга сигналів використовується для передачі інформації про налаштування від завантажувача до основного потоку, тоді як черга сегментів використовується для надсилання сегментів відео із завантажувача до основного потоку.

Основний потік використовує інформацію у файлі налаштування, щоб знати, скільки кадрів міститься у сегменті відео та час початку запису. Завантажувач використовує параметри максимального індексу відео та максимальних сегментів, щоб знати, який індекс відео буде найновішим.

Процес перегляду відео досить простий. Передача відеосегментів із завантажувача в основний потік є проектом виробника-споживача. Потік завантажувача завантажує файл налаштування та надсилає цю інформацію до основного потоку через чергу сигналів. Після цього завантажувач знаходить найновіше відео та поміщає його в чергу сегментів. Потім основний потік захоплює сегмент відео з черги сегментів і використовує заголовок у кожному сегменті для пошуку кожного кадру.

Відображувач захоплює кадр і відображає його у вікні. Основний потік чекає 1/FPS секунд і захоплює наступний кадр. Попередня вибірка дозволяє завантажувачу захопити наступний відеосегмент, у той час як основний потік завершує циклічний перегляд попереднього.

Якщо в черзі сегментів існує більше одного сегмента, старіші сегменти відкидаються, щоб зменшити затримку та дозволити глядачеві отримати найновіший сегмент. Цей процес триває, поки користувач не вийде з програми.

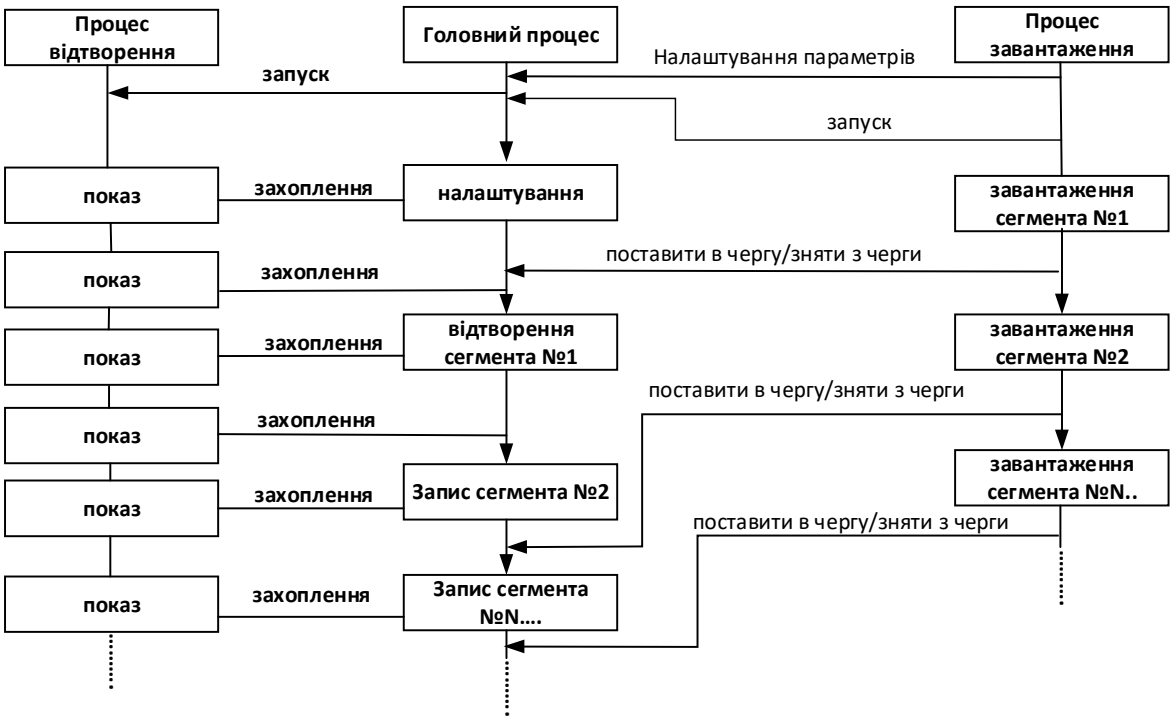


Рисунок 3.2 - Модель потоку клієнта

3.3 Аналіз відео за допомогою Kafka та Spark

Виконувати масштабну обробку даних у реальному часі в хмарі дуже важко через обчислювальну складність під час виконання аналізу даних. Дані в розподіленій мережі передаються з дуже високою швидкістю. Для аналізу відео було обрано розподілене сховище подій Kafka, оскільки воно забезпечує швидку, стійку до збоїв, масштабовану систему обміну повідомленнями та часто використовується в архітектурах потокового передавання даних у реальному часі, що забезпечує аналітику в реальному часі. Нижче наведено причини використання Kafka для аналізу відео:

- Розподілене середовище з архітектурою видавця та передплатника.
- Забезпечте зберігання до семи днів.
- Отримувати повідомлення в порядку з різними розділами з тією самою темою (підтримка строгого порядку).
- Підтримка кількох вузлів потоків для доставки даних кількох споживачів.

– Журнал фіксації для отримання даних після збою.

– Висока пропускна здатність.

– Підтримка великомасштабних даних.

– Розширене використання мікросервісів, керованих подіями, зберігання журналів, потокове передавання, джерело подій, збирання змінених даних (CDC), конвеєр корпоративних даних.

Також наліз відео потребує безперервної передачі великих обсягів даних для обчислень у хмарі. Існує багато фреймворків глибокого навчання, таких як Cafe, TensorFlow, Chainer тощоде відбувається послідовний процес, у якому клієнт отримує дані від Kafka за допомогою Spark, виконує свою бібліотеку машинного навчання та програму python, а пізніше викликається Chainer.

Дані зберігаються у формі стійкого розподіленого набору даних (RDD) у Spark. Конфігурація Kafka поділена на два підрозділи, перший – це один вузол, а другий – конфігурація з кількома вузлами, як зазначено нижче:

– Один вузол: у цьому випадку один потік призначається одному клієнту. Кількість використовуваних ядер машини - 8 ядер з налаштуванням кількох споживачів. Таким чином, кількість машинних ядер забезпечує ефективну обробку.

– Кілька вузлів: конфігурація від 1 до 2 вузлів споживачів означає, що пропускна здатність збільшується в 1,8 рази, але конфігурація від 2 до 4 вузлів споживачів покращується в 1,4 рази.

Вимоги Kafka щодо надійності залежать від різних випадків використання. У випадку потокової передачі відео дублікати повідомлень не повинні надходити до кінцевої точки. Для цього випадку використання Kafka надає три семантики доставки.

Перший підхід — це семантика at-most-once, коли виробник безперервно надсилає повідомлення, не чекаючи підтвердження від споживачів. Таким чином, у цьому випадку доставлене повідомлення може бути дублікатом або повідомлення може бути втрачено.

Друга семантика — принаймні один раз, коли виробник чекає підтвердження від серверів. Іноді такий підхід може призвести до дублювання повідомлень.

Третій підхід — семантика точного разу, яка гарантує, куди будуть доставлені всі повідомлення без дублювання. Для нашого підходу використано семантику точно один раз.

Spark SQL використовується для групування одного кадру ідентифікатора камери для аналізу. Алгоритм 1 (рисунок 3.3) показує етапи захоплення відео та перетворення кадрів у об’єкти JSON.

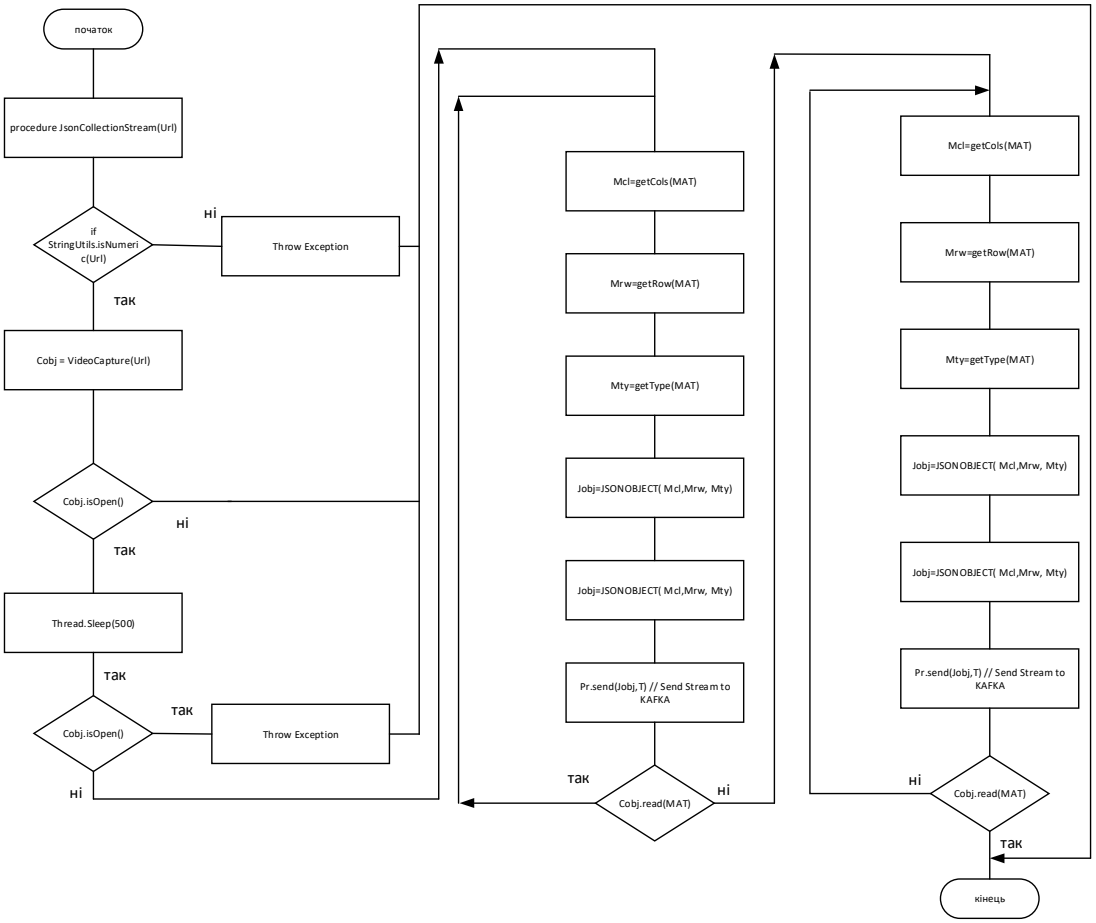


Рисунок 3.3 - Етапи захоплення відео та перетворення кадрів

В алгоритмі входом є URL-адреса камери, а виходом є об’єкт JSON. Необхідною умовою є наявність налаштованих параметрів для Kafka producer, і до проекту Java Spring необхідно додати зовнішню бібліотеку OpenCV. Якщо

джерелом відео є не камера пристрою, замість URL-адреси можна додати фактичний шлях до відео. За потреби можна змінювати параметри URL в алгоритмах. Ці алгоритми розроблені як для LVS, так і для VoDS.

3.4 Налаштування конфігурації Kafka об’єкта JSON для аналізу відео

Збереження порядку повідомлень в одному розділі є обов’язковим, коли аналіз відео виконується за допомогою об’єктів JSON. Kafka допомагає підтримувати порядок, додаючи значення ключа в об’єкт JSON під час створення даних. Щоб зберігати великі повідомлення, потрібно змінити деякі налаштування на стороні сервера. По-перше, потрібно наштувати файл властивостей message.max.bytes, а по-друге, replica.fetch.max.bytes на стороні сервера, а для споживача налаштувати max.partion.fetch.byte і max.poll.records. Усі ці конфігурації допомагають безперервно передавати об’єкти JSON у Spark для аналізу відео. Під час налаштування Kafka для обчислення великої кількості даних слід звернути увагу на деякі параметри. Необхідно встановити наступні параметри для конфігурації сторони споживача та сторони виробника.

1. Параметр конфігурації Kafka для збирача потоків

(a) kafka.ask = all

(b) kafka.retries = 1

(c) kafka.batch.size = 20971520

(d) kafka.linger.ms = 5

(e) kafka.compression.type = gzip

(f) kafka.max.request.size = 2097152

2. Параметр конфігурації Kafka для потокового процесора

(a) kafka.max.partition.fetch.bytes = 2097152

(b) kafka.max.poll.records = 500

На стороні збирача потоку kafka.ask = all означає, коли виробник отримує підтвердження від усіх синхронізованих реплік до лідера. kafka.retries

= 1 вказує кількість повторних спроб, якщо будь-який брокер не працює. kafka.batch.size означає загальну кількість байтів повідомлення, яке потрібно зібрати, перш ніж відправити його виробнику. kafka.linger.ms означає надання виробнику інформації про час очікування певної ms в очікуванні надходження додаткових записів.

GZIP — це тип стиснутого файлу, який стискає файл до меншого розміру та найкраще підходить для швидшої передачі по мережі. kafka.max.request.size позначає загальний розмір запису. Поточний процесор, kafka.max.partition.fetch.byte, вказує максимальний обсяг даних, отриманих з розділу. kafka.max.poll.records означає надання споживачеві інформації про максимальну кількість записів, що повертаються під час одного виклику споживача для функції poll().

На рисунку 3.4 описано, як об'єднати дані в потоки об'єктів JSON через Kafka та Spark із використанням OpenCV для аналізу.

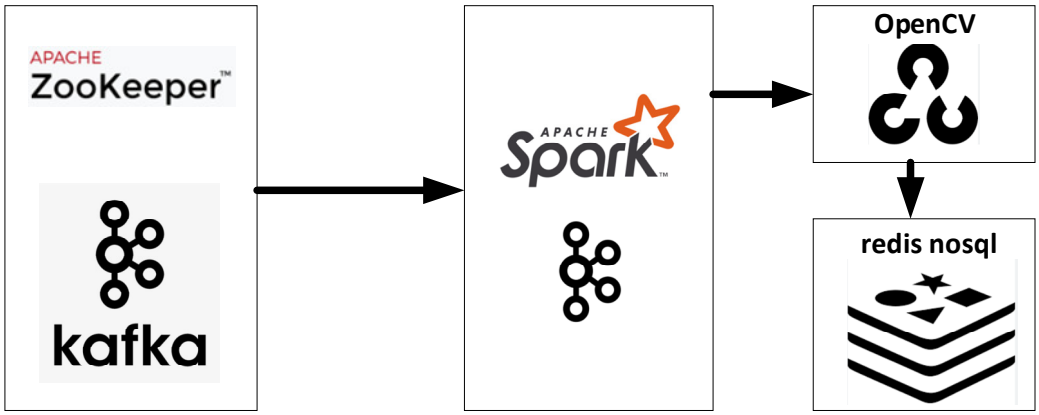


Рисунок 3.4 - Потік Kafka та ядро Spark

Деякі з джерел відео також можуть бути пристроями IoT. Проміжне програмне забезпечення системи підписки використовується для зберігання отриманих даних IoT у базі даних. Більшість пристроїв IoT потребують реалізації системи повідомлень із високим рівнем паралелізму та малою затримкою. Деколи для цього використовують структуру SSM. Spring — це фреймворк Java із відкритим вихідним кодом, який надає багато

функціональних можливостей для роботи з великомасштабними даними, наприклад, з'єднувач Spring-Kafka, Spring JPA, Spring Cloud і Spring CloudStream.

Для проміжного програмного забезпечення базі даних потрібен швидкий ключ-значення в пам'яті, тому база даних Redis використовується для зберігання отриманого набору даних із парами ключ-значення. Redis — це сховище структури даних із відкритим вихідним кодом, яке використовується як база даних, брокер повідомлень і кеш. Redis підтримує абстрактні структури даних, такі як растрові зображення, рядки, списки, набори, відсортовані набори, карти тощо.

Алгоритм 2 (рисунок 3.5) показує, як споживачі Kafka отримують дані та розміщують потоки в механізмах потоку Spark.

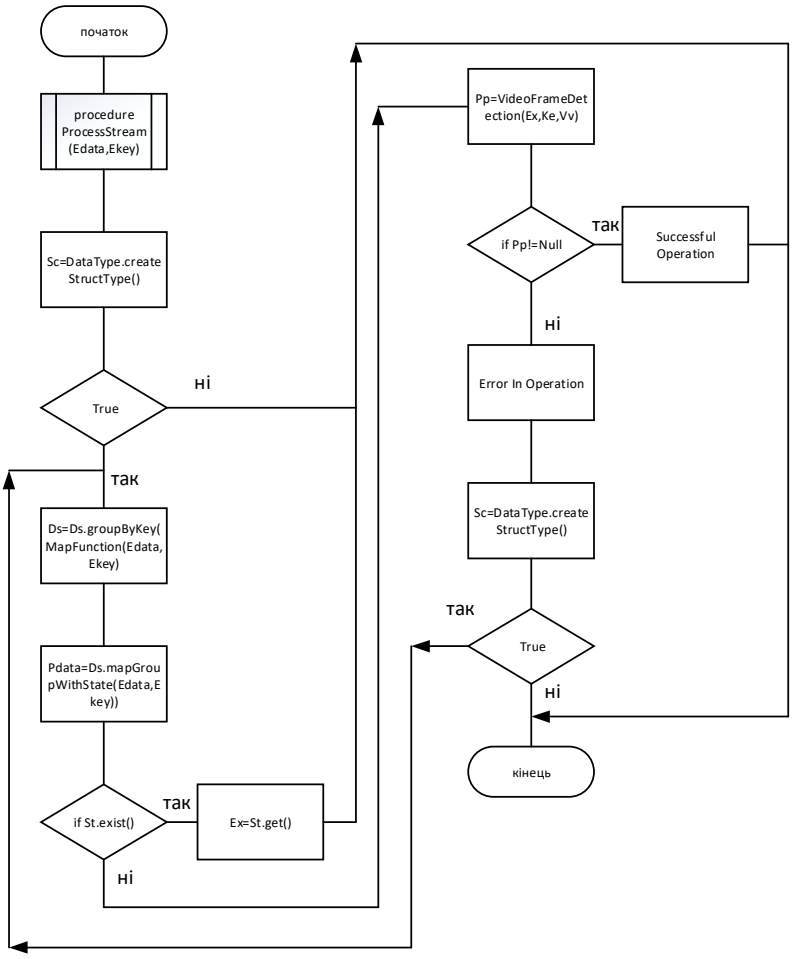


Рисунок 3.5 - Отримання даних та розміщення потоків в механізмах потоку

Spark використовує функцію MapFunction() для обробки даних кожного потоку окремо. Spark допомагає групувати потокові дані, використовуючи той самий ідентифікатор камери, створює стан із групи та передає дані групи станів у функцію події.

Функція Event() є VideoFrameDetection в алгоритмі замість VideoFrameDetection може використовувати функцію Event(), яка аналізує кожен кадр окремо та пише власний алгоритм для функції Event().

Функція groupByKey() створює групу за допомогою ідентифікатора камери з потоку JSON. Функція mapGroupWithState() створює стан для групи та обслуговує всі дані з кожної групи для функції Event().

3.5 Оцінювання результатів

3.5.1 Налаштування симуляції

Для симуляції було використано CloudSim [17], симулятор дискретних подій, щоб змоделювати систему та оцінити ефективність методів планування. Результати оцінок зображені на рисунках 3.6 і 3.7. Щоб вивчити систему за різних трафіків прямого ефіру.

Оцінка проводилась з різною кількістю запитів на прямий ефір протягом однієї одиниці часу. Використовувались порівняльні відео для імітації джерел потокового мовлення. Також було змодельовано продуктивність віртуальних машин на основі характеристик віртуальних машин. Задля точності кожен експеримент було повторено 10 разів, і вказано середнє значення та 95% довірчий інтервал результатів.

3.5.2 Вплив застосування планування з урахуванням QoS

На рисунку 3.6 показано, як середня затримка запуску потоків в реальному часі зменшується, коли застосовано запропонований метод планування з урахуванням QoS, у порівнянні з ситуацією, коли метод планування не підтримує QoS.

Використовуючи планування з урахуванням QoS, можна зберегти середню затримку запуску менше 1 секунди. Що ще важливіше, затримка запуску залишається майже постійною, оскільки кількість запитів на пряму трансляцію збільшується.

Рисунок 3.6 показує, що середня швидкість падіння майже завжди менше 7%. Суть цього експерименту полягає в тому, щоб продемонструвати, що можна транскодувати прямі відеопотоки під час їх трансляції та на основі вимоги.

Також на цьому рисунок показано витрати, понесені постачальником послуг із плануванням з урахуванням QoS і без нього. Вертикальна вісь показує понесені витрати на основі витрат на AWS. Однак, оскільки значення понесених витрат були невеликими для експериментальних контрольних показників, для кращого представлення значення на графіку було помножено на 1000. І як видно на рисунку, понесені витрати постачальника послуг майже однакові в обох випадках. Загальний час використання хмарних віртуальних машин також однаковий. Цей експеримент показує, що за допомогою планувальника з підтримкою QoS можна підвищити з QoS для користувачів без додаткових витрат для постачальника потоку.

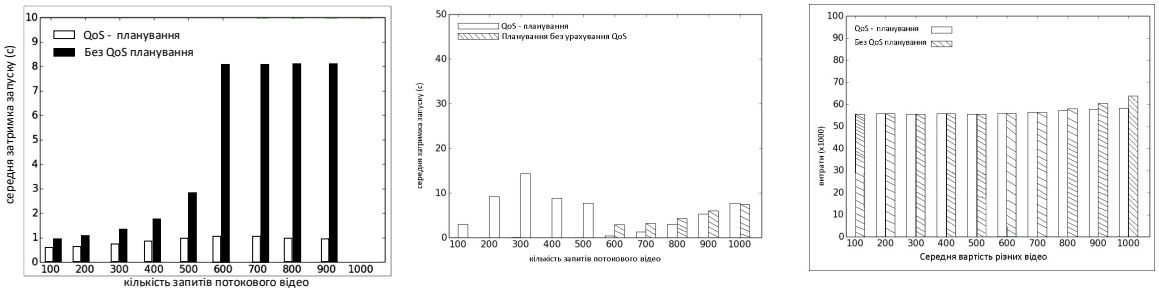
3.5.3 Вплив застосування різних політик черги

На рисунку 3.6 показано, як політика пакетної черги впливає на затримку запуску та швидкість падіння GOP, коли застосовано метод планування з урахуванням QoS. Щоб продемонструвати це, було оцінено три різні політики, а саме: «першим прийшов, перший обслужений» (FCFS), «найкоротша робота — спочатку» (SJF) і «найкоротший термін виконання» (SDF).

В результаті спостерігається, що зі збільшенням кількості запитів на пряму трансляцію SDF призводить до найнижчої затримки запуску та частоти падіння. Це пояснюється тим, що SDF спершу відображає найтерміновіші завдання GOP (тобто завдання, терміни виконання яких швидко

наближаються). Однак SJF надає пріоритет GOP з коротким часом перекодування, незалежно від терміновості їх виконання. Це призводить до великого відсотка падіння для завдань GOP із термінами, які часто зустрічаються в системах прямого ефіру. У політиці FCFS GOP у пакетній черзі мають чекати, доки всі GOP не надійдуть раніше, щоб бути перекодованими. Це призводить до найвищого відсотка падіння в порівнянні з іншими політиками черги.

На рисунку 3.7 показано, що всі три політики черги коштують майже однаково. Загальний час перекодування всіх відео є однаковим, і постачальник послуг потокового передавання несе майже однакову суму для будь-якої політики черги, коли виділено фіксовану кількість віртуальних машин.



a) b) c)

Рисунок 3.6 - Порівняння продуктивності за методом планування з урахуванням QoS і без нього,

На рисунку графік (а) показує середню затримку запуску, (б) ілюструє середню швидкість падіння, а (в) демонструє середні понесені витрати з різними числами запитів потокового відео в прямому ефірі.

Понесені витрати множаться на 1000 для кращого представлення.

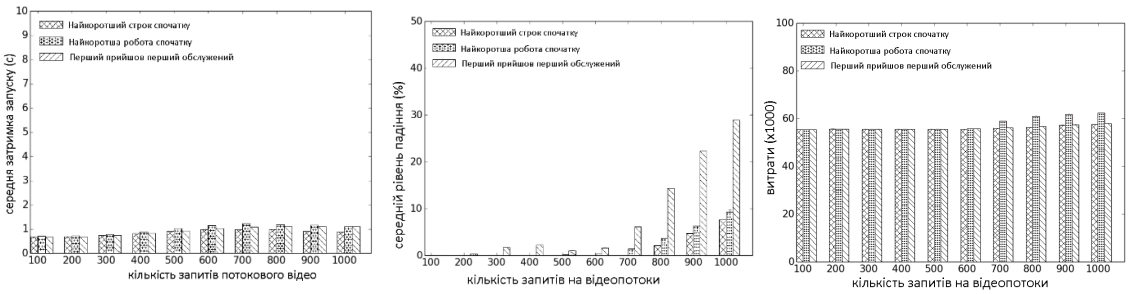


Рисунок 3.7 - Порівняння продуктивності з різними політиками постановки в черзі, застосованими до методу планування з урахуванням QoS

На графіку (a) показано середню затримку запуску, (b) ілюструє середню швидкість відключення, і (c) демонструє середні понесені витрати за різних черг політики. Понесені витрати множаться на 1000 для кращого представлення.

Висновки до розділу 3

1. Розроблена модель хмарної передачі відео в реальному часі демонструє ефективний підхід до трансляції мультимедійного контенту за умов мінімізації витрат та дотримання високих стандартів якості обслуговування (QoS). Завдяки інтеграції хмарних обчислень, розподілених систем та методів оптимізації, дана архітектура вирішує ряд важливих технічних викликів.

2. Було реалізовано комплексну архітектуру, яка включає компоненти для розподілу відео, оцінки часу виконання завдань, планування перекодування, віртуальні машини для обробки, менеджери віртуальних кластерів та злиття відеопотоків. Це дозволяє забезпечити обробку та трансляцію відео з мінімальними затримками та високою стійкістю до навантажень.

3. Застосування планувальника задач, орієнтованого на QoS, суттєво зменшує середню затримку запуску потоків (менше однієї секунди) та зберігає високу якість трансляції навіть при зростанні кількості запитів. Методика

також дозволяє уникати перевитрат ресурсів, що підтверджується стабільними показниками витрат на хмарні сервіси.

4. Використання різних політик черги (FCFS, SJF, SDF) продемонструвало, що найефективнішим підходом є "найкоротший термін виконання" (SDF), який мінімізує кількість втрачених завдань і затримок. Це важливо для реальних застосувань, де часові рамки мають критичне значення.

5. Інтеграція Kafka та Spark для аналізу потоків дозволяє виконувати масштабну обробку даних у реальному часі, що відкриває перспективи для впровадження аналітичних інструментів у системи потокового відео. Використання мікросервісів, таких як Spring і Redis, покращує ефективність роботи системи та забезпечує її масштабованість.

6. Запропонована модель була протестована за допомогою симуляції, яка підтвердила її продуктивність та економічну доцільність. Результати експериментів демонструють, що запропонований підхід може успішно застосовуватися для трансляції відео в реальному часі в різних масштабах.

ВИСНОВКИ

1. Розроблено та протестовано модель хмарної передачі відео в реальному часі, яка відповідає сучасним потребам мультимедійних технологій. Дана модель поєднує динамічне пакування, хмарне перекодування та адаптивні методи для забезпечення високої якості обслуговування (QoS).

2. Для системи використано безсерверну архітектуру з інтеграцією Amazon S3, що забезпечує масштабованість, безпеку та адаптивність відеопотоків.

3. Реалізація паралельного програмування та моделі «виробник-споживач» дало можливість оптимізації обробки і передачі даних, знижуючи затримки у складних умовах мережі. Крім того застосування контейнерних технологій, таких як Docker, покращило масштабованість сервісів.

4. Розроблено планувальник задач, орієнтований на QoS, який мінімізує затримки запуску потоків та забезпечує баланс між якістю обслуговування користувачів та економічними витратами постачальників послуг.

5. Результати симуляцій підтверджують ефективність підходу, зокрема підтримку стабільної затримки навіть за умов збільшення кількості запитів.

6. Забезпечено швидкий доступ до інформації та підтримку складних аналітичних запитів.

7. Використання Redis як швидкого сховища даних та мікросервісів Spring підвищило продуктивність і гнучкість системи.

Розроблена модель є ефективним рішенням для трансляції відео в реальному часі, що підтверджено результатами тестування, а з апропонований підхід дає можливості для розвитку мультимедійних сервісів, забезпечуючи високу якість обслуговування та економічну доцільність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global Internet Phenomena Report, 2023. https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2023/reports/Sandvine%20GIPR%202023.pdf

2. Netflix Is Now Bigger Than Cable TV, 2017. https://www.forbes.com/sites/ianmorris/2017/06/13/netflix-is-now-bigger-than-cable-tv/

3. Ali, Ashraf, and Abdullah Alourani. "An investigation of cloud computing and E-learning for educational advancement." International Journal of Computer Science & Network Security 21.11 (2021): 216-222.

4. Jin, Yili, et al. "3D Video Conferencing via On-hand Devices." IEEE Transactions on Circuits and Systems for Video Technology (2024).

5. Elharrouss, Omar, Noor Almaadeed, and Somaya Al-Maadeed. "A review of video surveillance systems." Journal of Visual Communication and Image Representation 77 (2021): 103116.

6. Mareen, Hannes, et al. "A study on keyframe injection in three generations of video coding standards for fast channel switching and packet-loss repair." Multimedia Tools and Applications 83.15 (2024): 44485-44501.

7. X. Li, M.A. Salehi, M. Bayoumi, N.-F. Tzeng, R. Buyya, Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services, IEEE Trans. Parallel Distrib. Syst. (TPDS) 29 (3) (2018) 556–571.

8. Shi, Wanxin, et al. "A Survey on Intelligent Solutions for Increased Video Delivery Quality in Cloud-Edge-End Networks." IEEE Communications Surveys & Tutorials (2024).

9. Moghaddam, Sara Kardani, Rajkumar Buyya, and Kotagiri Ramamohanarao. "Performance-aware management of cloud resources: A taxonomy and future directions." ACM Computing Surveys (CSUR) 52.4 (2019): 1-37.

10. X. Li, M.A. Salehi, M. Bayoumi, VLSC:video live streaming using cloud services, in: Proceedings of the 6th IEEE International Conference on Big Data and Cloud Computing Conference, October, 2016.

11. Netflix and AWS, 2021. https://aws.amazon.com/solutions/case-studies/netflix/.

12. Abid, Adnan, et al. "Challenges and Issues of Resource Allocation Techniques in Cloud Computing." KSII Transactions on Internet & Information Systems 14.7 (2020).

13. Holzmüller, David, et al. "A framework and benchmark for deep batch active learning for regression." Journal of Machine Learning Research 24.164 (2023): 1-81.

14. Reddy, Venkateswara, Khader Basha Sk, and D. Roja. "Qos-Aware Video Streaming Based Admission Control And Scheduling For Video Transcoding In Cloud Computing." 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS). IEEE, 2022.

15. Moina-Rivera, Wilmer, et al. "Cloud media video encoding: review and challenges." Multimedia Tools and Applications (2024): 1-48.

16. Wang, Desheng, et al. "Adaptive wireless video streaming based on edge computing: Opportunities and approaches." IEEE Transactions on services Computing 12.5 (2018): 685-697.

17. Izima, Obinna, Ruairí de Fréin, and Ali Malik. "A survey of machine learning techniques for video quality prediction from quality of delivery metrics." Electronics 10.22 (2021): 2851.

18. Yang, Wanting, et al. "Semantic communications for future internet: Fundamentals, applications, and challenges." IEEE Communications Surveys & Tutorials 25.1 (2022): 213-250.

19. Chen, Yue, et al. "An overview of core coding tools in the AV1 video codec." 2018 picture coding symposium (PCS). IEEE, 2018.

20. Graziosi, Danillo, et al. "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based

(G-PCC)." APSIPA Transactions on Signal and Information Processing 9 (2020): e13.

21. Bentaleb, Abdelhak, et al. "A survey on bitrate adaptation schemes for streaming media over HTTP." IEEE Communications Surveys & Tutorials 21.1 (2018): 562-585.

22. MPEG-2 Transport of Compressed Motion Imagery and Metadata, 2014. http:// www.gwg.nga.mil/misb/docs/standards/ST1402.pdf.

23. Walker, Gordon Kent, and Michael G. Luby. "Network streaming of media data." U.S. Patent No. 9,843,844. 12 Dec. 2017.

24. Bogdanov, Lubomir, and Hristo Mitrev. "Flash programming microcontrollers over the gsm network." 2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE, 2021.

25. Alvestrand, Harald. "RFC 8825: Overview: Real-Time Protocols for Browser-Based Applications." (2021).

26. Evans, Rhys Christopher. "Systems and methods for providing managed services." U.S. Patent No. 11,388,037. 12 Jul. 2022.

27. R. Pantos, W. May, HTTP Live Streaming, 2017

28. Luong, Nguyen Cong, et al. "Applications of deep reinforcement learning in communications and networking: A survey." IEEE communications surveys & tutorials 21.4 (2019): 3133-3174.

29. Mao, Hongzi, Ravi Netravali, and Mohammad Alizadeh. "Neural adaptive video streaming with pensieve." Proceedings of the conference of the ACM special interest group on data communication. 2017.

30. Zolfaghari, Behrouz, et al. "Content delivery networks: State of the art, trends, and future roadmap." ACM Computing Surveys (CSUR) 53.2 (2020): 1-34.

31. Izima, Obinna, Ruairí de Fréin, and Ali Malik. "A survey of machine learning techniques for video quality prediction from quality of delivery metrics." Electronics 10.22 (2021): 2851.

32. Chowdhury, Debanjan Roy, Sukumar Nandi, and Diganta Goswami. "Cost-effective live video streaming for internet of connected vehicles using heterogeneous networks." Ad Hoc Networks 153 (2024): 103334.

33. Zolfaghari, Behrouz, et al. "Content delivery networks: State of the art, trends, and future roadmap." ACM Computing Surveys (CSUR) 53.2 (2020): 1-34.

34. Yu, Jun, et al. "Leveraging content sensitiveness and user trustworthiness to recommend fine-grained privacy settings for social image sharing." IEEE transactions on information forensics and security 13.5 (2018): 1317-1332.

35. Rippel, Oren, et al. "Learned video compression." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.

36. Li, Xiangbo, et al. "Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services." IEEE Transactions on Parallel and Distributed Systems 29.3 (2017): 556-571.

37. X. Li, M. Amini Salehi, Y. Joshi, M.K. Darwich, B. Landreneau, M. Bayoumi, Performance analysis and modeling of video transcoding using heterogeneous cloud services, IEEE Trans. Parallel Distrib. Syst. 30 (4) (2018) 910–922.

38. Shrestha, Rakesh, Rojeena Bajracharya, and Seung Yeob Nam. "Challenges of future VANET and cloud-based approaches." Wireless Communications and Mobile Computing 2018.1 (2018): 5603518.

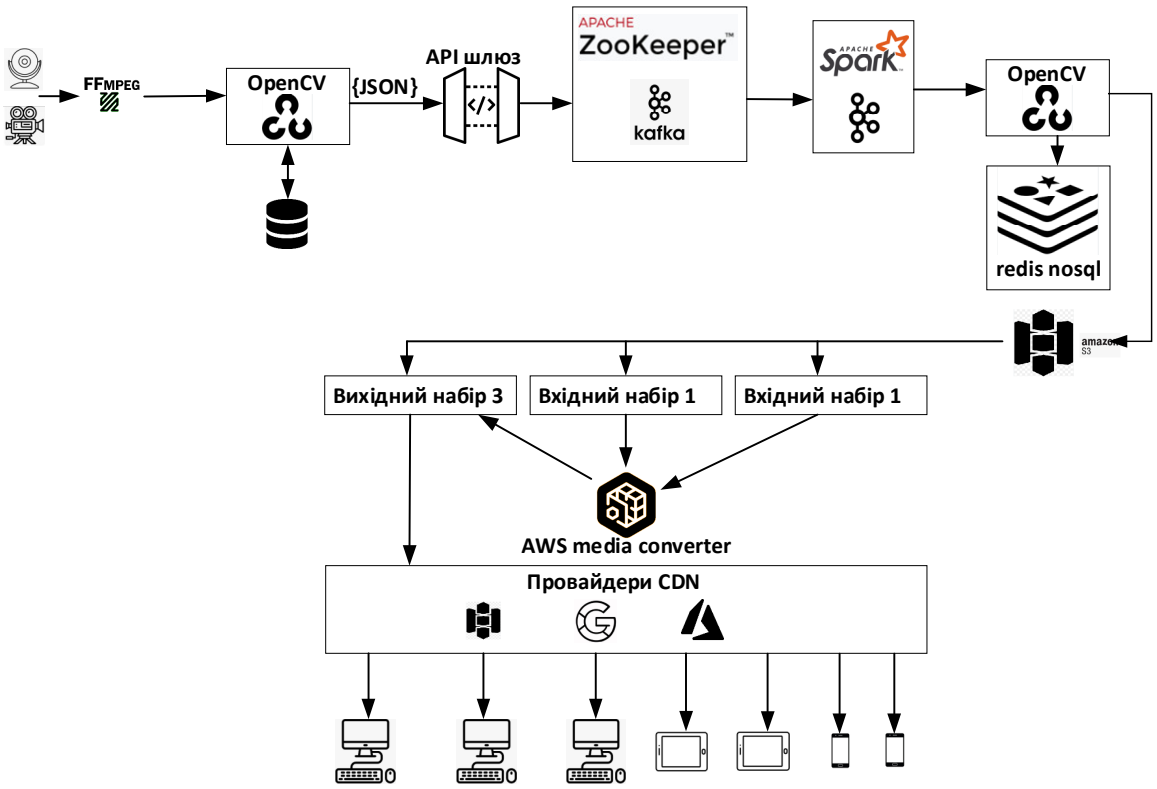
39. Грицай Ростислав Ігорович, “Модель Хмарної Передачі Відео“, матеріалами IV Міжнародної наукової конференції «Період трансформаційних процесів в світовій науці: задачі та виклики» (м. Рівне, Україна).

40. Грицай Ростислав Ігорович, «Архітектура системи трансляції відео в реальному часі за допомогою хмарних сервісів», матеріали VIII Міжнародної студентської наукової конференції «Актуальні питання та перспективи проведення наукових досліджень» (м. Дніпро, Україна).

41. 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

Загальна архітектура системи



Додаток Б

Код захоплення відео та перетворення кадрів

```
Алгоритм 1
import cv2
import json
import time
from kafka import KafkaProducer

def json_collection_stream(url, kafka_topic, kafka_server):
    producer = KafkaProducer(bootstrap_servers=kafka_server)

    while True:  # Simulating Thread.Run()
        if url.isdigit():  # Check if URL is numeric
            cobj = cv2.VideoCapture(int(url))
        else:
            raise Exception("Invalid URL: URL must be numeric")

        if cobj.isOpened():
            time.sleep(0.5)
            if cobj.isOpened():
                raise Exception("Camera is already in use or cannot be accessed")
            else:
                while cobj.isOpened():
                    ret, mat = cobj.read()
                    if not ret:
                        break
                    mcl = mat.shape[1]  # Number of columns
                    mrw = mat.shape[0]  # Number of rows
                    mty = str(mat.dtype)  # Type of the matrix
                    jobj = json.dumps({"cols": mcl, "rows": mrw, "type": mty})
                    producer.send(kafka_topic, value=bytes(jobj, 'utf-8'))  # Send JSON object to Kafka
        else:
            while cobj.isOpened():
                ret, mat = cobj.read()
                if not ret:
                    break
                mcl = mat.shape[1]  # Number of columns
                mrw = mat.shape[0]  # Number of rows
                mty = str(mat.dtype)  # Type of the matrix
                jobj = json.dumps({"cols": mcl, "rows": mrw, "type": mty})
                producer.send(kafka_topic, value=bytes(jobj, 'utf-8'))  # Send JSON object to Kafka
```

Додаток В

Код алгоритму отримання даних та розміщення потоків в механізмах потоку

```
Алгоритм 2
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType
from pyspark.sql.functions import col
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

def process_stream(edata, ekey, kafka_topic, kafka_server):
    # Ініціалізація SparkSession
    spark = SparkSession.builder.appName("Video Analysis Operation").getOrCreate()

    # Ініціалізація StreamingContext
    ssc = StreamingContext(spark.sparkContext, batchDuration=5)  # Batch size 5 seconds

    # Створення схеми для даних
    schema = StructType()

    # Підключення до Kafka
    kafka_stream = KafkaUtils.createDirectStream(
        ssc,
        topics=[kafka_topic],
        kafkaParams={"metadata.broker.list": kafka_server}
    )

    # Обробка потокових даних
    def map_group_with_state(record):
        edata = record[1]  # Отримати дані
        ekey = record[0]  # Отримати ключ
        if state.exists():
            ex = state.get()
        else:
            ex = None

        pp = video_frame_detection(edata, ekey, ex)
        if pp is not None:
            print("Successful Operation")
        else:
            print("Error In Operation")

    # Прив'язка функції до обробки даних
    kafka_stream.foreachRDD(lambda rdd: rdd.foreach(map_group_with_state))

    # Запуск потоку
    ssc.start()
    ssc.awaitTermination()

def video_frame_detection(ex, key, vv):
    # Реалізація аналізу відеокадрів
    try:
        # Приклад обробки відеокадрів
        return True  # Якщо обробка успішна
    except Exception as e:
        print(f"Error in VideoFrameDetection: {e}")
        return None
```

Додаток Г

Апробація отриманих результатів

СЕКЦІЯ XVIII.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

МОДЕЛЬ ХМАРНОЇ ПЕРЕДАЧІ ВІДЕО

Грицай Ростислав Ігорович
Магістр спеціальності 122 «Комп'ютерні науки»
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович
ORCID ID: 0000-0002-0136-395X
канд. техн. наук, доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

Вступ

Існує зростаючий попит на потокове відео в прямому ефірі, і поточні ринки пропонують високу вартість надання цієї послуги. Компанії повинні підтримувати сервери та сховище, у той час як відеоконтент зростає, як у якості, так і в кількості. Завдяки використанню хмарних служб зберігання, як Amazon Simple Storage Solutions (S3), потокове відео може запропонувати конкурентоспроможну продуктивність і вищу економію в порівнянні з доступними послугами. Хмарне потокове відео в реальному часі використовує Amazon S3, щоб усунути потребу в оренді сервера. Ця система добре підходить для поточного попиту на технологію потокового передавання.

Велика кількість компаній почали застосовувати прямі трансляції. Згідно з роботою [1], 44% компаній провели одну або кілька подій, що транслюються в прямому ефірі, 35% планують тестувати або проводити трансляцію, а 20% будуть тестувати одну або більше подій у прямому ефірі. Велику участь бізнес-спільноти можна пояснити зростаючою доступністю розумних пристроїв разом із інтеграцією широко використовуваних платформ соціальних мереж.

Окрім того індустрія потокового відео стрімко зростає, і відео в прямому ефірі впроваджується найбільшими компаніями у світі. Більшість компаній, які пропонують послуги потокового відео в прямому ефірі, стягують плату за використану пропускну здатність, доступ клієнтів, вартість зберігання, ліцензійні угоди та години перегляду [2] [3] [4]. Вартість використання використовує модель, яка полегшує ліцензійні угоди. У багатьох випадках ціна на прямі трансляції відео за допомогою хмарних служб набагато нижча, ніж на використання

сучасних служб потокового відео в прямому ефірі, але забезпечує порівнянну продуктивність.

Пряма трансляція відео за допомогою хмарних служб використовує деякі основні методи потокового відео, але має деякі ключові відмінності. Основною є безсерверна архітектура. Замість використання проміжних серверів вона використовує Amazon S3 як спільний простір для передачі даних. Щоб отримати доступ до потоку, глядач використовує відповідну назву сегмента та ключ, а також дійсні облікові дані, після чого відображається останній сегмент відео. Ця техніка має багато переваг, включаючи масштабованість, вартість, доступ до додаткових хмарних ресурсів, таких як мережі доставки контенту (CDN), і безпеку. Такий дизайн спрощує реалізацію потокового відео, усуває потребу у виділених серверах і є альтернативою потоковому відео в реальному часі.

Пропонований підхід для хмарної трансляції відео в реальному часі

Джерело відео записує подію, кожен кадр стискається, фрагмент стиснутих кадрів надсилається в хмарне сховище, клієнт отримує кожен фрагмент із хмарного сховища та переглядає відео. Рисунок 1 ілюструє подібність між моделлю і сучасним потоковим відео, тоді як рисунок 2показує ключові операції в хмарній трансляції відео в реальному часі.

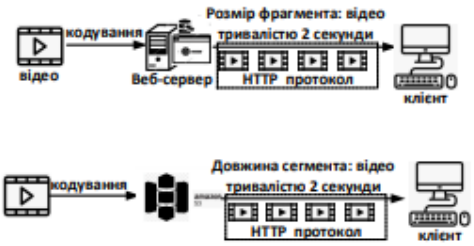


Рис. 1. **Основи прямої трансляції відео за допомогою хмарних служб**

Це рішення також відрізняється від деяких звичайних застосувань, тим що використовує ефективний підхід із застосуванням методів паралельного програмування, попередньої вибірки та споживчої моделі виробника.

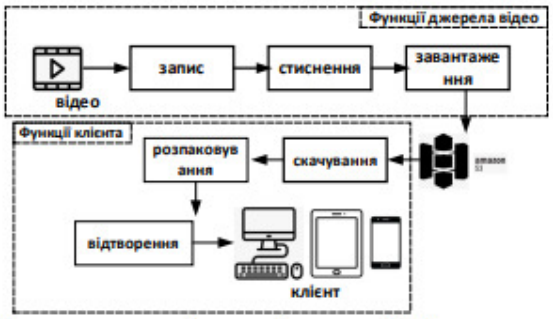


Рис. 2. **Загальна концепція прямої трансляції відео за допомогою хмарних служб**

Висновки. Запропонована модель хмарної передачі відео в реальному часі, демонструє значне покращення трансляції мультимедійного контенту. Основна ідея використання безсерверної архітектури дозволяє ефективно усунути потребу у виділених проміжних серверах, спрощуючи процес і знижуючи витрати. Використання технології Amazon S3, як центрального елемента зберігання та передачі даних підвищує масштабованість і безпеку системи, а також дозволяє адаптувати потоки до потреб користувачів у реальному часі.

Реалізація паралельного програмування, попереднього завантаження та моделі «виробник-споживач», значно оптимізує процеси обробки та доставки відео. Використання ефективного стиснення, сегментації та зберігання сегментів у хмарі створює умови для швидкої та надійної передачі контенту з мінімальними затримками.

Таким чином, модель пропонує підхід до організації потокового відео, який є економічно вигідним, технологічно простим і є високо адаптивним. Така модель відповідає сучасним тенденціям мультимедійних технологій та забезпечує перспективи для подальшого розвитку потокового відео, інтегрованого з хмарними сервісами, задовольняючи запити як користувачів, так і постачальників контенту.

Список використаних джерел:

1. Clímaco, Isabel N., and Manuela Larguinho. "Streaming Consumers: Series Versus Videos, What Distinguishes Them?." Media Watch 15.1 (2024): 53-72.

2. Colbjørnsen, Terje, Alan Hui, and Benedikte Solstad. "What do you pay for all you can eat? Pricing practices and strategies in streaming media services." Journal of Media Business Studies 19.3 (2022): 147-167.
3. DaCast, "Dacast homepage," 2024. [Electronic resource], Mode of access: http://www.dacast.com
4. Livestream, "Livestream platform plans," 2024. [Electronic resource], Mode of access: https://livestream.com/platform/pricing


МАТЕРІАЛИ VIII МІЖНАРОДНОЇ
СТУДЕНТСЬКОЇ НАУКОВОЇ
КОНФЕРЕНЦІЇ
АКТУАЛЬНІ ПИТАННЯ ТА
ПЕРСПЕКТИВИ ПРОВЕДЕННЯ
НАУКОВИХ ДОСЛІДЖЕНЬ
ISO
М. ДНІПРО, УКРАЇНА
13 ГРУДНЯ
2024 РІК

СЕКЦІЯ 16.
СИСТЕМНИЙ АНАЛІЗ, МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЯ

СЕКЦІЯ 17.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

Грицай Ростислав Ігорович, магістр спеціальності 122 "Комп'ютерні науки"
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович, канд.техн.наук, доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

АРХІТЕКТУРА СИСТЕМИ ТРАНСЛЯЦІЇ ВІДЕО В РЕАЛЬНОМУ ЧАСІ ЗА ДОПОМОГОЮ ХМАРНИХ СЕРВІСІВ

Вступ

Згідно зі звітом Global Internet Phenomena Report [1], потокове відео становить приблизно 64% усього інтернет-трафіку. Було передбачено, що трафік потокового відео зросте до 80% інтернет-трафіку до 2020 року [2].

Одним із все більш популярних типів потокового відео є служби потокового передавання в прямому ефірі, які дозволяють клієнтам (тобто видавцям відео) використовувати камеру та транслювати відео через Інтернет.

Щоб забезпечити високоякісну послугу потокового відео в прямому ефірі на різних пристроях глядачів, відеоконтент потрібно перекодувати (перетворити) на основі характеристик пристроїв глядачів, наприклад, роздільна здатність, пропускна здатність мережі та підтримуваний кодек.

Наразі, щоб підтримувати високоякісну пряму трансляцію на різних пристроях відображення, видавцям відео доводиться генерувати кілька версій (форматів) того самого відео, наприклад, кілька кодувань відео самостійно. Крім того, такий підхід не враховує запити глядачів. Тобто він потенційно може генерувати версії, які не запитуються глядачами. Таким чином, цей підхід залишається занадто дорогим і неефективним. Іншим підходом є перекодування потокового відео в прямому ефірі за запитом. У такому підході видавець створює лише одну версію відео. Відео в прямому ефірі можна транслювати глядачам, якщо їхні пристрої відображення сумісні з форматом потокової передачі [3].

Архітектура системи хмарного транскодування потокової трансляції

Пропонується архітектура для прямої трансляції з використанням хмарних сервісів. Загальний огляд архітектури представлено на рисунку 1.

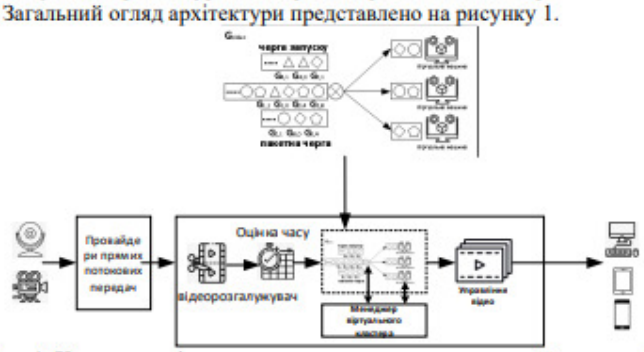


Рис. 1. **Хмарна архітектура транскодування потокової трансляції**

Дана архітектура показує послідовність дій, які виконуються, коли глядачі запитують відео від постачальника послуг потокового передавання. Архітектура включає шість основних компонентів, а саме відеорозподільник, оцінювач часу, планувальник завдань (групи зображень), віртуальні машини транскодування (VM), диспетчер віртуальних кластерів (VCM) і злиття відео. Комбінація цих компонентів веде до економічно ефективного перекодування потоків в реальному часі у хмарі з урахуванням якості обслуговування.

У компоненті розділювача відео відеопотік в реальному часі після надходження розбивається на кілька груп зображень, які потім можна перекодувати незалежно.

Планувальник завдань перекодування відповідає за відображення завдань групи зображень на серверах перекодування.

Віртуальні машини виділяються від хмарного постачальника для обробки завдань групи зображень.

Менеджер віртуального кластера відстежує роботу транскодування віртуальних машин в архітектурі потокової трансляції і змінює розмір кластера віртуальної машини, щоб відповідати вимогам клієнта щодо QoS і мінімізувати понесені витрати постачальника послуг потокового передавання.

Метою об'єднання відео є розміщення всіх перекодованих груп зображень у правильному порядку та створення прямого потоку.

Висновки. Розроблена архітектура демонструє ефективний підхід до трансляції мультимедійного контенту за умов мінімізації витрат та дотримання високих стандартів якості обслуговування (QoS). Завдяки інтеграції хмарних обчислень, розподілених систем та методів оптимізації, дана архітектура вирішує ряд важливих технічних завдань. Основною перевагою є реалізація комплексної архітектури, яка включає компоненти для розподілу відео, оцінки часу виконання завдань, планування перекодування, віртуальні машини для обробки, менеджери віртуальних кластерів та злиття відеопотоків. Це дозволяє забезпечити обробку та трансляцію відео з мінімальними затримками та високою стійкістю до навантажень.

Список використаних джерел:

1. 2024 Global Internet Phenomena Report [Electronic resource], Mode of access: https://www.sandvine.com/phenomena
2. Darwich, Mahmoud, et al. "Cost-efficient storage for on-demand video streaming on cloud." 2020 IEEE 6th World Forum on Internet of Things (WF-IoT). IEEE, 2020.
3. Reddy, Venkateswara, Khader Basha Sk, and D. Roja. "Qos-Aware Video Streaming Based Admission Control And Scheduling For Video Transcoding In Cloud Computing." 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS). IEEE, 2022.