МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____
(рівень вищої освіти)

Дослідження методів оптимізації динамічної обробки зображень з використанням рішень на AWS

Виконала:
студентка __2__ курсу групи ____ІПЗм-20-1__
_____Бабуріна Д. С._____
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного
_____забезпечення_____
Тип програми _____Освітньо-наукова_____
Керівник _____доц. Ревенчук І. А._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ З.В.Дудар

2022 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _____121 – Інженерія програмного забезпечення_____
(код і повна назва спеціальності)

Тип програми _____освітньо-наукова програма _____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Інженерія програмного забезпечення _____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)

«_____» _____20 ____ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента _____*Бабуріної Діани Сергіївни*_____
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів оптимізації динамічної обробки зображень з використанням рішень на AWS» _____

затверджена наказом університету від «24» березня 2022 р. № 412 Ст

2. Термін подання роботи до екзаменаційної комісії «__» _____ 2022 р.

3. Вихідні дані до роботи *методи динамічної обробки зображень, критерії їх оцінки, Node.js, AWS сервіси - EC2, Lambda, S3 Object Lambda, S3, CloudFront, Edge Locations, API Gateway.* _____

4. Перелік питань, що потрібно опрацювати в роботі *вступ, аналіз предметної області, аналіз методів динамічної обробки зображень, критерії їх оцінки, постановка задачі, проведення експериментів та аналіз їх результатів, формування подальших рекомендації.* _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	25.02.2022	виконано
2	Постановка задачі	10.03.2022	виконано
3	Проведення дослідження	01.04.2022	виконано
4	Підготовка пояснювальної записки	03.05.2022	виконано
5	Підготовка презентації та доповіді	06.05.2022	виконано
6	Попередній захист	10.05.2022	виконано
7	Перевірка на академічний плагіат	10.05.2022	виконано
8	Нормоконтроль	12.05.2022	виконано
9	Рецензування	13.05.2022	виконано
10	Знесення диплома в електронний архів	15.05.2022	виконано
11	Допуск до захисту у зав. кафедри	15.05.2022	виконано

Дата видачі завдання 17.01.2022 р.

Студент _____
(підпис)

Керівник роботи_____ доцент Ревенчук І. А.
(підпис)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 71 стор., 26 рис., 15 табл., 10 джерел, 6 додатків.

AWS, ЗОБРАЖЕННЯ, РЕСАЙЗИНГ, КОМПРЕСІЯ, КЕШУВАННЯ, ДИНАМІЧНА ОБРОБКА, NODE.JS.

Об’єктом дослідження є методи оптимізації динамічної обробки зображень на платформі AWS.

Метою дослідження є аналіз різних архітектурних підходів, що можуть бути використані для оптимізації динамічної обробки зображень на платформі AWS, порівняння інструментів для обробки зображень (Sharp, Imagemagick, JPEGmini) та огляд підходів до оптимізації сховищ даних.

У результаті роботи було проведено аналіз існуючих архітектурних рішень та підходів до обробки зображень, експериментально виявлено доцільність кожного з них та запропоновано метод, який є найбільш ефективним з точки зору продуктивності та ціни використання на платформі AWS.

AWS, IMAGE, RESIZING, COMPRESSION, CACHING, DYNAMIC IMAGE PROCESSING, NODE.JS.

The object of research is the methods of dynamic image processing optimization on the AWS platform.

The purpose of the research is to analyze different architectural approaches that can be used to optimize dynamic image processing on the AWS platform, compare image processing tools (Sharp, Imagemagick, JPEGmini) and review approaches to optimizing data warehouses.

As a result of research practice, the analysis of existing architectural solutions and approaches to image processing was carried out, the feasibility of each of them was experimentally identified and the method that is most effective in terms of performance and cost of use on the AWS platform was proposed.

Я, Бабуріна Діана Сергіївна, студентка групи ІПЗм-20-1, здобувач вищої освіти на другому (магістерському) рівні, кафедра Програмної інженерії, заявляю: моя кваліфікаційна робота на тему « Дослідження методів оптимізації динамічної обробки зображень з використанням рішень на AWS», що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	Application Programming Interface, прикладний програмний інтерфейс
AWS	Amazon Web Services, провайдер інфраструктури
CDN	Content Delivery Network, мережа доставки контенту
EC2	Elastic Compute Cloud, сервіс AWS, який надає можливість розгортати повноцінні сервери
HTTP	HyperText Transfer Protocol, протокол передачі даних
JPEG	Joint Photographic Experts Group, растровий графічний формат
JPEG-XR	JPEG extended range, розроблений Microsoft
PNG	Portable network graphics, растровий графічний формат який використовує стиснення без втрат
S3	Simple Storage Service, сховище даних на AWS
WebP	Формат зображень, розроблений Google

ВСТУП

Інтернет продовжує змінювати наше повсякденне життя: спілкування, обмін інформацією, роботу, розваги, тощо. З кожним днем він займає все більшу частину життя як і окремих людей, так і великих корпорацій. Відповідно до цього приріст користувачів на веб-сайтах збільшується. Разом із цим збільшуються і їх вимоги. Так наприклад якщо раніше користувачі не особливо звертали увагу на дизайн, швидкість роботи, зручність і легкість використання систем, то зараз ці критерії стають вирішальними при виборі. Одна справа, коли система несе більш інформаційний характер і не має на меті відображати велику кількість контенту. Проте більшість із них є платформами, на яких клієнти наприклад продають свої товари. Кількісті проданих товарів та їх прибуток залежить напряму від якості їх веб-сайтів.

За даними HTTP Archive [1], станом на листопад 2021 року зображення становлять в середньому 30% від загальної ваги веб-сторінки. Великі за розміром зображення уповільнюють роботу веб-сторінок та мобільних додатків. У результаті цього погіршується користувацький досвід. Тому оптимізація генерації та обробки зображень є досить актуальною проблемою, оскільки безпосередньо впливає на кожного з нас як кінцевого користувача щодня. По-перше, за допомогою оптимізації можна покращити швидкість завантаження сторінки. Типовою ситуацією є коли користувач просто втомлюється чекати, коли сторінка повністю завантажиться та переходить до іншого ресурсу. По-друге, зростає пропускна здатність: чим менше файл, ти менше ресурсів мережі необхідно для його завантаження. Це особливо критично для користувачів з повільним підключенням до Інтернету. Крім того, оптимізація зображень покращує SEO. Наприклад веб-сайт матиме вищу позицію в результатах пошуку, а відповідно і більшу кількість нових користувачів.

Для вирішення цієї проблеми існує ряд типових методів і підходів, наприклад компресія зображень, що дозволяє зменшити його розмір або поступове завантаження зображення, коли в залежності від швидкості мережі та пристрою

обирається відповідний розмір та якість зображення для завантаження. Ще одним із методів є використання JPEG формату, а не PNG, оскільки останній вважається форматом більш високої якості, тому витрачає більше ресурсів. Варто зазначити, що наведені приклади є скоріше точковими рішеннями, які не мають на мети розв'язати проблему на глобальному рівні. Звісно, вони мають ряд переваг і більшість систем їх використовують. Тому основним фокусом роботи є саме глобальний рівень вирішення проблеми.

Метою кваліфікаційної роботи є дослідження методів оптимізації динамічної обробки зображень. У ході роботи будуть розглянуті різні архітектурні підходи, які можуть бути використані, проведено їх порівняльний аналіз, а також сформовано рекомендації щодо їх подальшого використання. Ці методи будуть включати як оптимізації на глобальному, тобто архітектурному рівні, так і точково покращувати певний аспект рішення. Одними із головних задач є продемонструвати зазначену вище різницю рівнів вирішення та їх вплив на фінальний результат, а також знайти баланс між доцільністю використання певних оптимізацій та ціною підтримки такого рішення.

У якості платформи, на основі якої будуть розглянуті методи є AWS. За даними Statista [2] AWS займає майже 50 відсотків ринку постачальників інфраструктури на 2022 рік. Це майже в 5 разів більше, ніж Google Cloud Platform та приблизно у 1.5 рази більше, ніж Azure. Переважна більшість клієнтів обирає AWS за зручність використання, спектр сервісів, що надаються, та їх можливості, а особливо за приємну цінову політику. Серед головних сервісів, що будуть використовуватися у рамках роботи є EC2, Lambda, S3 Object Lambda, API Gateway, CloudFront та S3. Перші три із них є основними архітектурними підходами за допомогою яких може бути вирішена задача. EC2 є повноцінним сервером, Lambda є його легковісною версією, яка дещо обмежена за характеристиками, але не менш потужна по можливостях. S3 Object Lambda є модифікацією Lambda із швидким доступом до об'єктів у сховищі даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У 2018 році компанією Unbounce було проведено опитування [3] серед 750 споживачів та 395 власників інтернет-магазинів. Кожному із власників було запропоновано протестувати головну сторінку інтернет-магазинів через Google Test My Site для того, аби перевірити наскільки швидко вони завантажуються. У результаті лише 15% сторінок завантажується швидше, ніж 5 секунд. Середня швидкість завантаження сторінок становить 15 секунд. Далі кожному із споживачів запропонували оцінити інтернет-магазини за рядом характеристик. Результати цього опитування показали, що майже 70% споживачів визнають, що швидкість сторінки впливає на їхнє бажання купувати. В той же час лише 10% власників ставлять швидкість сторінки пріоритетом. Основним акцентом для власників є контент і отримання більшого доходу. В той же час, не маючи достатньої оптимізації, вони витрачають більше грошей на підтримку інтернет-магазинів. Саме зображення товарів становлять основну частину кожного інтернет-магазину. Відповідно їх оптимізація може збільшити задоволеність користувачів певним продуктом та покращити їх користувацький досвід.

Метою оптимізації зображень є створення високоякісних зображень з найменшим розміром файлу. Тут грають роль три основні елементи: розмір файлу зображення, рівень компресії, висота і ширина зображення. Прості оптимізації, такі як зміна розміру зображень до того, який потрібен для відображення на сторінці, перетворення зображення в більш ефективний формат, тощо, можуть призвести до значного покращення швидкості завантаження сторінки. Знайшовши баланс між цими трьома елементами, можна зменшити не тільки розмір зображення, але і покращити роботу веб-сайту. Існує п’ять основних стратегій, які можна використовувати для оптимізації файлів зображень.

Першою стратегією є вибір правильного формату зображень. JPEG – найпопулярніший тип зображень, в якому можна налаштувати якість і розмір файлу. Він є універсальним і підходить для більшої кількості випадків [4]. Формат PNG створює зображення високої якості, але також збільшує розмір файлу. GIF

підтримує лише 256 кольорів і забезпечує стиснення без втрат. Цей формат не часто використовується для статичних зображень, але він є більш популярним для анімації.

Другою стратегією є використання прогресивного JPEG та форматів зображень наступного покоління. Формат JPEG має дві форми візуалізації - звичайний та прогресивний. Звичайна форма полягає у підтримці найвищої якості зображення, однак вона може стати проблемою для користувачів з повільним підключенням. Суть прогресивної форми полягає у початковому відображенні низької якості зображення, а потім поступової збільшенні якості по мірі завантаження більшої частини зображення. Використання прогресивного формату JPEG може значно покращити роботу користувачів із повільним підключенням до Інтернету. Іншим варіантом для покращення часу завантаження є формати зображень наступного покоління, такі як WebP і JPEG-XR. Ці формати зображень дозволяють значно знизити розмір файлу завдяки компресії, не погіршуючи якість початкового зображення. Вони рекомендовані для використання в інструкціях Google з оптимізації веб-сайтів. Проблема з форматами наступного покоління полягає в тому, що вони не підтримуються всіма браузерами. Тому для кожного такого зображення необхідно зберігати формат JPEG або PNG і відображати його, якщо браузер користувача не підтримує новий формат.

Третьою стратегією є компресія зображень [5]. Існує два типи компресії: з втратами і без. Компресія з втратами зменшує розмір файлу зображення, але також може погіршити значно якість. Для компресії без втрат головним пріоритетом є якість над розміром файлу. Цей підхід дозволяє відновити початковий файл за потреби, однак значного зменшення розміру файлу не потрібно очікувати.

Наступною стратегією є зміна розміру зображення. Навіть якщо зображення неможливо повністю стиснути, потрібно збалансувати розмір файлу і роздільну здатність. Чим вище роздільна здатність, тим більше розмір файлу. Використання зображень з високою роздільною здатністю сповільнює завантаження сторінки. Аналогічно, якщо відвідувач отримує доступ до вашого веб-сайту через мобільний

телефон, пропускна здатність, ймовірно, буде більш обмеженою, а великі зображення завантажуються довше.

П’ятою і останньою стратегією є оптимізація доставки зображень. Навіть якщо зображення файл та розмір зображення оптимізовані, дотримуючись наведених вище кроків, можна зробити більше, щоб підвищити продуктивність веб-сайтів і мобільних додатків. Можна використовувати мережу доправлення і розповсюдження контенту CDN. За допомогою цього підходу зображення доставляються користувачам відповідно до їхнього фізичного місцезнаходження. Чим ближче сервер, тим швидше час відповіді.

Усі ці стратегії є скоріше порадами, які необхідно брати до уваги при розробці додатків. Однак, вони не є готовими рішеннями, які кожна із компаній може використовувати. З плином часу відсоток візуального контенту у додатках буде тільки збільшуватися. Якщо не використовувати певні підходи до оптимізації зображень, то швидкість завантаження сторінки буде тільки рости, що значно вплине на кінцевих користувачів.

2 АНАЛІЗ МЕТОДІВ ДИНАМІЧНОЇ ОБРОБКИ ЗОБРАЖЕНЬ

Розглянемо існуючі методи вирішення задачі оптимізації зображень на платформі AWS. Умовно поділимо їх на наступні групи:

− архітектурні підходи з точки зору сервісів AWS, які можуть бути використані;

− підходи до збереження даних на S3;

− інструменти для оптимізації зображень;

− методи кешування даних.

AWS (Amazon Web Services) − платформа, яка надає послуги хмарних обчислень [6]. До основних переваг AWS відносять простоту використання, гнучкість, економічність, надійність, масштабованість та безпеку. Платформа підтримує більшість мов програмування та операційних систем. Ви обираєте сервіси, які плануєте використовувати, при цьому сплата відбувається лише за обчислювальну потужність, ємність сховища та інші ресурси, які використовуються у додатку. Сплата за деякі сервіси відбувається лише за той час, коли вони реально використовуються. Сплати за час простоювання немає. Крім того, AWS надає послуги автоматичного масштабування сервісів в залежності від трафіку, який він отримує. На початок 2021 року AWS займає 50 % усього ринку хмарних обчислень. Платформу AWS можна порівняти із віртуальною лабораторією, в якій буде повністю розгорнута інфраструктура проекту [7].

2.1 Архітектурні підходи

Архітектура рішення має велике значення для будь-яких задач, в тому числі для задачі оптимізації зображень. Вона впливає на пропускну здатність (тобто на кількість запитів, які можуть бути виконані одночасно за певний час) та на час виконання запиту. При неправильно спроектованій архітектурі ці показники можуть не відповідати нефункціональним вимогам. Розглянемо три основні

архітектурні підходи, які можуть бути використані для задачі оптимізації генерації зображень.

Перший підхід полягає у використанні сервісу EC2 Elastic Cloud Computing. Цей сервіс надає віртуальні машини, які називаються інстансами. Користувачі можуть обрати операційну системи, тип процесору та кількість ядер, об'єм пам'яті, налаштування мережі, тощо. Основною перевагою EC2 є те, що ви отримуєте "голу" машину, яка може бути налаштована за бажанням та за потребами. Однак, це є і головним його недоліком: за відсутності достатньої кількості навичок, налаштування серверу може зайняти час і ресурси. У сервісі EC2 сплата відбувається за час роботи інстансу. Цей підхід полягає у тому, що створюється повноцінний сервер на EC2 інстансі (див. рис. 1). Він працює 24 години на добу та відповідає за отримання зображення зі сховища даних S3 та генерацію зображення, яке необхідне за якістю та розміром.

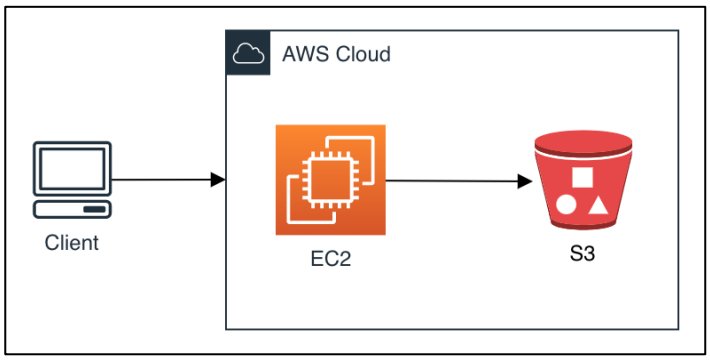


Рисунок 1 − Діаграма архітектури AWS з використанням EC2

На противагу EC2 є сервіс Lambda, який взагалі не потрібно налаштовувати. Він вже надає віртуальну машину на основі операційної системи Linux. Таким чином усі зусилля зосереджуються навколо написання коду. Функції у сервісі Lambda виконуються лише за потреби. Максимальний час життя таких функцій − 15 хвилин. Сплата відбувається лише за час виконання функцій. Якщо функція виконується 5 хвилин, то сплата буде лише за 5 хвилин відповідно. Другий підхід полягає у використанні Lambda функцій. Створюється функція для отримання

даних зі сховища S3 та генерації оптимізованого зображення (див. рис. 2). Цей підхід є кращим, порівняно з першим як мінімум за ціною його реалізації.

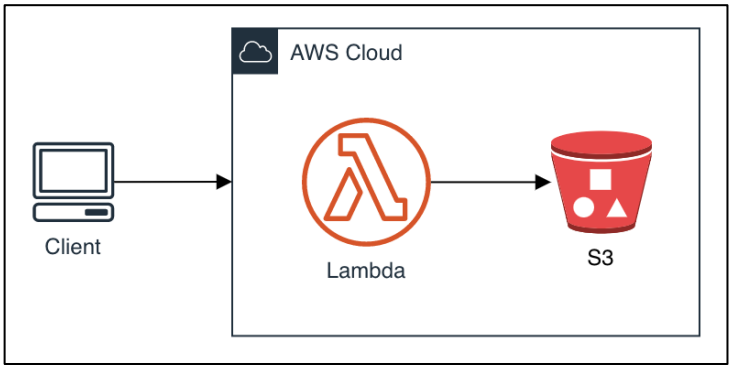


Рисунок 2 – Діаграма архітектури AWS з використанням Lambda

Альтернативним і відносно новим підходом, який можна використати для вирішення задачі з точки зору архітектури є сервіс S3 Object Lambda. AWS анонсували його у березні 2021 року. Головною перевагою цього сервісу є те, що він має прямий доступ до сховища S3. У традиційній реалізації Lambda + S3 або EC2 + S3, доступ до сховища відбувається по публічній мережі Інтернет. Публічна мережа працює повільніше, ніж приватна. Використання S3 Object Lambda вирішує цю проблему. Завдяки цьому сервісу Lambda має локальний доступ до сховища, а це означає, що час з'єднання є мінімальним (див. рис. 3).

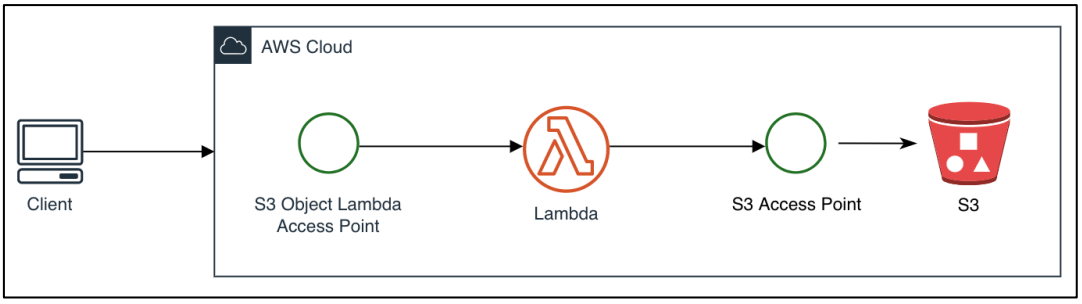


Рисунок 3 – Діаграма архітектури AWS з S3 Object Lambda

Суть цього підходу полягає у тому, що ми вводимо дві Lambda функції: перша відповідає за доступ до зображення, друга – за генерацію оптимізованого. Друга функція викликається автоматично при спробі першої функції отримати

доступ до зображення. Перевагою цього підходу є те, що ми виграємо на часі, який необхідно витратити для доступу до зображення. Крім того, з точки зору дизайну ми розділяємо відповідальності між сервісами.

2.2 Підходи до збереження даних на S3

AWS S3 − сервіс-сховище даних, головними перевагами якого є легкість, зручність і ціна використання.

Уявимо ситуацію, коли для якогось додатку необхідно декілька версій одного зображення, наприклад с шириною 1000, 600, 250 і 100 пікселів. Для швидкої доставки зображень клієнтам можна для кожного зображення зберігати усі необхідні його версії, початкову включно. Це забезпечить найшвидший доступ до них, оскільки не треба витрачати додаткового часу на їх динамічну обробку. Припустимо, що таких зображень мільйон, або мільярд. Таким чином розмір сховища значно виросте, а і відповідно ціна його використання. Цей підхід називається прегенерація зображень. Як бачимо, його можна вдосконалити, оскільки він гарно працює на маленькій кількості файлів, однак не є розширюваним. Другим підходом є зберігання лише початкових зображень і динамічна обробка кожного із зображень за запитом. Ми економимо на ціні використання сховища, однак витрачаємо більше на обробку. Важливо знайти баланс між ціною за сховище і ціною за комп’ютінг.

Альтернативним підходом є використання такого функціоналу S3 як Lifecycle rules (див. рис. 4). Суть підходу полягає у тому, щоб виявити ті розміри зображень, які будуть найчастіше використовуватися, і зберігати їх тимчасово у сховищі. При цьому для кожного об’єкту виставляється час життя. Як тільки час життя закінчується, він автоматично видаляється. Для розмірів зображень, які використовуються частіше − виставляємо більший час життя. Для тих, що зовсім рідко − взагалі не зберігаємо їх і завжди динамічно генеруємо. Таким чином ми враховуємо і ціну за сховище та ціною за комп’ютінг. Ціна не така велика

порівняно із ситуацією, коли ми зберігаємо усі картинки, а зображення ми генеруємо лише тоді, коли воно відсутнє у сховищі (див. рис. 4).

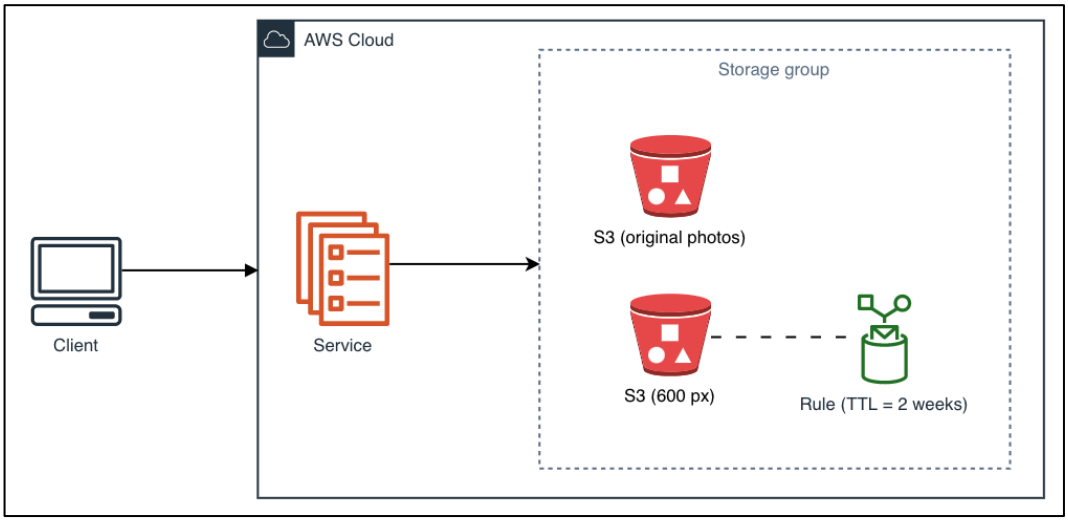


Рисунок 4 − Діаграма архітектури AWS з використанням TTL для S3 об’єктів

Менш оптимізованим підходом є збереження усіх розмірів зображень наприклад на декілька годин. Таким чином тільки для перших користувачів час запиту буде великим, оскільки необхідно динамічно згенерувати зображення. Однак для більшої кількості час відповіді запиту все ще буде досить малим.

2.3 Інструменти для оптимізації зображень

Існує три напрямки основні напрямки оптимізації зображень: конвертація формату, ресайзинг та компресія. Конвертація формату полягає у зміні формату зображення, наприклад з PNG до JPEG. Ресайзинг − зміна розмірів зображення, тобто його висоти і ширини. Компресія − зміна розміру файлу зображення без значного впливу на його якість. Розглянемо детально інструменти, які використовують для кожного із напрямків, їх продуктивність та бенчмарки.

ImageMagick − безкоштовний, крос-платформенний інструмент з відкритим кодом для роботи з зображеннями. Він не має графічного інтерфейсу та складається із набору CLI команд. Він включає команди для ресайзингу, обрізання, ротації

зображень, конвертацію формату, накладання тексту і інших об’єктів, створення GIF анімацій та багато іншого. Цей інструмент признано стандартом і він використовується у великій кількості додатків [8]. На противагу ImageMagick є інструмент Sharp, який з’явився відносно нещодавно. Його функціонал дещо обмежений, порівняно з ImageMagick, але він підтримує основні команди для маніпуляції з зображеннями. Основною його перевагою є те, що він працює в 4-5 разів швидше, ніж використання найшвидших налаштувань ImageMagick. В основі модуля Sharp лежить надзвичайно швидка бібліотека обробки зображень libvips. На рисунку 5 зображено бенчмарки усіх популярний бібліотек для роботи з зображеннями [9].

Module	Input	Output	Ops/sec	Speed-up
jimp	buffer	buffer	0.83	1.0
squoosh-cli	file	file	1.09	1.3
squoosh-lib	buffer	buffer	1.83	2.2
mapnik	buffer	buffer	3.41	4.1
gm	buffer	buffer	8.34	10.0
imagemagick	file	file	8.67	10.4
gm	file	file	8.82	10.6
sharp	stream	stream	29.44	35.5
sharp	file	file	29.64	35.7
sharp	buffer	buffer	31.09	37.5

Рисунок 5 − Порівняння бібліотек для маніпуляції з зображеннями

Як бачимо, Sharp може обробляти приблизно 30 операцій в секунду, в той час як ImageMagick лише 8. Прискорення майже у 4 рази. При цьому, у Sharp оптимізована робота з пам’яттю, що є ще однією його перевагою порівняно з ImageMagick.

Для компресії зображень існує безліч інструментів. Найпопулярнішим та найбільш широко використовуваним є JPEGmini. Головною його особливістю є компресія без втрат, тобто початкова якість зображення зберігається. Крім того, він

підтримує зображення у будь-якій роздільній здатності до 50 мегапікселів, використовує всі ядра процесора одночасно, що робить його досить швидким.

2.4 Методи кешування даних

AWS CloudFront – це сервіс, який пришвидшує доставку користувачам статичного та динамічного контенту. CloudFront доставляє контент через всесвітню мережу центрів обробки даних, які називаються Edge locations. Коли користувач робить запит, який обслуговується за допомогою CloudFront, він направляється до найближчої Edge location (див. рис. 6). Це забезпечує найнижчу затримку, тобто контент доставлявся з найбільшою можливою продуктивністю до кінцевого користувача.

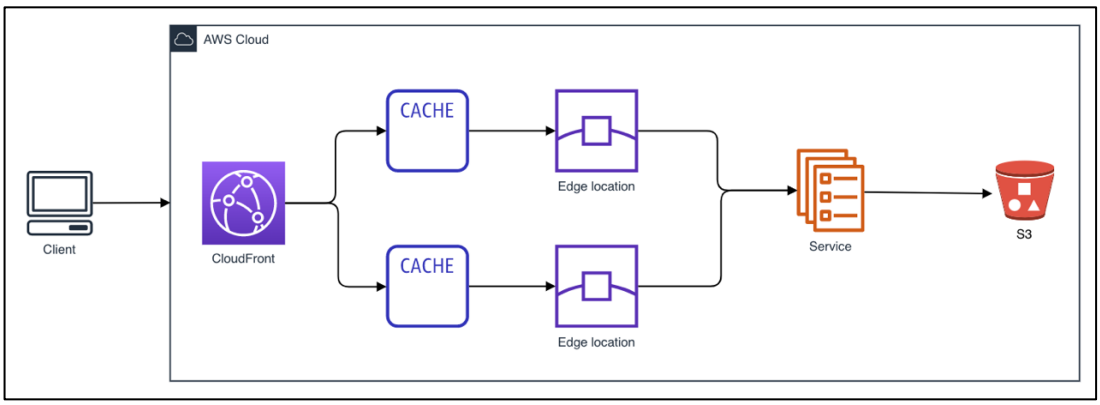


Рисунок 6 – Діаграма архітектури AWS з використанням Edge locations

Суть цього методу полягає у використанні CloudFront та Edge locations. Якщо запит на контент не є закешованим, то він перенаправляється на сервер. Сервер генерує контент, який кешується на локації. У противному випадку контент відразу доставляється до користувача.

3 ПОСТАНОВКА ЗАДАЧІ

У кваліфікаційній роботі методи динамічної обробки зображень будуть вирішуватися на прикладі типової задачі про інтернет-магазин. Нехай інтернет-магазин має N товарів. У середньому в інтернет-магазині може розміщуватися одночасно 5000 товарів. Кількість товарів N може змінюватися як в більшу так і в меншу користь. Інтернет-магазином користуються споживачі з усього світу, однак переважна кількість із них з Європи. Для кожного із товарів використовуються зображення у наступних розмірах: 1000, 600 і 100 пікселів. При цьому зображення 1000 і 600 пікселів використовуються у ході основного шляху користувача − при перегляді товарів інтернет-магазину. Зображення 100 пікселів використовується лише при оформленні замовлення товару і подальшої його сплати. Переважна кількість користувачів лише переглядають товари, і лише приблизно відсотків 20 доходить до оформлення замовлення. Початкові зображення товарів можуть бути завантажені у форматі JPEG.

Трафік інтернет-магазину є нерівномірним. Зранку та вдень максимальна одночасна кількість користувачів на сайті приблизно 1500. Ближче до вечора це число збільшується майже у 2 рази до приблизно 2800. Напередодні великих свят або у період знижок трафік зростає майже у 5 разів. Тобто 7500 вранці та вдень, 14 000 ввечері. Для споживачів важливий їх користувацький досвід і мінімально можливий час завантаження зображень.

Відповідно до проведеного аналізу переметної області, у кваліфікаційній роботі необхідно розглянути наступні методи вирішення задачі динамічної обробки зображень чи їх оптимізації:

− AWS сервіс, на якому буде розгорнута інфраструктура: Lambda, S3 Object Lambda, EC2;

− з точки зору інструментів для зміни розміру зображення: ImageMagick або Sharp;

− увімкнена або ні компресія зображень за допомогою інструменту JPEGmini;

− з точки зору оптимізації сховища: збереження усіх зображень у сховище

S3, збереження лише початкових зображень, збереження зображень у сховище с різним часом життя;

− з точки зору кешування: використання CloudFront та Edge locations.

Метою кваліфікаційної роботи є виявлення комбінації найбільш ефективних методів динамічної обробки зображень на прикладі вирішення типової задачі. Для досягнення поставленої мети необхідно виконати наступні етапи роботи:

− реалізувати кожен із методів динамічної обробки зображень;

− сформувати апарат оцінки їх ефективності, враховуючи продуктивність системи, споживання пам’яті, ціну використання;

− провести оцінку кожного із методів за попередньо визначеними критеріями;

− проаналізувати, як комбінації між методами впливають на фінальний результат доставки зображень для користувачів;

− зробити висновки та надати перелік рекомендацій про доцільність використання кожного із них в залежності від вимог.

4 КРИТЕРІЇ ОЦІНКИ МЕТОДІВ ДИНАМІЧНОЇ ОБРОБКИ ЗОБРАЖЕНЬ

Оцінка кожного із методів вирішення поставленої задачі буде відбуватися за наступними критеріями: час генерації зображення, час доставки зображення для користувача, пропускна спроможність системи при різних навантаженнях, кількість пам’яті, що використовується та ціни щомісячної підтримки рішення на платформі AWS.

4.1 Оцінка часу генерації зображення

Час генерації зображення − це основна метрика, яка безпосередньо впливає на швидкість доставки кінцевого зображення користувачеві. Час генерації залежить від багатьох факторів, наприклад як частота процесору, кількість оперативної пам’яті, інструменту для зміни розміру, а головне від розміру і ваги зображення. Для того, або експеримент зробити максимально наближеним до реалій світу, ми будемо розглядати набір зображень різного розміру і ваги. Для кожного зображення отримуємо час до початку та у кінці операції зміни розміру. Вимірювання часу буде відбуватися за допомогою обраної мови програмування і зберігатиметься у вигляді метрики до бази даних. Формула для знаходження середнього часу генерації певного розміру вихідного зображення має вигляд (7).

$$t_k = \frac{\sum_1^N (t_{\text{кін}} - t_{\text{поч}})}{N}, \tag{7}$$

де $t_{\text{кін}}$ − час завершення операції;

$t_{\text{поч}}$ − час початку операції;

t_k − час генерації зображення розміром k, ;

t_k − кількість початкових зображень для оцінки.

4.2 Оцінка часу доставки зображення до користувача

Час доставки зображення до користувача є багатокомпонентною метрикою, яка залежить від інфраструктури розгорнутої системи, поточного навантаження та місця знаходження користувача.

По-перше, при розгортанні високонавантажених систем, кожен із серверів має декілька клонів для більш швидкої обробки запитів. Розповсюдженою практикою є розгортання серверів системи по декільком локаціями. Так наприклад, якщо ми знаємо, що 90% користувачів системи знаходяться в Європі, то має сенс розгорнути 9 із 10 серверів саме у Європі. Завдяки тому, що сервер, з яким буде взаємодіяти клієнт, буде знаходитися максимально близько до нього, час доставки контенту є мінімальним.

По-друге, одним із основних практик високонавантажених систем є кешування даних. При отриманні запиту сервер перш за все перевіряє, чи закешований контент, який клієнт намагається запросити. Якщо так, то користувач отримає закешований контент. У противному випадку сервер обробляє запит. Два основних сервіси, які дозволяють реалізувати задану логіку у рамках AWS є CloudFront та Edge locations. Edge locations − це центри обробки даних AWS, розроблені для надання послуг із мінімальною затримкою. CloudFront кешує дані на Edge locations.

У загальному випадку, коли кешування не налаштовано, час доставки зображення до користувача розраховується за формулою (8).

$$t = t_{client} + t_{cache} + t_{server} + t_{location}\ , \tag{8}$$

де t_{client} − час обробки запита клієнтом;

t_{cache} − час пошуку контенту в кеші;

t_{server} − час обробки сервером запиту;

$t_{location}$ − час пошуку найближчої локації.

Коли кешування налаштоване, час запиту значно зменшується, оскільки в цьому випадку запит не обробляється сервером (9).

$$t = t_{client} + t_{cache} + t_{location}\ , \tag{9}$$

де t_{client} – час обробки запита клієнтом;

t_{cache} – час пошуку контенту в кеші;

$t_{location}$ – час пошуку найближчої локації.

У рамках експериментів нам важливий загальний час доставки зображення до користувачів. AWS у свою чергу не надає інформацію про час пошуку контенту в кеші і час пошуку найближчої локації. Тому ми будемо використовувати інструмент k6. Це платформа, яка дозволяє проводити тестування навантаження систем у максимальному наближенні до реальних умов. Він є досить легким у використанні, має зручний користувацький інтерфейс, а головне, за допомогою нього можна зробити висновки щодо поведінки систем при навантаженнях (див. рис. 10).

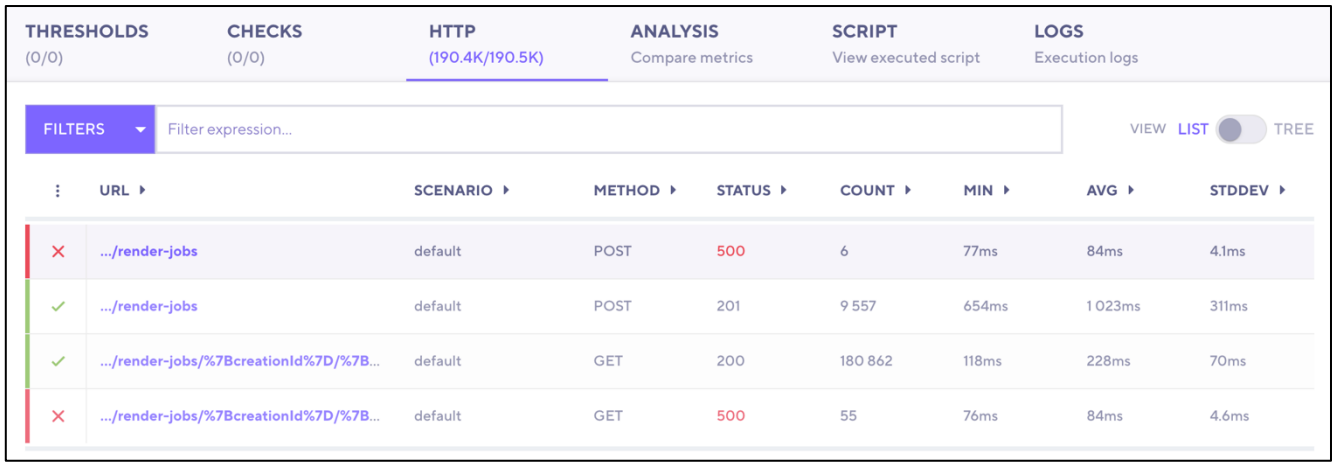


Рисунок 10 – Користувацький інтерфейс інструменту k6

У рамках експериментів ми створимо скрипт із запитами для отримання кожного зображення. За допомогою k6 ми отримаємо статистику по часу виконання запитів: мінімальний, максимальний та середній час у мілісекундах.

4.3 Оцінка пропускної спроможності системи при різних навантаженнях

Пропускна спроможність системи − це кількість запитів у секунду, які вона може обробляти без значних втрат по продуктивності та впливу на час виконання запитів. Маючи інформацію про кількість одночасних активних користувачів системи, за допомогою того ж інструменту k6 можна симулювати трафік, схожий на реальну поведінку користувачів (див. рис. 11).

Рисунок 11 − Симуляція віртуальних користувачів за допомогою k6

Інструмент k6 дозволяє повноцінно описати поведінку користувачів та симулювати трафік, плавно його нарощуючи чи зменшуючи. У контексті k6 кожен активний користувач називається віртуальним. За допомогою простої конфігурації можна зазначити, яка кількість віртуальних користувачів у який проміжок часу буде використовувати систему. Наприклад, протягом перших 15 хвилин у системі буде 300 віртуальних користувачів, далі 500 протягом 10 хвилин та 800 протягом наступних 15 хвилин. Із рисунку 13 можемо побачити, що k6 збирає метрику PEAK RPS. Це і є максимальна пропускна здатність системи, тобто та метрика, яку ми будемо використовувати при порівнянні різних методів обробки зображень.

4.4 Оцінка кількості пам’яті, що використовується

Одним із основних сервісів AWS є CloudWatch − сервіс для збереження журналів логів протягом останнього місяця. Він легко інтегрується з основними сервісами, на яких може бути розгорнута інфраструктура: EC2, Lambda, S3 Object Lambda та інші. Для інтерактивного пошуку і аналізу журналів логів у CloudWatch є спеціальний сервіс CloudWatch Log Insights. Він автоматично розбирає повідомлення по ключовим словам та надає можливість використовувати спеціальну мову запитів для їх аналізу.

Одною із метрик, яку можна отримати за допомогою CloudWatch Log Insights є кількість пам’яті, що використовується сервісом у межах певного запиту. На рисунку 12 наведено приклад запиту, який може бути використаний для аналізу кількості використаної пам’яті.

```
filter @type = "REPORT"
| fields
        @timestamp as Timestamp,
        @requestId as RequestID,
        @logStream as LogStream,
        @memorySize/1000000 as MemorySetInMB,
        @maxMemoryUsed/1000000 as MemoryUsedInMB
| filter MemoryUsedInMB > 2048
| sort MemoryUsedInMB desc
| head 10
```

Рисунок 12 − Запит для аналізу пам’яті у CloudWatch Log Insights

Звернемо увагу на поля @memorySize та @maxMemoryUsed. Поле @memorySize відповідає за розмір оперативної пам’яті, яка надана сервісу за допомогою початкової його конфігурації. Поле @maxMemoryUsed відповідає за фактичний розмір пам’яті, що використовується. Максимальний розмір пам’яті, який можна встановити на AWS Lambda − 10 GB, на самому потужному EC2 інстансі − 24 TiB.

У результаті виконання запиту отримаємо статистику по кількості пам’яті, що використовується для кожного із запитів (див. рис. 13). Червоним прямокутником безпосередньо позначено розмір пам’яті у MB.

Рисунок 13 − Запит для аналізу пам’яті у Cloudwatch log insights

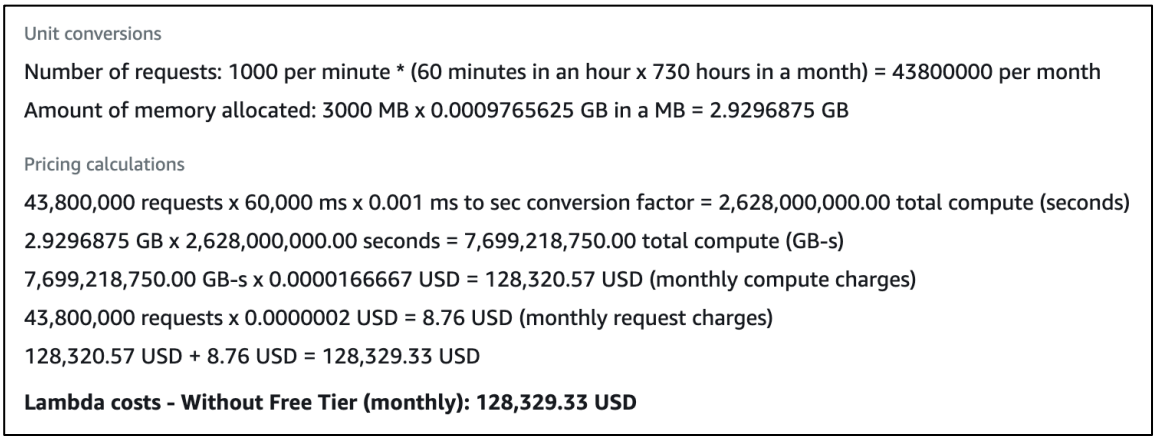
Саме сервіс CloudWatch Log Insights буде використовуватися для оцінки кількості пам’яті у рамках кваліфікаційної роботи. Для кожного із сервісів (EC2, AWS Lambda, S3 Object Lambda) будемо підраховувати середнє значення для зображень різних розмірів.

4.5 Оцінка ціни щомісячної підтримки рішення на платформі AWS

Однією із основних переваг AWS є його цінова політика. Сплата відбувається лише за ті сервіси та протягом того часу, в який вони використовуються. AWS не вимагає підписання довгострокових контрактів чи комплексного ліцензування. Для більшості клієнтів доступні знижки по мірі збільшення використання сервісів чи об’єму даних, що обробляються. Так наприклад на сервісі S3 ціна є не фіксованою, а динамічною. Тобто чим більше ви використовуєте сховище даних, тим менше ціна за один ГБ.

Для оцінки ціни щомісячної підтримки рішення на платформі AWS будемо використовувати AWS Pricing Calculator. Він дозволяє ввести основні дані конфігурації для більшості сервісів та легко прорахувати ціну використання. Наведемо приклади розрахунків ціни використання для наступних сервісів: Lambda та S3. Варто зазначити, що дані розрахунки є лише приблизними, оскільки конфігурація сервісів потребує час для того, аби знайти баланс між найліпшими налаштуваннями на найбільш прийнятною ціною. Для цього існують окремі інструменти, наприклад Lambda Powering tool [10] та інші.

Розглянемо сервіс AWS Lambda. Сплата за сервіс Lambda відбувається лише за той час, коли він реально використовується. Наприклад, якщо функція викликатиметься раз на місяць протягом 5 хвилин, то сплата буде відбуватися лише за ці 5 хвилин. Ціна використання AWS Lambda залежить від: кількості пам’яті, кількості запитів та часу виконання одного запиту. При 3000 МБ пам’яті, 1000 запитів на хвилину та часу запиту одна хвилина, ціна використання сервісу на місяць становить 128,329 доларів (див. рис. 14).


Unit conversions

Number of requests: 1000 per minute * (60 minutes in an hour x 730 hours in a month) = 43800000 per month

Amount of memory allocated: 3000 MB x 0.0009765625 GB in a MB = 2.9296875 GB

Pricing calculations

43,800,000 requests x 60,000 ms x 0.001 ms to sec conversion factor = 2,628,000,000.00 total compute (seconds)

2.9296875 GB x 2,628,000,000.00 seconds = 7,699,218,750.00 total compute (GB-s)

7,699,218,750.00 GB-s x 0.0000166667 USD = 128,320.57 USD (monthly compute charges)

43,800,000 requests x 0.0000002 USD = 8.76 USD (monthly request charges)

128,320.57 USD + 8.76 USD = 128,329.33 USD

Lambda costs - Without Free Tier (monthly): 128,329.33 USD

Рисунок 14 − Розрахунок вартості Lambda

Розглянемо сервіс AWS S3, який дозволяє зберігати дані у хмарі. Ціна його використання залежить від об’єму пам’яті, який будуть займати безпосередньо файли, кількості запитів на завантаження та їх отримання і об’єму пам’яті, який буде сканувати S3 для повернення необхідної інформації.

Так при розмірі сховища, що дорівнює 10 ГБ на місяць, 8000 запитів на завантаження файлів та 500 000 запитів на їх отримання, ціна за місяць використання становить 26 доларів (див. рис. 15).

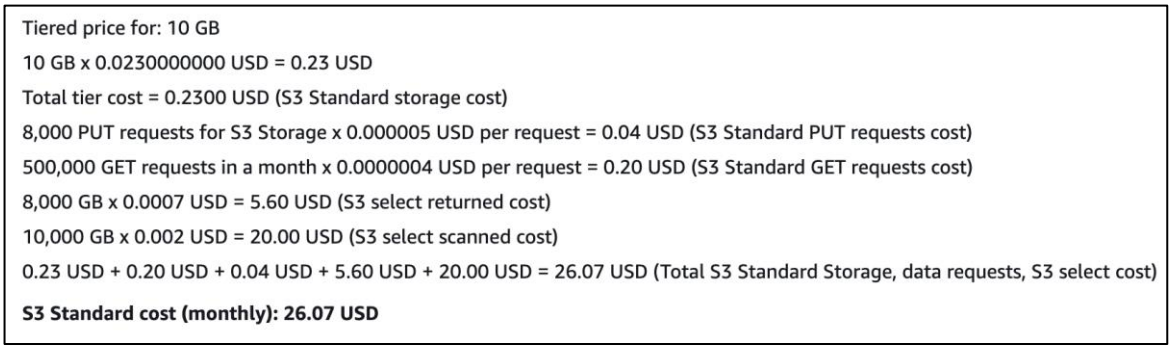


Рисунок 15 − Розрахунок вартості S3

Як бачимо, AWS має досить приємну цінову політику незважаючи на об’єми його використання. Саме такий підхід буде використовуватися при підрахунку ціни щомісячної підтримки рішення. До сервісу AWS Pricing Calculator буде вводитися поточне налаштування сервісів, задані об’єми навантаження та розраховуватиметься остаточна ціна.

5 ОПИС ПРОВЕДЕНИХ ЕКСПЕРИМЕНТІВ

Для оцінки результатів роботи буде проведено 8 експериментів у 3 раунди. Метою першого раунду є виявити AWS сервіс, який найліпше підходить до вирішення заданого типу задач: Lambda, EC2 чи S3 Object Lambda. Метою другого раунду є вибір найкращого інструменту обробки зображення: ImageMagick або Sharp. Метою третього раунду є виявити доцільність використання додаткових оптимізацій. До них відносяться компресія зображень за допомогою інструменту JPEGmini, оптимізація сховища (збереження усіх зображень у сховище S3, збереження лише початкових, збереження зображень у сховище с різним часом життя), кешування за допомогою використання сервісів CloudFront та Edge locations. Перелік експериментів та їх налаштування занесено до таблиці 16.

Таблиця 16 − Опис експериментів

Раунд	**Номер**	**Метод оптимізації**				
		AWS сервіс	Інструмент	Компресія	Кешування	Оптимізація сховища
1	1	Lambda	Sharp	-	-	-
	2	EC2	Sharp	-	-	-
	3	S3 Object Lambda	Sharp	-	-	-
2	4	Lambda	ImageMagick	-	-	-
	5	Lambda	Sharp	+	-	-
3	6	Lambda	Sharp	-	+	-
	7	Lambda	Sharp	-	-	+
	8	Lambda	Sharp	+	+	+

Для реалізації методів оптимізації динамічної обробки зображень використовуватимо платформу Node.js та фреймворк Serverless. Serverless дозволяє

швидко і зручно створювати ресурси на AWS, використовуючи звичайний код [11]. AWS надає простий і гейміфіковайний інтерфейс [12] для створення ресурсів, однак більшість надає перевагу підходу infrastructure as a code. Для симуляції трафіка використовуємо інструмент k6. Протягом пів години навантажуємо систему: 10 хвилин симулюємо 200 віртуальних користувачів, наступні десять − 500, останні − 800. Один віртуальний користувач робить запит на 10 зображень.

5.1 Визначення доцільного AWS сервіса

У першому експерименті основним сервісом, що використовується, є AWS Lambda. За допомогою інструменту Sharp генеруємо зображення розміром 1000, 600 і 100 пікселів. Додаткових оптимізацій не передбачено. На рисунку 17 наведено архітектуру даного експерименту.

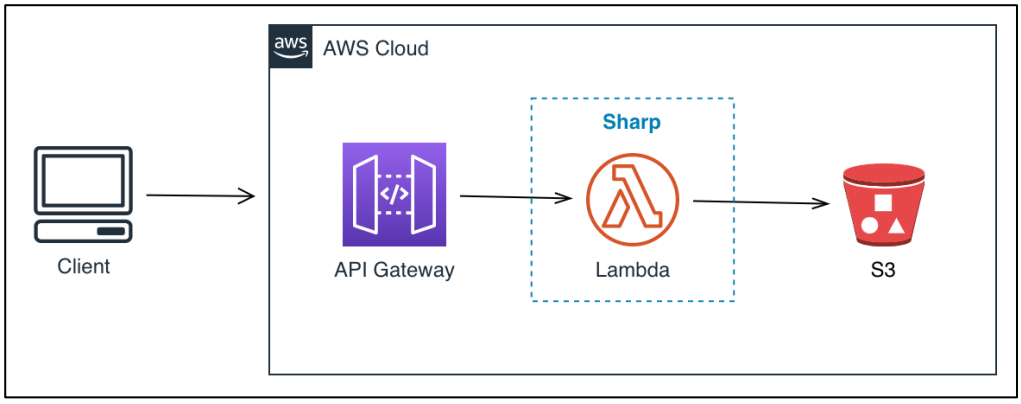


Рисунок 17 − Діаграма архітектури AWS з використанням Lambda

Lambda має наступні налаштування:

− операційна система Amazon Linux 2, x86_64, arm64;

− середовище виконання: Node.js14.x;

− 3 ГБ оперативної пам’яті;

− паралельність (concurrency) − 1500 інстансів;

− максимальний час виконання функції становить 3 хвилини.

Виміряємо показники ефективності даного рішення. Бачимо, що розмір зображення прямо пропорційно залежить від часу його генерації та часу його доставки: чим більше зображення, тим вищі показники часу. Кількість пам’яті, що використовується, здебільшого залежить від розміру початого та розміру остаточного зображення. Для того, аби змінити його розмір, зображення перш за все потрібно завантажити у локальну файлову систему. Тому мінімальна кількість пам’яті буде приблизно дорівнювати середньому розміру зображення. Ціна за рішення на AWS включає вартість сервісу Lambda з поданими вище налаштуваннями та API Gateway. Результати першого експерименту занесено до таблиці 18.

Таблиця 18 − Результати експерименту Lambda + Sharp

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	1.12 с	0.85 с	0.48 с
Час доставки зображення	1.52 с	1.28 с	0.88 с
Максимальна кількість запитів	1022 зап/хв	1232 зап/хв	1385 зап/хв
Кількість пам’яті	1725 МБ	1572 МБ	1336 МБ
Ціна рішення на AWS	8 USD	8 USD	8 USD
Ціна рішення S3	5.27 USD	4.62 USD	3.51 USD

Другий експеримент аналогічний першому, однак замість Lambda використовуємо сервіс EC2, який буде працювати 24 години на добу. EC2 має наступні налаштування:

− тип інстансу: m5n.xlarge;

− процесор Intel Xeon Scalable Processors, vCPU 4 ядра, частота 3.1-3.5 ГГц;

− 16 ГБ оперативної пам’яті;

− операційна система Ubuntu 20.

Цей тип інстансу є одним із базових і відноситься до групи загального призначення. У ньому збалансовано обчислювальні, мережеві ресурси та пам’ять.

Однією із його переваг є підвищена пропускна здатність передачі пакетів. На рисунку 19 наведено архітектуру даного експерименту.

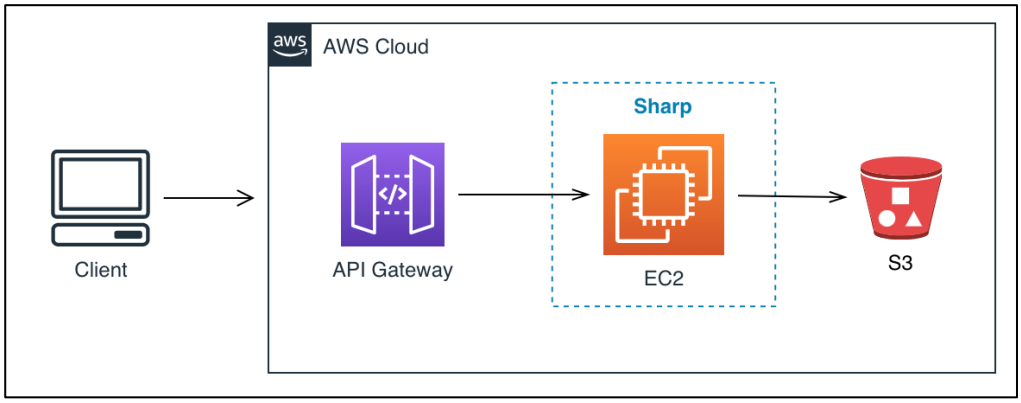


Рисунок 19 – Діаграма архітектури AWS з використанням EC2

Вимірявши показники ефективності, бачимо, що час генерації зображення значно зменшився. Це пов'язано із тим, що сервіси EC2 мають значно більший можливий об'єм пам'яті для виконання операцій. У сервісі Lambda він є обмеженим. Час доставки зображення, кількість пам'яті та ціна за сховище S3 залишилися на тому рівні, що і у попередньому експерименті. Варто звернути увагу на те, що максимальна кількість запитів на хвилину зменшилася, а ціна рішення на AWS зросла у рази. Це залежить від типу обраного EC2 інстансу. Якщо ми би обрали більш ефективний тип, то кількість запитів були би більшою. Однак і ціна вищою аналогічно. Результати другого експерименту занесемо до таблиці 20.

Таблиця 20 – Результати експерименту EC2 + Sharp

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	0.83 с	0.32 с	0.05 с
Час доставки зображення	1.63 с	1.41 с	0.90 с
Максимальна кількість запитів	702 зап/хв	823 зап/хв	934 зап/хв
Кількість пам'яті	1532 МБ	1376 МБ	1214 МБ
Ціна рішення на AWS	61.54 USD	61.54 USD	61.54 USD
Ціна рішення S3	5.27 USD	4.62 USD	3.51 USD

Відповідно до результатів, отриманих у першому і другому експерименті можемо зробити висновок, що генерація зображень на сервісі EC2 відбувається швидше, але час доставки зображення до кінцевого користувача довше. Це відбувається тому що Lambda з самого початку спроектована таким чином, що вона більш легковісна. Вона запускається швидко, не потребує багато ресурсів, а відповідно може обробити більшу кількість запитів на хвилину. Крім того, бачимо, що ціна використання сервісу EC2 майже в 4 рази більше, ніж Lambda.

Основною перевагою S3 Object Lambda є її швидкий доступ до об'єктів у сховищі S3. За всіма іншими показниками це є та ж сама Lambda. Третій експеримент проведемо саме з використанням цього сервісу (див. рис. 21). Конфігурацію лямбди залишимо такою ж як і у першому експерименті.

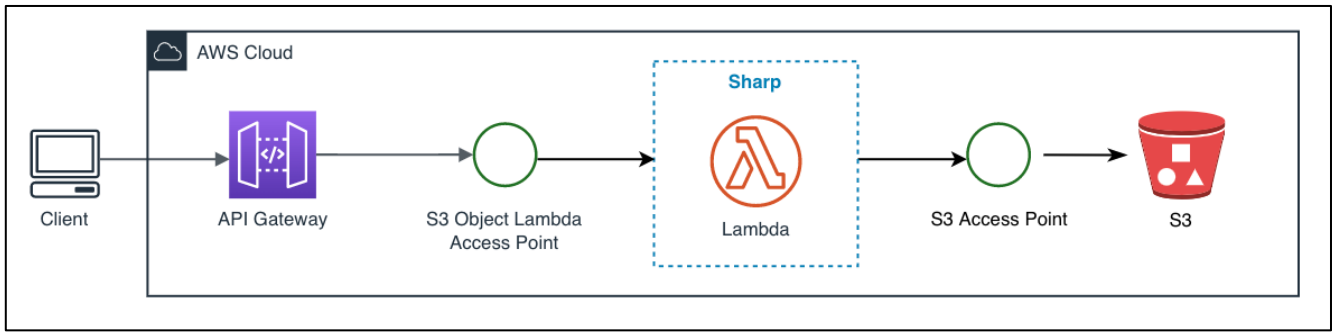


Рисунок 21 – Діаграма архітектури AWS з використанням S3 Object Lambda

Відповідно до отриманих результатів у ході експерименту (див. табл. 22) бачимо, що результати приблизно однакові з тими, що ми отримали у експерименті з Lambda функцією.

Таблиця 22 – Результати експерименту S3 Object Lambda + Sharp

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	1.03 с	0.76 с	0.37 с
Час доставки зображення	1.56 с	1.34 с	0.95 с
Максимальна кількість запитів	1121 зап/хв	1301 зап/хв	1455 зап/хв
Кількість пам'яті	1748 МБ	1562 МБ	1349 МБ

Кінець таблиці 22

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Ціна рішення на AWS	8.5 USD	8.5 USD	8.5 USD
Ціна рішення S3	5.27 USD	4.62 USD	3.51 USD

Дійсно, час генерації зображення зменшився приблизно на 5%. Однак і ціна рішення збільшилася приблизно на 6-8%. Це відбувається завдяки тому, що для забезпечення швидкого доступу до об’єктів сховища, необхідні і додаткові ресурси, за які здійснюється окрема сплата. На відміну від Lambda, яка завантажує зображення по загальній AWS мережі (приватній або публічній, не має різниці), S3 Object Lambda має прямий доступ до об’єктів, що і пришвидшує доступ. Саме на моменті завантаження зображення ми і виграємо час.

Провівши перші три експерименти, можемо зробити висновок про найбільш підходящий сервіс, який стане основою подальших експериментів. Незважаючи на те, що у другому експерименті з використанням EC2 ми виграємо трішки на продуктивності, однак ціна підтримки цього рішення на сервісі в рази більша, ніж на Lambda. Щодо S3 Object Lambda, висновок доволі суперечливий, оскільки невеликий виграш на швидкості генерації і дещо завищена ціна робить використання цього сервісу приблизно аналогічним Lambda. Беручи це до уваги, продовжимо подальші експерименти на сервісі Lambda, оскільки маємо на меті досягти найкращу і найбільш збалансовану комбінацію методів динамічної обробки зображень для подальшої їх рекомендації.

5.2 Вибір інструменту для обробки зображень

За основу третього експерименту візьмемо перший з використанням Lambda. Конфігурація сервісу залишається такою ж. Однак у якості інструменту для генерації зображень використовуємо застосунок ImageMagick. Відповідно до

документації, Sharp повинен працювати швидше і потребує набагато менше ресурсів, аніж ImageMagick. Архітектура рішення на AWS наведена на рисунку 23.

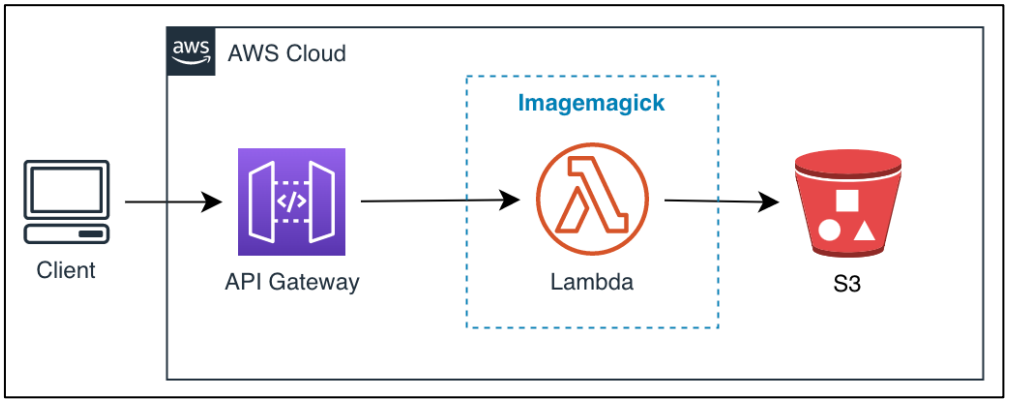


Рисунок 23 – Діаграма архітектури AWS з використанням Lambda та ImageMagick

Результати третього експерименту занесено до таблиці 24. Як бачимо, час генерації зображень для кожного розміру збільшився у більше, ніж 2 рази. Тобто Sharp дійсно працює швидше, ніж ImageMagick. Крім того, варто звернути увагу на те, що і кількість пам'яті, що використовується, теж збільшилася. Це відбувається тому, що окрім розміру самого файлу, який зберігається у файловій системі Lambda, додаткові ресурси витрачаються ще на команди зміни розміру зображення.

Таблиця 24 – Результати експерименту Lambda + ImageMagick

Показник	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	2.3 с	1.89 с	0.93 с
Час доставки зображення	1.64 с	1.39 с	0.92 с
Максимальна кількість запитів	1099 зап/хв	1205 зап/хв	1343 зап/хв
Кількість пам'яті	1844 МБ	1658 МБ	1395 МБ
Ціна рішення на AWS	8 USD	8 USD	8 USD
Ціна рішення	5.27 USD	4.62 USD	3.51 USD

У ході перший двох раундів експериментів можемо зробити висновок, що сервіс Lambda доцільно використовувати з інструментом Sharp для отримання

кращих результатів. Це вже і є готове рішення. Однак, його можна покращити, реалізувавши додаткові методи оптимізації, такі як компресія, кешування чи переосмислення правил збереження зображень у S3.

5.3 Використання додаткових оптимізацій

У третьому раунді експериментів основний фокус покладається на подальші оптимізації. Проведемо четвертий експеримент, де окрім інструменту Sharp використовуємо JPEGmini на етапі завантаження зображень у S3. Відповідно до документації в залежності від налаштувань інструменту можна досягти до 80% зменшення зображення на S3 без значної втрати якості (див. рис. 25). Запускаємо інструмент JPEGmini з наступними налаштуваннями:

– для зображення 1000 пікселів якість вихідного зображення найкраща, для 600 пікселів висока, для 100 пікселів середня;

– опція progressive ввімкнена, де дозволить браузеру завантажувати зображення поступово по мірі надходження нових даних.

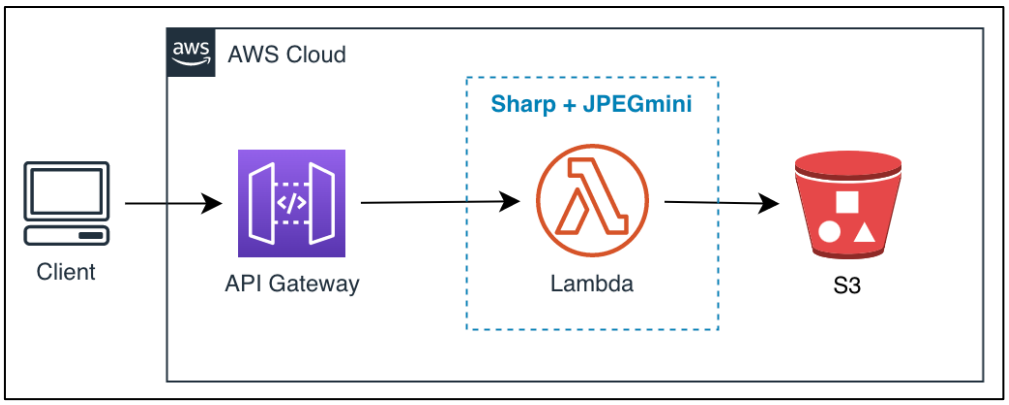


Рисунок 25 – Діаграма архітектури AWS з компресією зображення з JPEGmini

Завдяки налаштуванням JPEGmini зазначеним вище, загальний розмір 5000 файлів, збережений у трьох різних форматах, в S3 зменшився з 8.4 до 4.9 ГБ, тобто майже на 42%, що вже є значним покращенням.

Подивимось, як JPEGmini вплине на фінальні значення критеріїв ефективності (див. табл. 26).

Таблиця 26 − Результати експерименту Lambda + Sharp + JPEGmini

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	1.34 с	0.94 с	0.58 с
Час доставки зображення	1.02 с	0.81 с	0.64 с
Максимальна кількість запитів	1147 зап/хв	1294 зап/хв	1358 зап/хв
Кількість пам’яті	1519 МБ	1327 МБ	1269 МБ
Ціна рішення на AWS	8 USD	8 USD	8 USD
Ціна рішення S3	2.83 USD	2.01 USD	1.59 USD

У результаті отримали досить цікаві результати. Перш за все, є незначне збільшення у часі генерації зображення. Це є досить логічним, оскільки ми додали додаткову команду для зменшення ваги зображення. По друге, є значне зменшення часу доставки зображення до користувача. Аналогічно, це пов’язано зі зменшенням ваги зображення. Чим вона менша, тим менше даних потрібно передавати по мережі. Крім того, ціна за збереження зображень на S3 зменшилася майже у 2 рази.

Продовжимо підключення додаткових оптимізацій. Проведемо наступний експеримент із ввімкненим кешуванням за допомогою сервісу CloudFront. Він працює наступним чином: сервіс CloudFront отримує інформацію про регіон, з якого роблять запит. І далі він переадресує запит на найближчу локацію. Такі локації називаються Edge locations і їх може бути декілька в залежності від інформації, звідки будуть поступати запити. Наприклад, якщо в CloudFront налаштовано локацію у Європі і клієнт фізично там знаходиться, то саме ця локація буде обробляти запит. Таким чином час доставки контенту клієнтові значно зменшіться. Аналогічно на локаціях можна включити кешування. Страждає лише перший клієнт. Він буде робити запит на зображення і саме він отримає максимальний час доставки зображення. Наступні клієнти отримуватимуть дані з кеша. Тобто генерація зображення буде відбуватися лише один раз в певний

проміжок часу, в залежності від налаштувань часу життя контенту на локаціях. Діаграма архітектури рішення, описаного вище, на AWS зображена на рисунку 27.

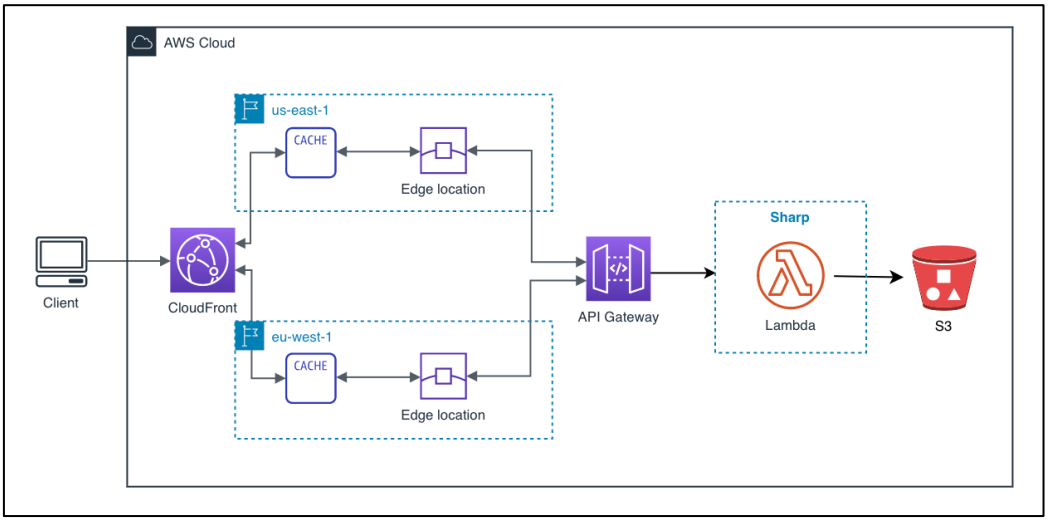


Рисунок 27 – Діаграма архітектури AWS з кешуванням зображень

У ході цього експерименту отримали значно кращі показники часу доставки зображення (див. табл. 28). Вони зменшилися майже на 70%, що є великим кроком вперед. Крім того, збільшилася на 40% і максимальна кількість запитів на сервер. Звісно, кешування не є дешевим, ціна зросла в приблизно в 10 раз, однак отримані результати сповна це компенсують.

Таблиця 28 – Результати експерименту з кешуванням

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	1.02 с	0.79 с	0.43 с
Час доставки зображення	0.32 с	0.23 с	0.11 с
Максимальна кількість запитів	2501 зап/хв	2649 зап/хв	2753 зап/хв
Кількість пам’яті	1754 МБ	1514 МБ	1371 МБ
Ціна рішення на AWS	85.10 USD	85.10 USD	85.10 USD
Ціна рішення S3	5.27 USD	4.62 USD	3.51 USD

Проведемо останній експеримент у цьому раунді, де ми сконцентруємося на оптимізації сховища. Як вже було зазначено вище, загальний розмір 5000 файлів,

збережених у трьох різних форматах на S3 становить з 8.4 ГБ. За допомогою компресії зображення ми зменшили цей показник у 2 рази, однак це не кінцевою межею. Якщо проаналізувати поведінку користувача, можна побачити, що кожний розмір зображення використовують з різною частотою. Наприклад, зображення 600 пікселів використовують найчастіше, оскільки вони знаходяться на головній сторінці інтернет магазину, 1000 пікселів дещо рідше, а 100 пікселів найменше. Відповідно до цього ми можемо виставити різний час життя об’єктів у сховищі. З плином часу вони будуть автоматично видалені. Діаграма архітектури рішення, описаного вище, на AWS зображена на рисунку 29.

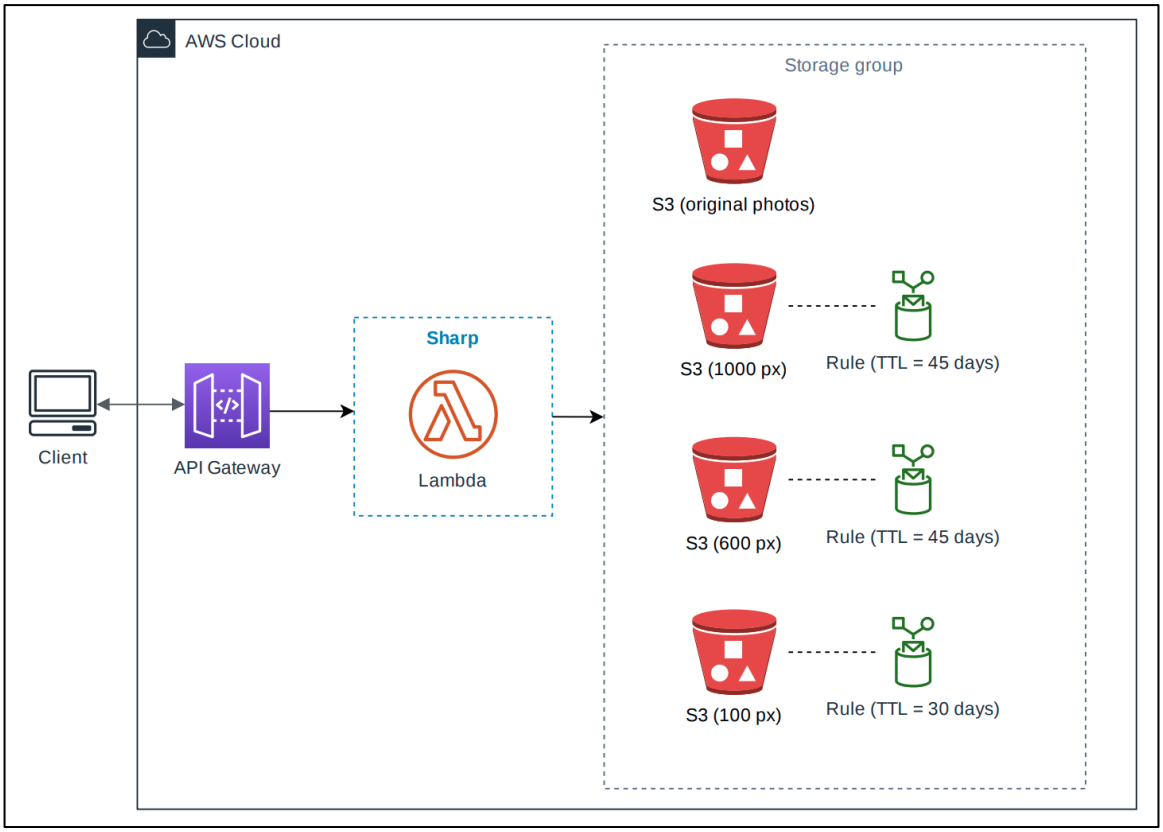


Рисунок 29 – Діаграма архітектури AWS з оптимізацією сховища

Цей експеримент залежить від часу, тому для швидкості його проведення було виставлено менші показники тривалості життя об’єктів. Ціна за S3 у цьому випадку буде розрахована як середнє значення за 3 місяця підтримки рішення. Відповідно до отриманих результатів, бачимо, що ми отримали зменшення вартості сховища і трохи збільшений час генерації зображення та пам’яті. Це пов’язано із

тим, що у випадку, коли об’єкт було видалено зі сховища, його потрібно заново створити. Результати експерименту занесено до таблиці 30.

Таблиця 30 − Результати експерименту з оптимізацією сховища

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	1.32 с	0.98 с	0.62 с
Час доставки зображення	1.41 с	1.26 с	0.85 с
Максимальна кількість запитів	1102 зап/хв	1243 зап/хв	1376 зап/хв
Кількість пам’яті	1737 МБ	1549 МБ	1293 МБ
Ціна рішення на AWS	8 USD	8 USD	8 USD
Ціна рішення S3	4.29 USD	3.34 USD	2.79 USD

У ході третього раунду експериментів можемо зробити висновок, що додаткові оптимізації мають місце бути. Незважаючи на те, що вони покращують роботу системи більш точково, на відміну від попередніх двох раундів, наведені вище оптимізації дійсно дають хороші результати. Їх можна використовувати як окремо, так і у комбінації один з одним.

На основі проведених експериментів та зроблених висновків щодо результатів кожного із них, запропонуємо рішення, яке будемо вважати максимально ефективним і готовим для подальшого використання. Це рішення включатиме:

− Lambda сервіс як основу для генерації зображення;

− Sharp як інструмент для зміни розміру зображення;

− компресія зображення за допомогою JPEGmini;

− кешування за допомогою CloudFront;

− оптимізація сховища за допомогою конфігурації часу життя об’єктів у ньому.

Тому проведемо останній експеримент у ході кваліфікаційної роботи, де реалізуємо це рішення і отримаємо значення критеріїв його ефективності. Діаграма архітектури рішення наведена у додатку А.

Відповідно до результатів, наведених у таблиці 31, бачимо, що ми досягли доволі гарних показників по усім критеріям. Наша система ефективно працює, дані віддаються клієнтам швидко, а ціна підтримки цього рішення всього приблизно 86 доларів на місяць, що доволі мало, вважаючи її спроможності.

Таблиця 31 − Результати фінального експерименту

Критерій ефективності	**1000 пікселів**	**600 пікселів**	**100 пікселів**
Час генерації зображення	1.04 с	0.83 с	0.49 с
Час доставки зображення	0.32 с	0.26 с	0.14 с
Максимальна кількість запитів	2538 зап/хв	2695 зап/хв	2801 зап/хв
Кількість пам’яті	1623 МБ	1413 МБ	1246 МБ
Ціна рішення на AWS	85.10 USD	85.10 USD	85.10 USD
Ціна рішення S3	3.12 USD	2.68 USD	1.99 USD

У результаті проведених дослідів, можемо сформувати ряд стверджень, які були експериментально доведені:

− Lambda сервіс працює дещо повільніше, ніж EC2, однак ціна його використання значно приємніша;

− Sharp є більш ефективним, порівняно з ImageMagick, він потребує менше пам’яті і з більшою швидкістю обробляє запити;

− компресія зображення обов’язково повинна бути реалізована, вона дозволяє значно зберегти місце у сховищі;

− для максимально швидкої доставки контенту необхідно реалізовувати шар кешування на сервісі CloudFront, або на будь-якому іншому;

− оптимізація сховища за допомогою конфігурації часу життя об’єктів дає невелике покращення по ціні використання сховища, однак вона є не такою значною.

6 АНАЛІЗ РЕЗУЛЬТАТІВ ЕКСПЕРИМЕНТІВ

Провівши ряд експериментів для дослідження методів динамічної обробки зображень, проаналізуємо кожен із критеріїв оцінки їх ефективності: час генерації, час доставки зображень, кількість пам'яті, що використовується, ціну рішення на AWS та ціну сховища S3. Детальні результати по кожному із критеріїв у ході експериментів наведено у додатку Б.

Час генерації зображення – це час, який необхідний для того, аби зменшити розмір зображення до заданого, у нашому випадку до 1000, 600 та 100 пікселів. На графіку (див. рис. 32) зображено, як цей час зменшувався чи збільшувався у ході проведення експериментів.

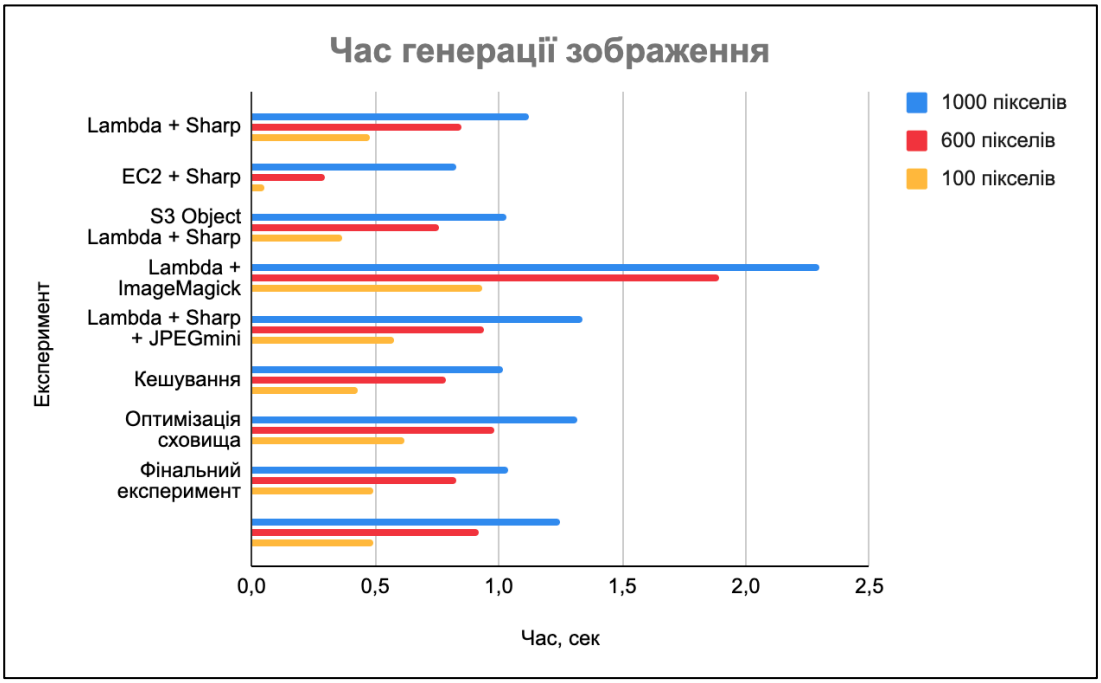


Рисунок 32 – Порівняння часу генерації зображення

На графіку бачимо, що протягом усіх експериментів час генерації зображення тримався майже на одному рівні: у середньому для 1000 пікселів – у проміжку від 1 до 1.5 секунди, 600 пікселів – від 0.5 до 1 секунди, 100 пікселів – до 0.5 секунди. Виключеннями стали результати двох експериментів: Lambda + ImageMagick та EC2 + Sharp. У першому випадку час збільшився майже у 2 рази,

порівняно із середніми значенням. У другому випадку навпаки − зменшився приблизно на 40%. Відповідно до цього, можемо зробити 2 висновки. По-перше, інструмент ImageMagick значно програє Sharp по ефективності. По-друге, на сервісі EC2 результати краще, ніж на Lambda, однак і ціна його використання відповідно в рази більша. Тому якщо час є більш пріоритетним критерієм, ніж ціна, то необхідно використовувати саме EC2.

Час доставки зображення − це час, від відправки сервером зображення до моменту його отримання клієнтом. Як вже зазначалося у попередніх розділах, він залежить від віддаленості клієнта до локації сервера, розміру зображення та можливості кешувати відповіді запитів. На рисунку 33 зображено графік порівняння часу доставки для кожного із експериментів.

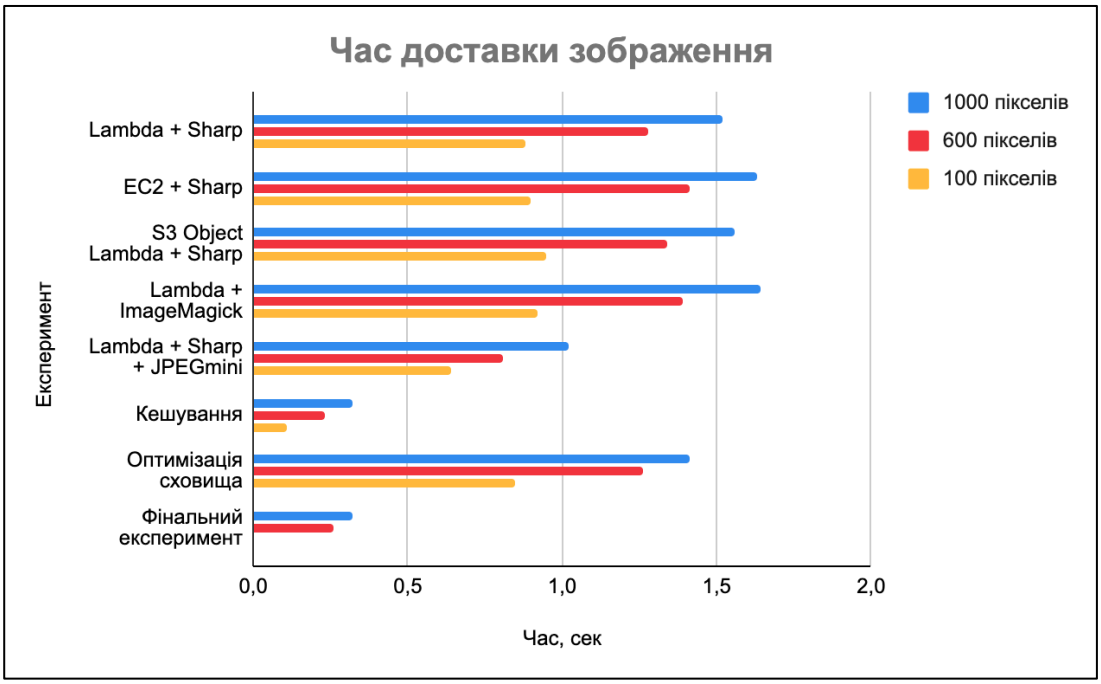


Рисунок 33 − Порівняння часу доставки зображення

Відповідно до даних на графіку, є два експерименти, в яких час доставки значно зменшився порівняно із іншими. Перший із них − ввімкнена компресія зображень. Оскільки вона дозволила зменшити розмір вихідних зображень приблизно у половину, то і розмір контенту, що передається по мережі зменшився у приблизно 2 рази відповідно. Це дозволило отримати незначне зменшення часу

доставки − приблизно на 0.4 секунди для кожного розміру зображення. Значний вплив на нього мало саме розміщення серверів у декількох локаціях і ввімкнення кешування. Наприклад, для зображення 1000 пікселів час зменшився з 1.52 секунд до 0.32, що є зменшенням майже на 80%. Кешування не є обов'язковою вимогою до високонавантажених систем, однак воно є бажаним. Більшість сучасних систем у даному випадку фокусуються на користувацькому досвіді та вважають за необхідним його мати.

Кількість пам'яті − це розмір пам'яті у МБ, який використовується у запиті. Вона складається із двох основних компонентів: пам'яті, що використовується для здійснення операції зменшення розміру зображення та пам'яті, яку займає файл вихідного і результуючого зображень. Графік порівняння кількості пам'яті, що використовується, для всіх експериментів наведено на рисунку 34.

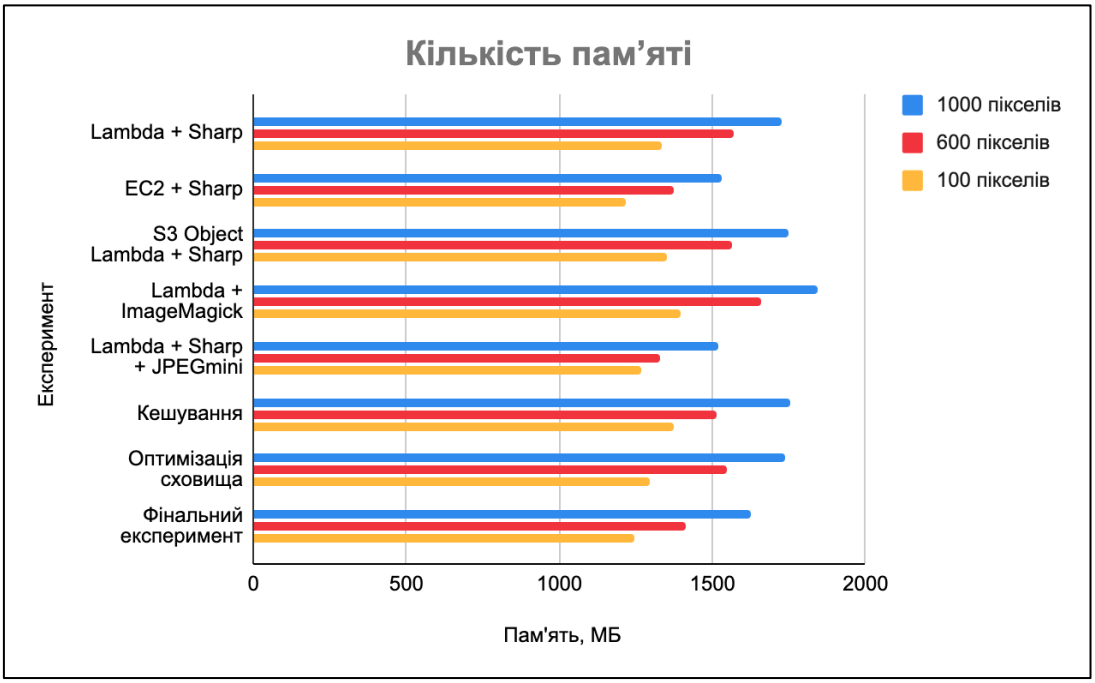


Рисунок 34 − Порівняння кількості пам'яті

Серед усіх проведених експериментів, є два, що показали мінімальні результати. Це експеримент, у якому було використано сервіс EC2 та компресія зображень. Щодо останнього, як вже було зазначено вище, на кількість пам'яті впливає розмір зображення. Оскільки компресія його зменшує, то відповідно і

менше пам’яті потрібно. Дещо цікавіший результат ми отримали у експерименті з EC2. Незважаючи на те, які б переваги не мала Lambda, в першу чергу вона спроектована як легковісний сервіс, який швидко запускається, швидко виконує свою задачу і швидко закінчує свою роботу. Тому існує можливість, що деякі операції з управлінням пам’яті є не до кінця оптимізованими. EC2 є сервісом, на якому можна розгорнути повноцінний сервер. Відповідно по потужності, навіть самий простий EC2 сервіс буде вигравати Lambda по деяким параметрам і можливостям.

Максимальна кількість запитів − це пропускна спроможність, тобто та кількість запитів при якому система може повноцінно працювати без значного просідання по її потужності. На рисунку 35 зображено графік зміни цього критерію для кожного із експериментів.

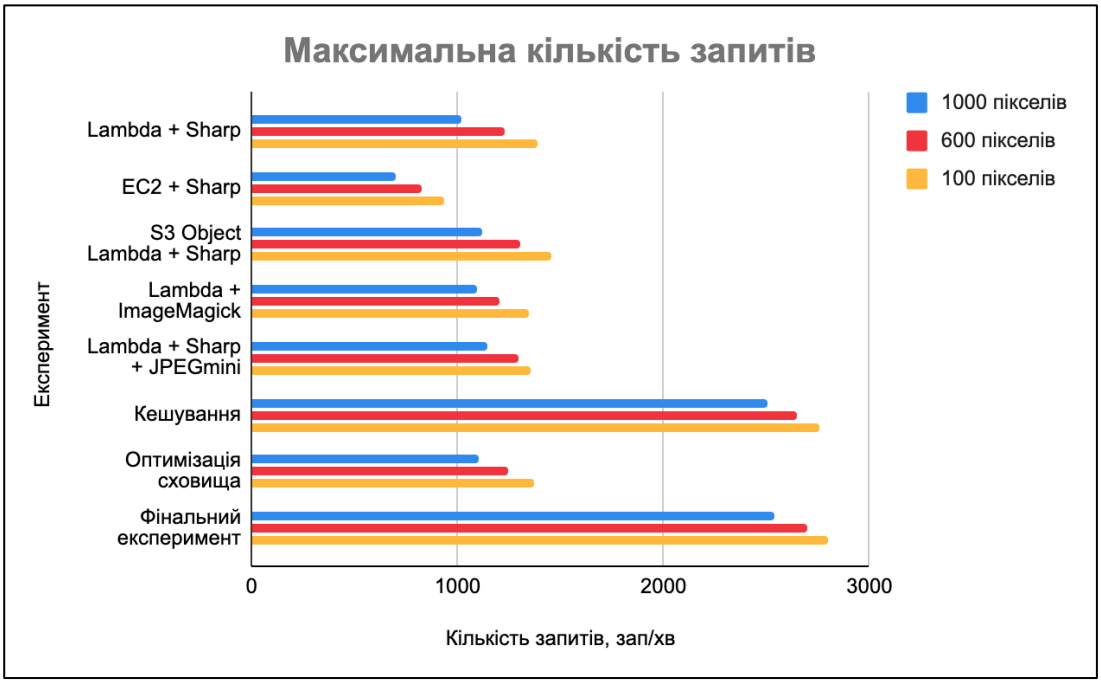


Рисунок 35 − Порівняння максимальної кількості запитів

Відповідно до результатів експериментів, максимальної кількості запитів вдалося досягти лише у одному випадку − при використанні CloudFront з декількома Edge локаціями на ввімкненим кешуванням. В той час як без даної оптимізації максимальна кількість запитів становила приблизно 1000 запитів на

хвилину, то с нею це число збільшилося майже у 1.5 рази і наш сервіс вдалося розіграти до 2500 запитів на секунду.

Ціна рішення на AWS − це витрати на підтримку системи без урахування сховища. Сюди входить підтримка наступних сервісів в залежності від експерименту: Lambda, EC2, API Gateway, CloudFront, Edge locations, CloudWatch. Графік порівняння ціни для усіх експериментів показано на рисунку 36.

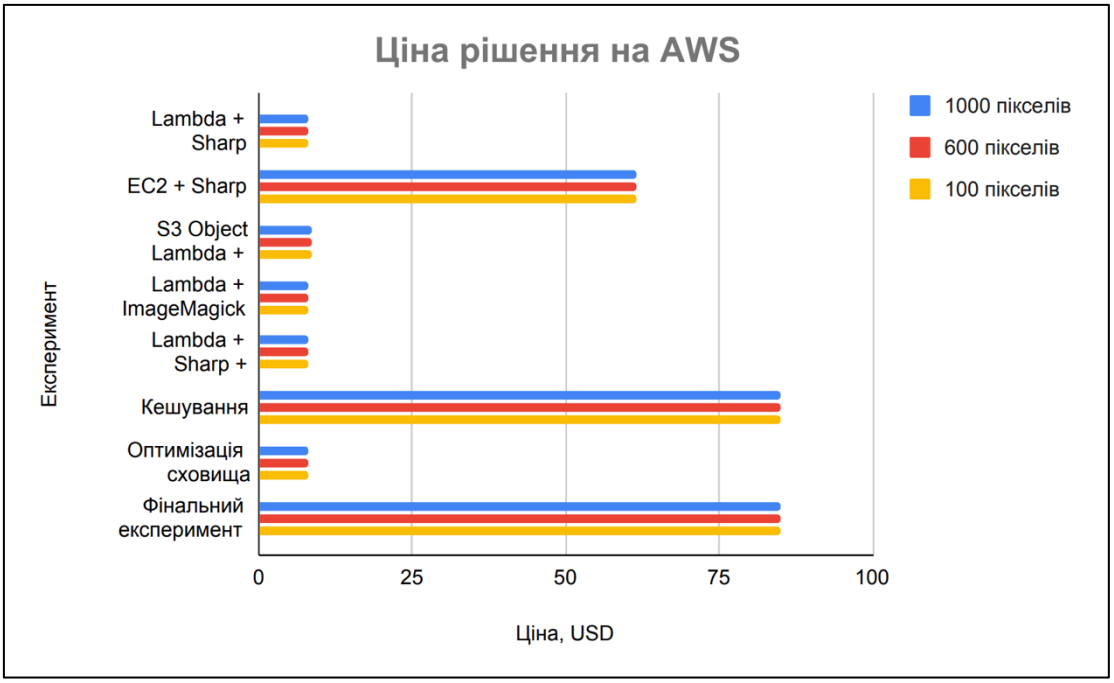


Рисунок 36 − Порівняння ціни рішення на AWS

Як бачимо, для більшості експериментів ціна залишається незмінною: у межах 8-9 доларів на місяць. Це відбувається тому, що вартість типової комбінації Lambda + API Gateway доволі низька, а саме вона і була використана у більшості експериментів. Вартість користування сервісом EC2 в рази більше − близько 60 доларів на місяць. Варто зазначити, що для експерименту ми взяли досить типовий вид EC2. Тому 60 доларів − це скоріше найбільший мінімум, при якому ми не побачимо значного просідання системи. Звісно, можна було би обрати більш потужні інстанси, однак і ціна була би відповідною. Щодо експерименту з CloudFront, він є одним із найдорожчих серед розповсюджених сервісів. Основою ціноутворення є те, що AWS дозволяє фізично розташувати сервери у декількох

локаціях. Кешування контенту також відбувається на їх серверах. Однак, його можна реалізувати і самому на тих сервісах, які є більш доступними.

Ціна рішення S3 − це витрати на підтримку сховища. Вона складається із наступних компонентів: ціна за збереження оригінальних зображень, це приблизно 40% суми. Наступні 25% становить збереження зображень після зміни їх розміру. Останні 35% − витрати на операції із об’єктами у сховищі. На графіку (див. рис. 37) зображено, як змінювалася ціна у ході проведення експериментів.

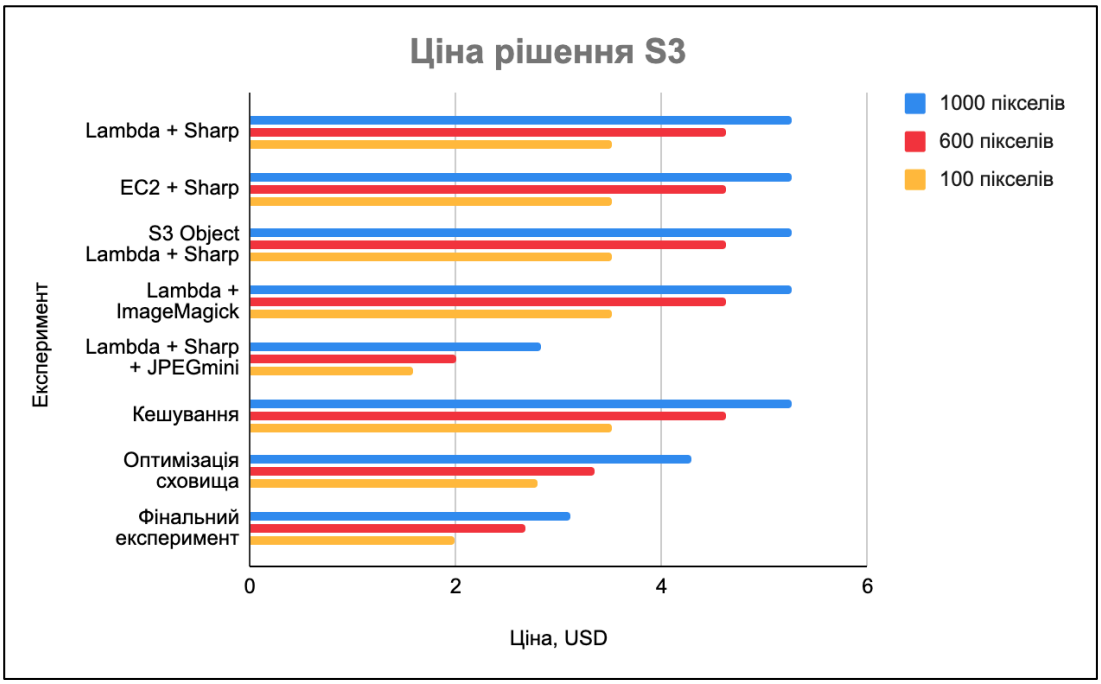


Рисунок 37 − Порівняння ціни сховища S3

Відповідно до даних на графіку, ціна за сховище є досить невеликим, у проміжку 2-5 доларів за місяць. Значне зменшення ціни відбулося у двох випадках: при використанні інструменту JPEGmini для компресії зображень та при оптимізації самого сховища, коли кожен із об’єктів мав час життя і автоматично видалявся при його закінченні.

Таким чином, графіки, подані у даному розділі, а також проведений аналіз результатів дозволяє сформувати ряд рекомендацій щодо методів динамічної обробки зображень та сфер їх використання.

7 ПОДАЛЬШІ РЕКОМЕНДАЦІЇ

У ході кваліфікаційної роботи було проведено ряд експериментів, які розглядають можливості різноманітних методів оптимізації динамічної обробки зображень. Деякі із них є оптимізаціями на глобальному рівні, тобто вирішують проблему в цілому і дають значне покращення по усім критеріям ефективності. Інші є більш точковими, тобто діють направлено, покращуючи певну область рішення, наприклад управління сховищем даних. У результаті проведених дослідів, можемо сформувати ряд стверджень, які можна використовувати як рекомендації для вирішення даного або схожого типу задач у майбутньому.

Аналізуючи останні тренди у області розробки програмного забезпечення, все більше проектів відходить від ідеї серверів та надають перевагу безсерверним рішенням. Lambda є прикладом такого рішення. В першу чергу вона була спроектована як невелика функція, виконання якої може займати не більше 15 хвилин. Вона легка і доволі швидка. Відповідно до результатів, отриманих у ході експериментів, ми побачили, що EC2 обробляє запити швидше, ніж Lambda, однак ця різниця становить лише пів секунди. Така поведінка є доволі очевидною, оскільки EC2 є повноцінним і дуже потужним сервером. Lambda − така собі демо версія цього серверу. Відповідно ціна користування EC2 набагато більше, ніж Lambda. Тому однозначної рекомендації, що краще − EC2 чи Lambda − немає. Кожен із цих сервісів підходить до свого типу задач. При виборі варто задати собі два основних питання: чи готові ми перейти повністю на безсерверне рішення та чи готові ми платити в рази більше, за невеликий виграш у часі при динамічній генерації зображень. На мою думку, для даного типу задач більше підходить Lambda, оскільки це рішення є певним балансом між продуктивністю і ціною підтримки.

Щодо інструменту для обробки зображень, однозначно варто надавати перевагу Sharp, а не ImageMagick. Останній є інструментом, який використовується у доволі великій кількості легасі проектів. Sharp працює на основі платформи libvips, яка вважається набагато продуктивнішою, ніж ImageMagick. Однак, варто

зазначити, що Sharp − це бібліотека лише для Node.js. Тому якщо використовується інша мова програмування, то все одно необхідно знайти інструмент, який би працював на основі libvips. Це дозволить виконувати запити на маніпуляції з зображеннями в рази швидше. Крім того, Sharp потребує менше пам’яті, що також вплине на ціну підтримки рішення, трохи зменшивши її.

Компресія − це простий і майже безкоштовний спосіб пришвидшити роботу із зображеннями та їх доставку до кінцевого користувача. Тому її реалізація є не тільки рекомендованою, а радше обов’язковою. У ході експериментів було доведено, що одна проста команда може зменшити розмір сховища даних у майже 2 рази без впливу на якість самого зображення. Існує безліч інструментів для компресії, тому його вибір лягає на плечі самої системи. Інструмент JPEGmini, який було протестовано у ході експериментів, доволі достойно показав себе і значно покращив фінальний результат.

Наступною обов’язковою оптимізацією є кешування запитів. Кешування є одним із типових способів прискорити роботу серверу та значно зменшити навантаження на нього. Воно доволі часто використовується у високонавантажених системах. У наш час важко уявити систему, яка би не використовувала кешування. Його можна реалізувати за допомогою багатьох сервісів. В даному випадку головним моментом є те, хто буде відповідати за менеджмент кешу. В проведених експериментах всі обов’язки було перекладено на AWS та на сервіс CloudFront. Він є одним із недешевих. Звісно, кешом можна управляти самостійно, тоді ціна такого рішення буде значно менша, однак зусиль на побудову гарного слою кешування буде покладено чимало. Тобто головною рекомендацію по цьому є те, що кеш обов’язково повинен бути. Вибір сервісу для кешування залишається за розробниками системи. На мою думку, CloudFront є одним із базових і тих, що найчастіше використовують.

Ще однією можливістю CloudFront у поєднанні з Edge локаціями є направлення запитів на найближчу локацію. У даному випадку потрібно орієнтуватися лише на географію основних користувачів системи. Перш за все, якщо користувачі лише с однієї локації, наприклад із західної Європи, то немає

сенсу платити більше. Вистачить серверу, який буде знаходитися у цій локації і все. Однак, якщо наприклад ваша система є аналогом Amazon із користувачами зі всього світу, то варто задуматися над Edge локаціями. Таким чином запити користувачів будуть направлені на найближчу локацію. Це значно підвищить пропускну здатність системи та зменшить час запиту. У результаті це покращить користувацький досвід.

Щодо оптимізації сховища за допомогою конфігурації часу життя об’єктів, однозначно рекомендації немає. Серед переваг, вона дає невелике покращення по ціні використання сховища при незначній кількості об’єктів. Ця оптимізація матиме значний сенс при збереженні сотень ГБ даних у сховищі. Тому у даному випадку необхідно дивитися на об’єми даних і робити висновок: якщо даних багато, то дана оптимізація однозначно дасть змогу зекономити. У противному випадку вона не нестиме цінності.

Останній проведений експеримент включав в собі усі оптимізації, які були розглянуті у ході кваліфікаційної роботи. Відповідно до отриманих результатів, можемо зробити висновок, що нам вдалося досягти ідеальної комбінації між продуктивністю і ціною. Рекомендації, подані вище, експериментально підтверджені. Отримане рішення можна використовувати як основу для подальшого проектування систем, основною метою яких є маніпуляції з великою кількістю зображень.

ВИСНОВКИ

У ході кваліфікаційної роботи було проведено дослідження задачі оптимізації зображень та різні варіанти їх генерації для подальшого використання у додатках. Вирішення цієї задачі може відбуватися на різних рівнях: архітектурному та програмному. На архітектурному рівні вирішення задачі оптимізації полягає у використанні рівних сервісів AWS, наприклад таких як EC2 або Lambda. На програмному рівні можна використовувати різні інструменти для генерації розмірів зображень: ImageMagick або Sharp. Крім того, до кожного із рішень можна додати ряд додаткових оптимізації, які значно покращать результати. Наприклад, можна робити компресію зображення без втрати його якості за допомогою інструменту JPEGmini. Це зменшує розмір вихідного зображення. Використання кешування за допомогою сервісу CloudFront значно зменшить час доставки зображення до користувача. А якщо правильно спроектувати структуру сховища даних та час життя об’єктів у ньому, то можна значно зменшити ціну за користування сервісом S3.

У ході проведених експериментів було практично доведено ряд тверджень і сформовано ряд рекомендацій для їх подальшого використання у системах заданого типу. Наприклад сервіс EC2 хоч і генерує зображення швидше, але ціна його використання майже у 4 рази більше, ніж Lambda. Інструмент Sharp працює швидше і споживає менше пам’яті порівняно з ImageMagick. Тому саме його доцільно використовувати для ресайзингу зображень. Додаткова компресія зображень показала, що можна досягти значного зменшення ціни використання сховища без втрати якості самих зображень. Кешування на рівні CloudFront та Edge locations значно знизило час доставки контенту до користувачів, оскільки кожному із них повертається контент із найближчої до нього локації.

Отримане у ходи експериментів рішення можна використовувати як основу для AWS інфраструктури систем, основною метою яких є динамічна обробка великої кількості зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Report: Page Weight 2017-2021. *HTTP archive* : веб-сайт. URL: https://httparchive.org/reports/page-weight?start=2017_04_15&end=latest&view=list (дата звернення: 25.01.2022).

2. Worldwide infrastructure as a service (IaaS) and platform as a service (PaaS) hyperscaler market share from 2020 to 2022, by vendor. *Statista* : веб-сайт. URL: https://www.statista.com/statistics/1202770/hyperscaler-iaas-paas-market-share (дата звернення: 28.01.2022).

3. Think Fast. The Page Speed Report. Stats & Trends for Marketers. *Unbounce* : веб-сайт. URL: https://unbounce.com/page-speed-report/ (дата звернення: 25.01.2022).

4. Білоконенко В.М , Ревенчук І.А. Алгоритми сегментації зображень на базі побудови матриць збігів. Східно-Європейський журнал передових технологій.-№2(62), том 2.-2013.-С.43-45.

5. Storer J. Data Compression: Methods and Theory : навч. посіб. – W.H. Freeman & Company, 1988. – 413 с.

6. Wittig M., Wittig A. Amazon Web Services in Action : навч. посіб. – Hakking, 2015. – 424 с.

7. Sus, B., Revenchuk, I., Tmienova, N., Bauzha, O., Chaikivskyi, T. Software System for Virtual Laboratory Works.- 2020 IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2020 - Proceedings, 2020, 1, Pp. 396–399, 9322046.

8. Still M. The Definitive Guide to ImageMagick : навч. посіб. – Apress, 2005. – 359 с.

9. Sharp image processing Performance Results. *Sharp* : веб-сайт. URL: https://sharp.pixelplumbing.com/performance (дата звернення: 30.02.2022).

10. Profiling functions with AWS Lambda Power Tuning. *AWS* : веб-сайт. URL: https://docs.aws.amazon.com/lambda/latest/operatorguide/profile-functions.html (дата звернення: 12.03.2022).

11. Sbarski P. Serverless Architectures on AWS : навч. посіб. – Manning, 2017. – 376 с.

12. Sus, B., Tmienova, N., Revenchuk, I., Bauzha, O., Stirenko, S. Gamification approach to the creation of virtual laboratory works and educational courses.- CEUR Workshop Proceedings, 2020, 2711, P. 68–78.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

4. Білоконенко В.М , Ревенчук І.А. Алгоритми сегментації зображень на базі побудови матриць збігів. Східно-Європейський журнал передових технологій.-№2(62), том 2.-2013.-С.43-45.

7. Sus, B., Revenchuk, I., Tmienova, N., Bauzha, O., Chaikivskyi, T. Software System for Virtual Laboratory Works.- 2020 IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2020 - Proceedings, 2020, 1, Pp. 396–399, 9322046.

12. Sus, B., Tmienova, N., Revenchuk, I., Bauzha, O., Stirenko, S. Gamification approach to the creation of virtual laboratory works and educational courses.- CEUR Workshop Proceedings, 2020, 2711, P. 68–78.