Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Інформаційних управляючих систем_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

__Дослідження методів і технології підвищення ефективності інформаційних_
_____інформаційних Cloud систем з Big Data сегментами_____.
(тема)

Виконав:
студент __2__ курсу, групи ___ІУСТм-18-1___
_____Шилін О.С._____
(прізвище, ініціали)

Спеціальність __122 Комп'ютерні науки_____
_____
(код і повна назва спеціальності)

Тип програми __освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма____Інформаційні управляючі системи та технології_____
(повна назва освітньої програми)

Керівник _____проф. Саєнко В.І._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ІУС _____ _____Петров К.Е._____
(підпис) (прізвище, ініціали)

2019р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерних наук_____
(повна назва)

Кафедра____Інформаційних управляючих систем_____
(повна назва)

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _____122 Комп'ютерні науки_____
(код і повна назва)

Тип програми __освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма__Інформаційні управляючі системи та технології_____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові_____Шиліну Олександру Сергійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи ___Дослідження методів і технології підвищення ефективності __інформаційних Cloud систем з Big Data сегментами_____
_____
затверджена наказом по університету від _31_ __жовтня___2019 р. № _____
2. Термін подання студентом роботи до екзаменаційної комісії _17_ __грудня__ 2019 р.
3. Вихідні дані до роботи _Науково-технічні публікації та інтернет-джерела за тематикою атестаційної роботи_____
_____
_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі _Технологія хмарного_____ шифрування та дешифрування даних без використання Hadoop кластеру,____ технології токенізації табличних даних без використання Hadoop кластеру,__ метод шифрування та дешифрування за допомогою хмарних сервісів без____ Hadoop кластеру, метод токенізації табличних даних з використанням _____ хмарних сервісів без Hadoop кластеру, критерії ефективності_____
_____
_____
_____
_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка

Дата видачі завдання _____ _____20__ р.

Студент _____
(підпис)

Керівник роботи _____ _____проф. Саєнко В.І._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до атестаційної роботи містить: 90 сторінок, 25 рисунків, 6 таблиць, 17 джерел.

АРХІТЕКТУРА, BIGDATA, ВЕЛИКІ ДАНІ, КЛАСТЕР, MAPREDUCE, APACHE HADOOP, APACHE SPARK, GOOGLE CLOUD PLATFORM, BIGQUERY, KMS, OPENPGP, ШИФРУВАННЯ, ТОКЕНІЗАЦІЯ, ОБРОБКА ДАНИХ

Метою даної роботи є оптимізація існуючої Big Data інфраструктури та підвищення швидкості обробки даних. Головний засіб досягнення цих цілей – це дослідження нових методів та технологій оборобки великих даних з використанням хмарних сервісів.

В рамках даної роботи було впроваджено використання нових технологій з міграцією до хмарних сервісів, проведено дослідження методів шифрування та дешифрування великих файлів та токенізації і детокенізації табличних великих даних за допомогою хмарних сервісів, була доведена актуальність дослідження, проаналізовані поточні методи обробки даних, обґрунтована мета розробки нового методу, яка розглядається в цієї роботі та сформовані критерії ефективності, поставлена задача дослідження.

На основі цих досліджень підготовлено грунт до повномасштабного впровадження нових технологій до існючої Big Data інфраструктури, розроблено два нових методи шифрування та дешифрування великих файлів та токенізації і детокенізації табличних великих даних за допомогою хмарних сервісів.

В роботі було проведено повномасштабне тестування нових технологій та методів з використаннями різних підходів та на різних об’ємах даних. На базі цих тестувань була доведена ефективність нових технологій та методів шифрування і дешифрування та токенізації і детокенізації великих даних у порівнянні зі старими.

ABSTRACT

Explanatory note to the certification work contains 90 pages, 25 pictures, 6 tables, 17 sources.

ARCHITECTURE, BIGDATA, CLUSTER, MAPREDUCE, APACHE HADOOP, APACHE SPARK, GOOGLE CLOUD PLATFORM, BIGQUERY, KMS, OPENPGP, ENCRYPTION, TOKENIZATION, DATA PROCESSING

The purpose of this work is to optimize the existing Big Data infrastructure and improve data processing speed. The main ways of achieving these goals is to explore new methods and technologies for processing big data using cloud services.

The amount of job that was done within that work includes the use of new technologies with migration to cloud services. The researches of new methods of encryption and decryption of big files and tokenization and detonation of big data tables using cloud services were conducted. Also the relevance of the research was proved, the current methods of data processing were analyzed, the purpose of new method development was substantiated, which is considered in this work and formed the criteria of effectiveness, the task of research.

Based on these studies, there were done all the preparations for the full scale implementation of new technologies in the existing Big Data infrastructure, two new methods for encrypting and decrypting big files and tokenizing and detonating big data tables using cloud services were developed.

The full scale testing of new technologies and methods was carried out using different approaches and data volumes. On the basis of these tests, the effectiveness of new technologies and methods of encryption and decryption and tokenization and detonation of big data in comparison with the old ones was proved.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

AWS – Amazon Web Services

BD – Big Data

BQ – BigQuery

GCP – Google cloud platform

HDFS – Hadoop Distributed File System

IaaS – Infrastructure as a service

KMS – Google Key Management Service

PGP – Pretty Good Privacy

RDD – Resilient Distributed Dataset

UDF – User-defined functions

UML – Unified Modeling Language

СУБД – Система управління базами даних

ВСТУП

Нова парадігма розробки і підтримки інформаційних систем – це використання віртуальних ресурсів і хмарних просторів. Системи стають все більше і більше, це означає, що число користувачів системи може досягати декількох мільйонів. Кількість запитів щомиті також може бути кілька мільйонів. Дані генеруються величезними обсягами. Виникають потреби в величезних ресурсах і високопродуктивних серверах. Старі підходи для обробки даних стають неефективними. З'являються абсолютно нові технології роботи з даними, що дозволяють виробляти розподілені обчислення. Наприклад, до таких підходів відноситься Map-Reduce framework. Його відкрита реалізація Apache Hadoop – це величезний інфраструктурний компонент, який встановлюється на цілий кластер серверів і дозволяє автоматизувати розподілені обчислення. Завдяки використанню хмарних технологій ми можемо гнучко налаштовувати конфігурацію кластера, додавати нові або видаляти зайві елементи кластера вручну, а також уникнути витрат на покупку і підтримку фізичного кластера серверів. Останнім часом стали популярні комерційні реалізації Map-Reduce фреймворка. Однією з таких є Google Big Query, що є компонентом екосистеми Google Cloud Platform. Даний інструмент має ряд переваг над стандартною реалізацій, головним з яких є лінійне зростання часу обчислення з ростом кількості даних. На жаль, він також не позбавлений недоліків, головним з яких є відсутність деяких звичних інструментів для шифрування і токенізації даних, що не дозволяє повністю відмовитися від Hadoop кластера і змушує використовувати комбінований підхід до обчислення даних, особливо коли справа йде про шифрування та компресію даних.

Основне завдання дослідження полягає в пошуку нових методів та технологій, що дозволяють спростити архітектуру системи та значно зменшити витрати на підтримку існуючої.

Метою роботи є позбутися або ж значно зменшити навантаження на Hadoop Cluster. Для цього пропонується використання готових або створення аналогічних

інструментів для Google BigQuery для перенесення операцій шифрування і дешифрування на Google Cloud Platform.

Для досягнення поставленої мети розглядається вирішення наступних завдань:

- дослідити можливість шифрування файлів за допомогою засобів Google Cloud Platform;
- виконати аналіз поточного засобу токенізації даних та реалізувати аналог засобами Google Cloud Platform та BigQuery;
- розглянути особливості практичної реалізації запропонованих модифікацій
- виконати перформанс тестування нових компонентів та порівняти зі старими.

Звіт виконано відповідно до рекомендацій методичних вказівок з дослідницької практики [1].

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДОЛОГІЙ BIG DATA ОБЧИСЛЕНЬ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз історичного розвитку розподілених Big Data обчислень

Коли ми говоримо про великі дані, то маємо на увазі настільки великі об’ємі інформації, що не піддаються класичним засобам обчислення. Дуже часто мова йде про дані розміром від кількох гігабайт до кількох петабайт. Достатньо довгий час аналізувати такі дані не було ніякої можливості, тому їх просто складали у базу, а іноді просто видаляли. Ситуація почала змінюватись, коли у 2004 році співробітники Google Дуглас Каттінг та Майк Кафарелла сформували модель розподіленного обчислення великих об’ємів даних та опублікували статтю «MapReduce: Simplified Data Processing on Large Clusters» [3]. Це був переломний моменту світі інформаційних технологій. Одразу ж почалась реалізація даної методології, і нарешті у березні 2006 року побачила світ перша імплементація MapReduce моделі – Apache Hadoop версії 0.1.0.

MapReduce – це методоголія обробки паралельних завдань у великих наборах даних за допомогою великої кількості комп’ютерів, вузлів. Спільно їх називають кластером, якщо всі вузли в одній локальній мережі та використовують аналогічне обладнання, або сіткою, тобто якщо вузли поділяються між географічно та адміністративно розподіленими системами та використовують більш неоднорідне обладнання. Обробка може відбуватися на даних, що зберігаються або у файловій системі (неструктуровані, денормалізовані), або в базі даних (структуровані, номалізовані).

Методологія MapReduce складається з Map процедури, або методу, яка виконує фільтрацію та сортування, наприклад, сортування учнів за прізвищем у черги, одна черга на кожне ім’я, та метод Reduce, який виконує операцію підсумків, наприклад, підрахунок кількості учнів у кожній черзі, даючи частоту імен. Назви Map та Reduce походять від назв аналогічних функцій з функціонального програмування, хоча їх призначення в рамках MapReduce не таке, як у їхніх оригінальних формах.

Система MapReduce, також її називають «інфраструктура» або «фреймворк», організує обробку шляхом маршалінгу даних до розподілених серверів, паралельно виконуючи різні завдання, керуючи всіма комунікаціями та передачею даних між різними частинами системи та забезпечуючи надмірність і відмовостійкість.

Обчислення у MapReduce фреймворку завжди базуються на трьох операціях – Map, shuffle та reduce (Рис 1.1).

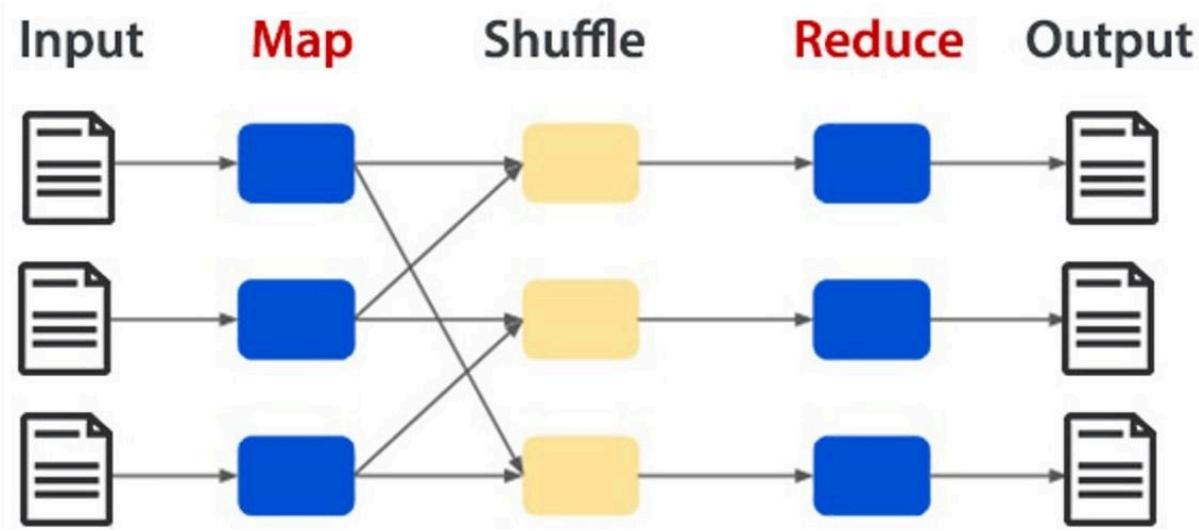

Рис. 1.1 – Схема обчилень за фреймворком MapReduce

Кожна з операцій має свою специфіку:

- Map – кожен робочий вузол застосовує функцію map до локальних даних і записує вихід у тимчасове сховище. Головний вузол забезпечує обробку лише однієї копії надлишкових вхідних даних;
- Shuffle – допоміжна операція, під час якої робочі вузли перерозподіляють дані на базі ключів, що виробляються функцією map, так що всі дані, що належать одному ключу, розташовані на одному робочому вузлі;
- Reduce – робочі вузли тепер обробляють кожну групу вихідних даних, за ключем, паралельно.

Також дуже часто виділяють окремо фази зчитування вхідних даних, тобто завантаженя даних з диску до оперативної пам’яті вузлів, та запис вихідної інформації, тобто результатів обчислень.

Наведемо наочний приклад типового обчилення на кластері за фреймворком MapReduce (Рис. 1.2).

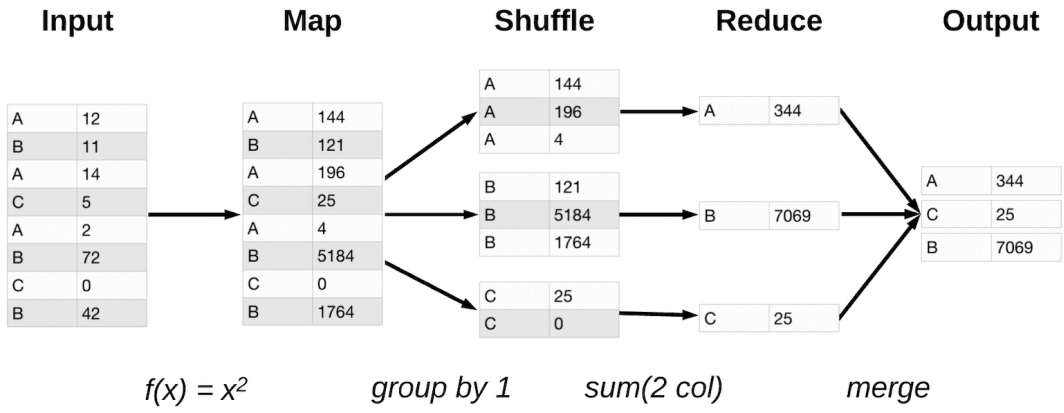

Рисунок 1.2 – Приклад обчислень за фреймворком MapReduce

Як видно з рисунку, обчислення відбувається за наступною схемою:

- фаза Input – необроблені денормалізовані дані зчитуються з диску до оперативної пам’яті;
- фаза Map – дані рівномірно розподіляються по робочих вузлах, над кожним записом виконується функція або декілька, у даному випадку функція зведення у квадрат;
- фаза Shuffle – групуємо дані по значенню першого поля, таким чином маємо, що записи з ключем A переміщуються до одного, дані з ключем B переміщуються до другого, а дані з ключем C – до третього робочого вузла;
- фаза Reduce – послідовно рахуємо суму другого поля по ключам на кожному з робочих вузлів;
- фаза Output – зберігаємо результати обчислень до файлової системи.

Apache Hadoop – перша типова реалізація MapReduce фреймворку. Hadoop встановлюється на кластер серверів, де має один керуючий вузел та багато робочих вузлів (Рис. 1.3). Ядро Apache Hadoop складається з частини зберігання, відомої як розподілена файлова система Hadoop (HDFS), і частини обробки, яка є моделлю програмування MapReduce [4]. Hadoop розбиває файли на великі блоки та розподіляє їх по вузлах кластеру. Потім він передає упакований код у вузли для

паралельної обробки даних. Цей підхід дає перевагу локальної обробки, адже вузли маніпулюють тільки тими даними, дотуп до яких їм дав керуючий вузел, та нічого не знають про інші.

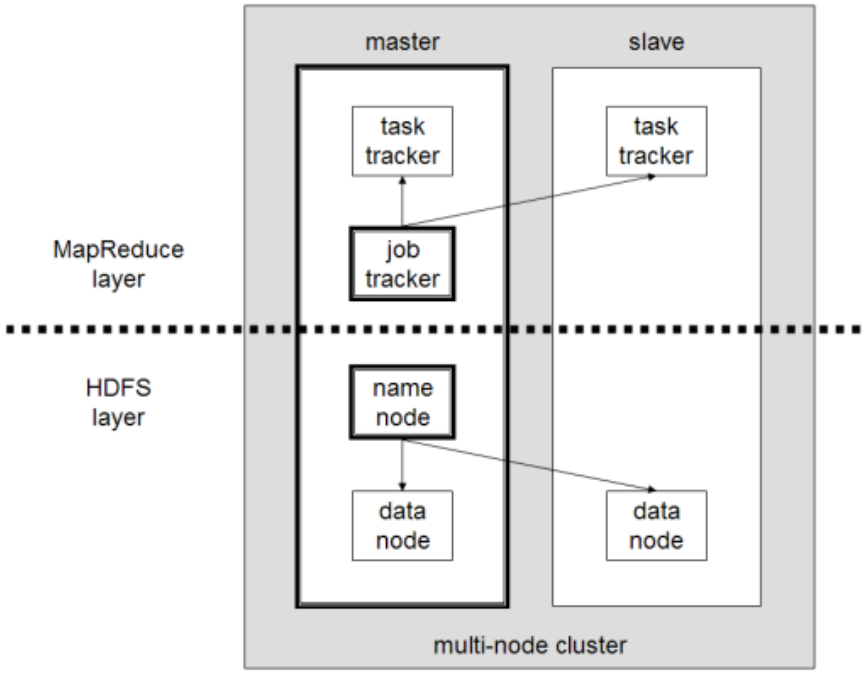

Рисунок 1.3 – Типова архітектура Hadoop кластеру

З часом на базі програмного забезпечення Hadoop стали формауватися фремворки з більш високим рівнем абстракції (Рис. 1.4). До них відносяться:

- Apache Pig;
- Apache Hive;
- Apache Spark.

Apache Pig – це платформа високого рівня для створення програм, які працюють на Apache Hadoop. Даний фреймворк використовує свою власну мову програмування, що називається Pig Latin. Pig Latin абстрагує від програмування на Java з Hadoop бібліотеками до написання скриптових сценаріїв, що робить MapReduce програмування більш високорівним, подібним до програмування на SQL для реляційних систем управління базами даних. Pig Latin можна розширити за допомогою визначених користувачем функцій (UDF), які користувач може

програмувати на Java, Python, JavaScript, Ruby або Groovy, а потім викликати безпосередньо зі скрипта.

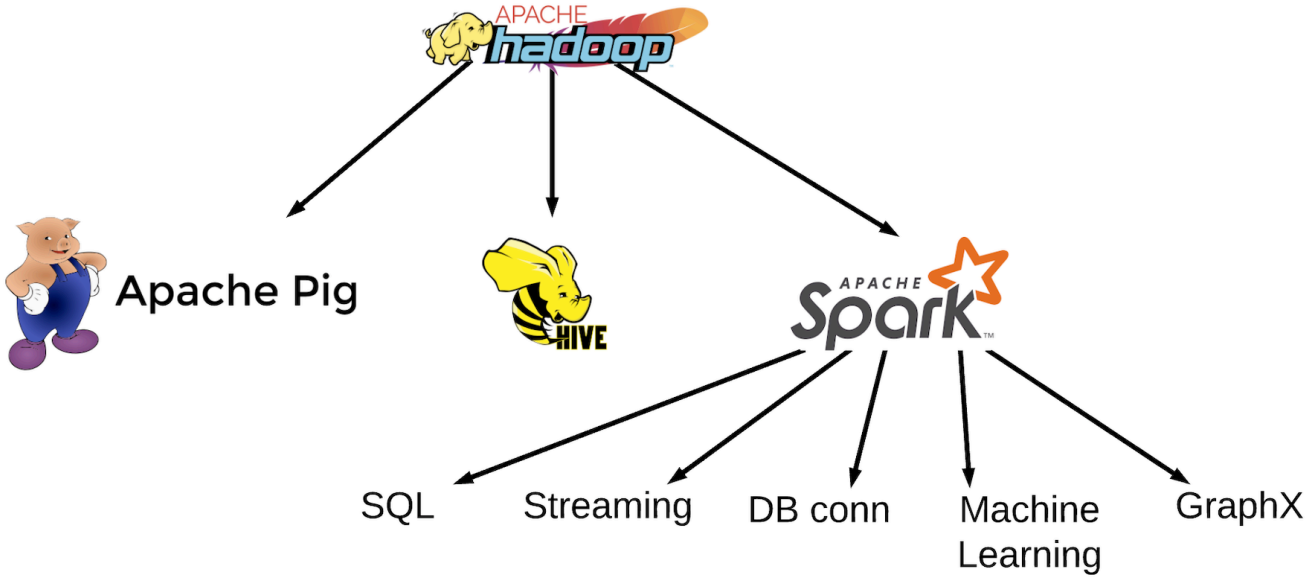

Рисунок 1.4 – Еволюція Apache Hadoop до високорівневих фреймворків

Через кілька років після релізу Apache Pig виходить принципово новий фреймворк – Apache Hive. Даний програмний продукт являє собою табличне сховище даних, подібне до схем та таблиць у реляційних базах даних, але, на відміну від яких, зберігає та обробляє дані на базі Hadoop. Hive надає SQL-подібний інтерфейс під назвою HiveQL для запиту, трансформації, обробки та зберігання даних, конвертуючи дані SQL запити у традиційні MapReduce програми більш низького рівня.

Ще через декілька років румуно-канадский інформатик Матей Захарія ствоює принципово новий фреймворк на базі MapReduce – Apache Spark. На відміну від попередників, Spark не використовує інтерфейси Hadoop, а має абсолютно нове ядро. Головна відмінність від класичного Hadoop обчислення – менша кількість циклів запису даних на дисковий простір та більша ступінь використання даних у оперативній пам’яті, що дає значний виграш у швидкості роботи [5]. Складається

з ядра і кількох розширень, таких як Spark SQL (дозволяє виконувати SQL-запити над даними), Spark Streaming (надбудова для обробки потокових даних), Spark MLlib (набір бібліотек машинного навчання), GraphX (призначена для розподіленої обробки графів). Може працювати як в середовищі кластера Hadoop, так і без компонентів ядра Hadoop. Окрім HDFS, підтримує ще декілька розподілених файлових систем, таких як Amazon S3 та NoSQL-СУБД Cassandra. Багато також змін відбулось і з точки зору написання програм – фреймворк надає інтерфейси до мов Java, Scala, R та Python, а маніпулювання даними відбувається через так звану RDD – абстракцію, подібну до класичних колекцій з функціональних мов програмування.

Але на цьому розвиток Big Data інтсрументарію не зупинився.

На сьогоднішний день останнім ступенем у еволюції Big Data засобів є Big Data as service (Рис. 1.5).

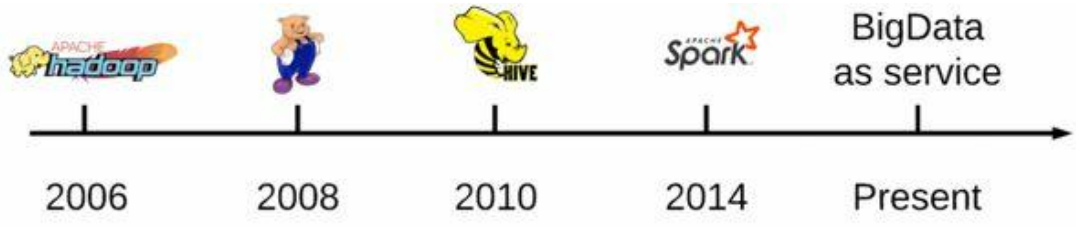

Рисунок 1.5 – Еволюція від Hadoop до хмарних сервісів

Суть цього підходу у тому, що уся Big Data інфраструктура та увесь інструментарій надає стороння компанія у якості послуги [6]. Як правило сервіси мають надзвичайно високу швидкість обчисленя. Це відбувається завдяки кільком факторам:

- постачальник сервісів може дозволити собі підтримувати надзвичайно великі кластери за рахунок великої кількості клієнтів з різних часових поясів;
- за рахунок величезного кластеру можна дозволити створювати динамічні віртуальні кластери, що будуть зменшуватися або збільшуватися у залежності від навантаження у конкретний час;

- використання пропрієтарних та патентованих алгоритмів компресії даних зменшує час передачі даних між вузлами.

1.2 Огляд існуючих постачальників Big Data сервісів

Google Cloud Platform – набір хмарних служб, що надається компанією Google, які виконуються на тій же самій інфраструктурі, яку Google використовує для своїх продуктів, призначених для кінцевих споживачів, таких як Google Search і YouTube. Крім інструментів для управління, також надається ряд модульних хмарних служб, таких як хмарні обчислення, зберігання даних, аналіз даних і машинне навчання. Для реєстрації потрібно мати банківську карту або банківський рахунок.

Google Cloud Platform надає такі послуги, як інфраструктура як послуга, платформа як послуга, і бессерверной обчислення.

У квітні 2008 року Google анонсувала App Engine – платформу для розробки та хостингу веб-додатків в дата-центрах Google. Це був перший хмарний сервіс, представлений компанією. Для громадськості сервіс став доступний в грудні 2011 року. З моменту анонса App Engine, Google встигла додати численні хмарні служби до своєї платформі.

Google Cloud Platform є частиною Google Cloud, який також включає G-Suite, корпоративні версії Android і Chrome OS, а також API для машинного навчання і Google Maps.

У січні 2019 року Google запустила чотири нові програми сертифікації для хмарних розробників та інженерів: Professional Cloud Developer, Professional Cloud Network Engineer (beta), Professional Cloud Security Engineer (beta), а також G Suite. Пройти навчання можна на платформі Coursera і у інших партнерів компанії.

Google Cloud Platform надає наступні сервіси [10]:

- App Engine – платформа як послуга для хостингу додатків;

- BigQuery – інфраструктура як послуга, що масштабується аналітика для баз даних;
- BigTable – інфраструктура як послуга, що масштабується NoSQL база даних;
- Cloud AutoML – набір продуктів для машинного навчання, які дозволяє розробникам з обмеженим досвідом роботи в області машинного навчання використовувати технології навчання і створення нейронних мереж;
- Cloud Datastore – документоорієнтованих хмарна база даних;
- Cloud Pub/Sub – послуга для публікації і підписки на потоки даних і повідомлення. Додатки можуть обмінюватися даними через публікацію/ підписку, без прямого обміну повідомленнями;
- Compute Engine – інфраструктура як послуга, надає віртуальні машини;
- Kubernetes Engine – система автоматичного розгортання, масштабування і управління додатків в контейнерах для Kubernetes;
- Google Genomics – аналіз геномів в хмарі;
- Google Video Intelligence;
- Cloud Vision;
- Cloud Storage – інфраструктура як послуга, надає онлайн REST-доступ до файлів і змістом сховищ даних.

Amazon Web Services – це комерційний публічний набір хмарних сервісів, який підтримується і розвивається компанією Amazon з 2006 року. Надає передплатникам послуги як по інфраструктурної моделі (віртуальні сервери, ресурси зберігання), так і платформного рівня (хмарні бази даних, хмарне сполучна програмне забезпечення, хмарні бессерверной обчислення, засоби розробки).

AWS у значній мірі (поряд з Google Cloud Platform) вплинуло на формування концепції хмарних обчислень в цілому, і визначило основні напрямки розвитку публічної моделі розгортання. Тривалий час було найбільшим в світі по виручці публічним хмарою, у другій половині 2010-х років поступившись за цим показником Azure від Microsoft, при цьому зберігаючи домінування в сегментах

інфраструктурних і платформних послуг. Станом на 2017 рік річна виручка від послуг AWS перевищила $ 20 млрд, що склало близько 11,5% доходів Amazon [7].

Головна інфраструктурна послуга – служба оренди віртуальних серверів EC2. Передплатникам надаються віртуальні машини, що працюють на гіпервізора Xen.

Xen – це кросплатформений гіпервізор, розроблений в комп'ютерній лабораторії Кембриджського університету і розповсюджуваний на умовах ліцензії GPL [8]. Основні особливості: підтримка режиму паравіртуалізації крім апаратної віртуалізації, мінімальність коду самого гипервизора за рахунок виносу максимальної кількості компонентів за межі гипервизора.

AWS надає вибір з різних по обчислювальної потужності машин, а також машин з доступом до спеціалізованого устаткування, наприклад відеокартам для GPGPU, програмованим вентильним матрицями, які часто використовують в машинному навчанні. EC2 тісно інтегрована з іншими інфраструктрнимі послугами хмари, перш за все – Elastic File System, що забезпечує приєднувану до віртуальних машин файлову систему, Elastic Block Store, які надають приєднуються до віртуальних машин томи як блокові пристрої, і S3, що забезпечує хмарне файлове сховище великого обсягу.

Серед платформених хмарних сервісів поширена оренда хмарних СУБД різноманітних категорій. Серед найбільш поширених NoSQL-систем:

- Amazon SimpleDB;
- DynamoDB;
- СУБД ElastiCache;
- графова СУБД Neptune.

Серед найбільш поширених SQL-систем [9]:

- MySQL;
- Oracle Database;
- Microsoft SQL Server;
- PostgreSQL;
- Amazon Aurora, сумісна з MySQL і PostgreSQL.

Служба Amazon Athena дозволяє проводити аналіз даних в Amazon S3, використовуючи стандартний SQL, притому для її роботи не потрібно виділених

обчислювальних потужностей, а передплатники оплачують тільки лічені в рамках виконаних запитів дані. За своєю суттю є повним аналогом Google BigQuery, що буде активно використовуватись у дослідженнях в рамках цієї атестаційної роботи.

Служба Elastic MapReduce дозволяє передплатникам створювати Hadoop-кластери, оснащені відповідною екосистемою продуктів класу Big Data, що включає в себе, але не обмежується:

- Apache Spark;
- Apache Hive;
- Apache Pig;
- HBase;
- Apache Presto.

1.3 Обгрунтування мети розробки нових методів та використання нових технологій

До переваг кластерного підходу у побудові Big Data інфраструкутри відносяться:

- потенційно більш низька вартість на аренду кластерів;
- високий потенціал для конфігурації серверів;
- інформація на серверах знаходиться повністю у власності споживача Big Data інформації.

До недоліків кластерного підходу у побудові Big Data інфраструкутри відносяться:

- більш низька швидкість обчислення у пікових навантаженнях через неможливість оперативного розширення кластеру;
- потребує багато людей для підтримки усієї інфраструктури;
- продуктивність сильно залежить від правильних архітектурних рішень та правильно написанного коду (часто, але не завжди).

До переваг хмарного підходу у побудові Big Data інфраструкутри відносяться:

- висока швидкість обчислення;
- значно простіша підтримка інфраструктури;
- масштабування під об’єм обчислювальних даних;
- підготовлені архітектурні рішення.

До недоліків хмарного підходу у побудові Big Data інфраструкутри відносяться:

- жорстка залежність на постачальника послуг;
- більш висока ціна.

Сучасний вектор розвитку Big Data систем передбачає перехід до хмарних обчислень [2]. Головний недолік – більш висока ціна – перекривається в довгостроковій перспективі тим, що бізнес підрозділ швидше отримує дані. Більш свіжі дані дозволяють значно ефективніше корегувати бізнес стратегію. Саме тому у роботі буде виконана міграція технологій, що виконуються на Hadoop кластері до хмарних сервісів, та розроблені методи щодо використання нових технологій на хмарних сервісах, що у перспективі дозволить повністю відмовитися від використання Hadoop кластеру та підвищити швидкість генерації даних.

1.4 Формування задачі дослідження

Головна задача дослідження – розробка нових методів та впровадження нових технологій, що дозволять відмовитися від використання неактуального на сьогоднішній день Hadoop кластеру, та дадуть змогу користуватися тільки хмарними сервісами без втрати необхідного функціоналу (Рис. 1.8). Для досягнення цих результатів потребується виконати дослідницьку роботу з можливості міграції поточного функціоналу, що використовує для своєї роботи Hadoop кластер, до хмарних сервісів.

В рамках роботи планується розробити да використати на практтиці наступні методи [2]:

- метод хмарного шифрування та дешифрування файлів великих об’ємів з використанням пари асиметричних ключів;
- метод хмарної токенізації та детокенізації табличних великих даних з використанням симетричного ключа.

Для реалізації тих методів потребуються впровадження наступних технологій:

- інструменти хмарного SSL шифрування та дешифрування з використанням хмарного сховища ключів Google KMS;
- інструменти хмарної токенізації та детокенізації табличних даних на базі Google BigQuery.

Відмова від використання неактуального на сьогоднішній день Hadoop кластеру має наступні наслідки:

- підвищена швисткість обчислення за рахунок використання Google BigQuery;
- підтримка суто хмарних сервісів потребує значно меншу кількість людських ресурсів;
- підтримка суто хмарних сервісів потребує значно меншу кількість матеріальних витрат, адже оренда декількох потужних віртуальних кластерів є значною статтєю витрат.

2 ВПРОВАДЖЕННЯ НОВИХ ТЕХНОЛОГІЙ ТА РОЗРОБКА НОВИХ МЕТОДІВ ХМАРНОЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ БЕЗ ВИКОРИСТАННЯ HADOOP КЛАСТЕРУ

У результаті досліджень існуючої інфраструктури для аналізу великих даних, які були проведені згідно з описаними шляхами організації та проведення досліджень у рамках поставленої наукової задачі, були отримані дані, алгоритми та опис існуючих методів та технологій. На їх основі були розроблені нові методи обробки даних з використанням шифрування без Hadoop кластеру та токенізації великих табличних даних без Hadoop кластеру, а також впроваджені нові технології, які будуть проаналізовані та детально описані в цьому розділі наукової атестаційної роботи.

2.1 Аналіз існуючої інфраструктури

Існуюча архітектура побудована на базі Google Cloud Platform. GCP має велику кількість сервісів (Рис. 2.1). До головних інструментів відносяться:

- BigQuery;
- Cloud Storage;
- Key Management Service;
- Compute Engine.

BigQuery – це хмарний веб-сервіс для інтерактивного широкомасштабного аналізу великих наборів даних, розташованих в Google Storage. Швидкість – це основна перевага BigQuery, але не єдина. BigQuery – хмарний сервіс, тому при його використанні не знадобиться орендувати сервер і оплачувати підтримку. Вартість BigQuery значно нижче вартості оренди самого примітивного сервера: навіть якщо ви дуже постараєтеся і будете щодня записувати в цю базу даних

мільйони рядків, все одно навряд чи зможете перевешити затрати на оренду та адміністрування повноцінного кластеру.

Google Storage – це веб-служба хостингу файлів для зберігання і доступу до файлів через REST в інфраструктурі GCP. Служба поєднує в собі продуктивність і масштабованість хмари Google з поліпшеними можливостями безпеки і спільного використання. Це інфраструктура як послуга (IaaS), подібна сервісу онлайн-зберігання Amazon S3.

Рисунок 2.1 – Сервіси Google Cloud Platform

Google KMS – це хмарний сервіс управління ключами, який дозволяє керувати криптографічними ключами для хмарних служб так само, як і в локальних серверах. Має можливість генерувати, використовувати, обертати та знищувати криптографічні ключі AES256, RSA-2048, RSA-3072, RSA-4096.

Google Compute Engine – це IaaS компонент Google Cloud Platform. Google Compute Engine дає змогу користувачам хмарних сервісів запускати віртуальні машини. Віртуальні машини можуть бути створені як з нуля, так і з будь-якого образу. Підртимує створення хмарних кластерів.

Поточна Big Data інфраструктура знаходиться у перехідному періоді від кластерного підходу до хмарного підходу, що почався нещодавно. Саме тому у побудові архітектури використовується комбінований підхід. Діаграма розгортання поточної архітектури подана на Рис. 2.2.

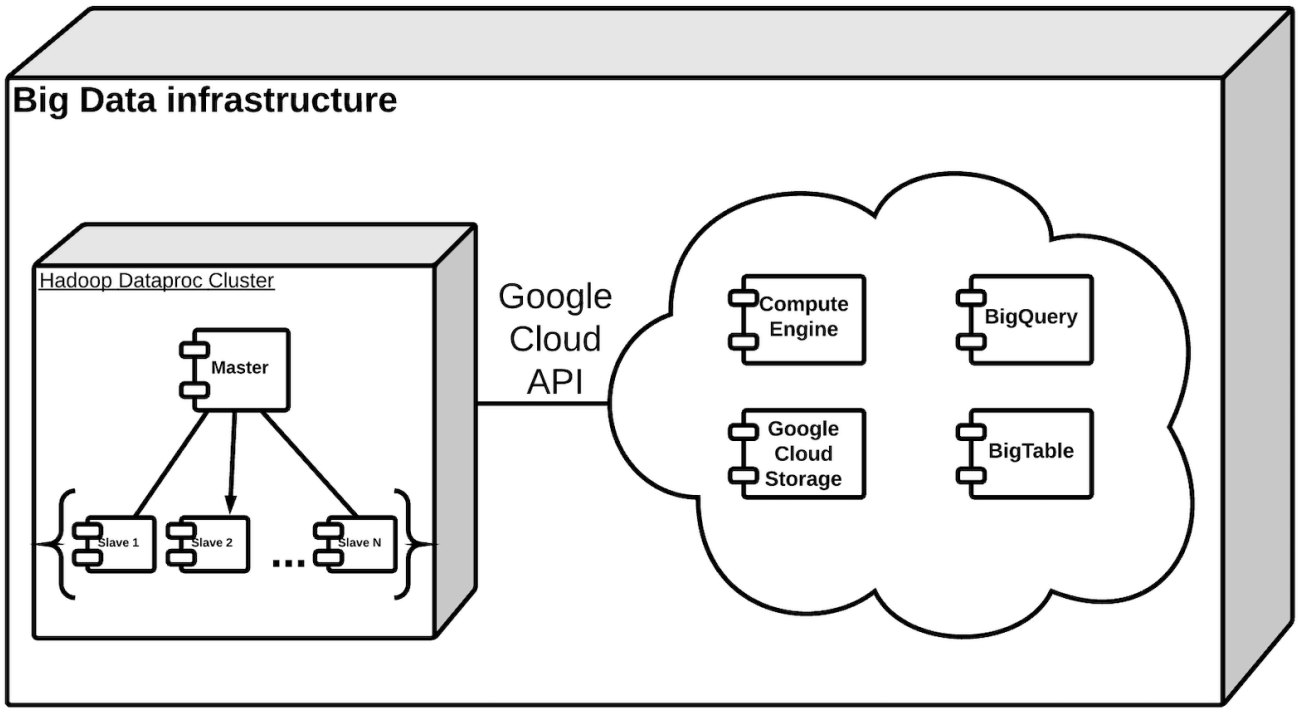

Рисунок 2.2 – Діаграма розгортання з поточною архітектурою

Як видно з рисунку, поточна архітекутра складається з наступних компонентів:

- Hadoop кластер (Hadoop Dataprocessing Cluster на рисунку)
- Хмарні компоненти на Google Cloud Platform

До недоліків такого архітектурного підходу можна віднести:

- Значно збільшений час комунікації між Hadoop кластером та хмарними сервісами у порівнянні з використанням суто Hadoop кластеру або суто хмарних сервісів, адже на даному етапі відбувається передача даних за інтернет каналами, що завжди є слабким місцем;
- підтримка Hadoop кластеру потребує значну кількість людських та матеріальних ресурсів;

- компанія, що споживає результати Big Data обчислень та володіє поточною Big Data інфраструктурою, вимушена нести подвійні витрати, адже їй потрібно орендувати віртуальні машини на Google Compute Engine та додатково сплачувати роботу хмарних Big Data сервісів.

Поточне архітектурне рішення є вимушеним, адже раніше ми мали тільки Hadoop кластер для виконання на ньому Hive, Pig та Spark програм. Зараз ми знаходимося у пероді переходу з кластерного обчислення до хмарного обчислення. Головна причина для вимушеності продовження експлуатації неактуального на сьогоднішній день Hadoop кластеру – наявність унікальних операцій, що відсутні у GCP або наявні не у повному обсязі.

Еталонна архітектура Big Data інфраструктури наведена на рисунку 2.3.

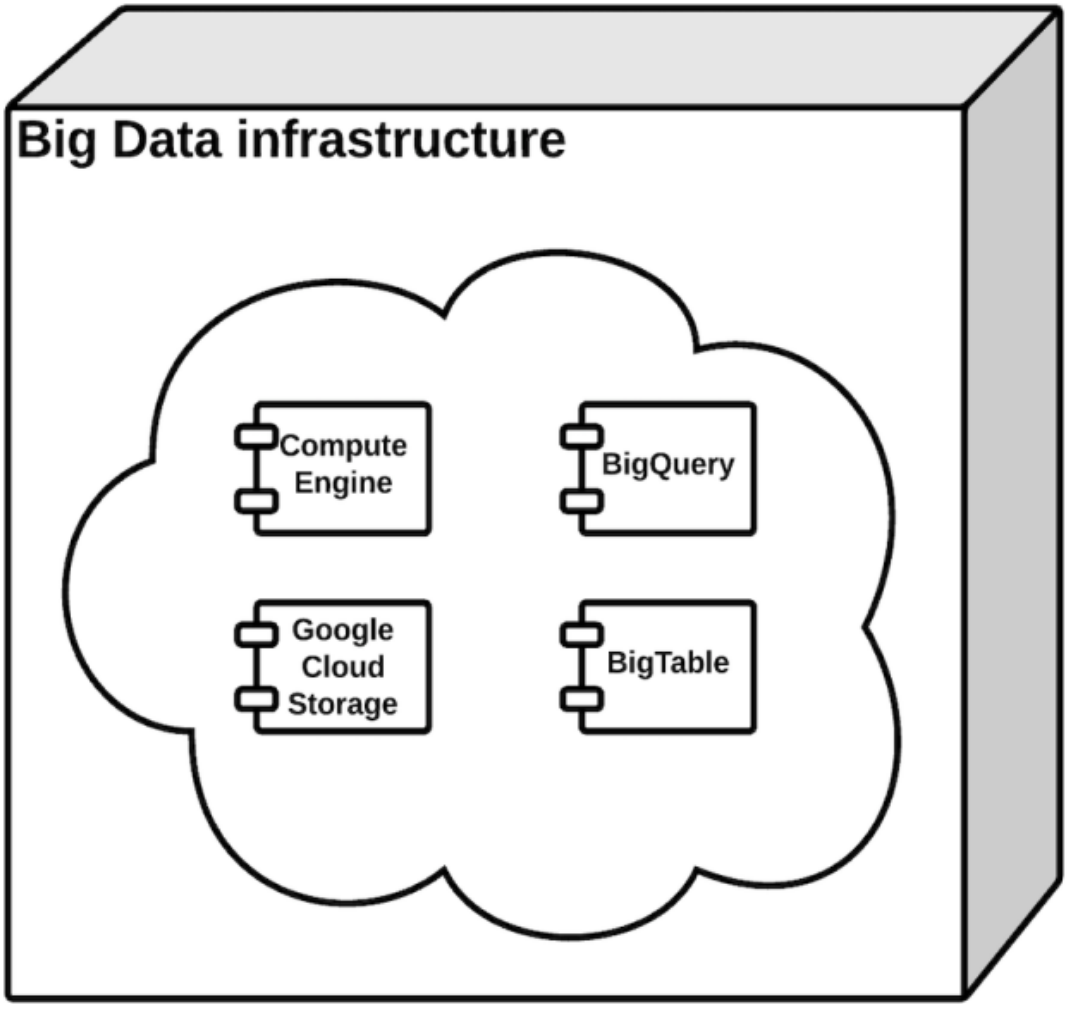

Рисунок 2.3 – Діаграма розгортання еталонної Big Data інфраструктури, що позбавлена Hadoop кластеру

Як видно з рисунку, еталонна архітектура передбачає відмову від Hadoop кластеру та повний перехід до хмарних обчислень.

2.2 Аналіз існуючих технологічних рішень

На даний момент існують дві базових технології, що використовують Hadoop кластер (Рис 2.4):

- шифрування та дешифрування файлів великих об’ємів;
- токенізація та детокенізація великих табличних даних.

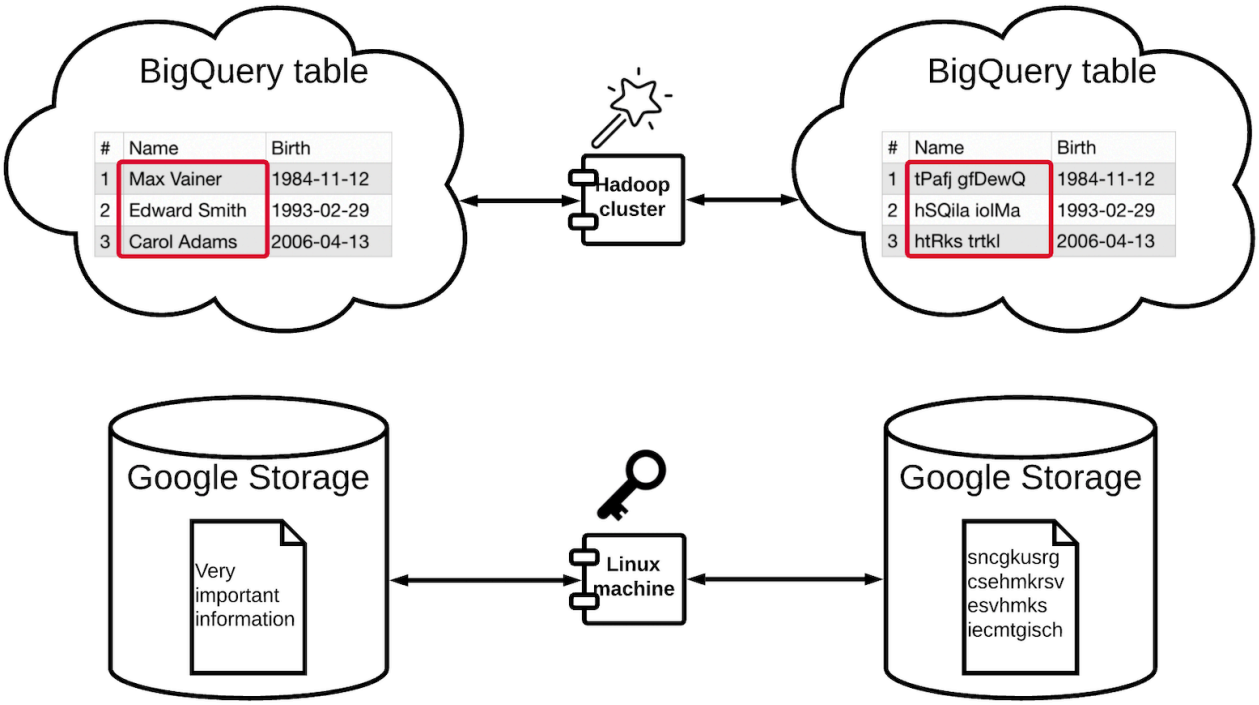

Рисунок 2.4 – Схематичне подання роботи існуючих технологій

Шифрування файлів – це процес обротного перетворення інформації, що міститься у файлах, з метою приховання цієї інформації від неавторизованих осіб. Дешифрування файлів – це процес відтворення попередньо зашифрованої інформації. Дана операція виконується за допомогою утіліти OpenPGP . Даний

клас операцій навантажує головний вузел кластеру, адже по суті являє собою MapReduce програму з суто однією фінальною Reduce функцією (Рис 2.5).

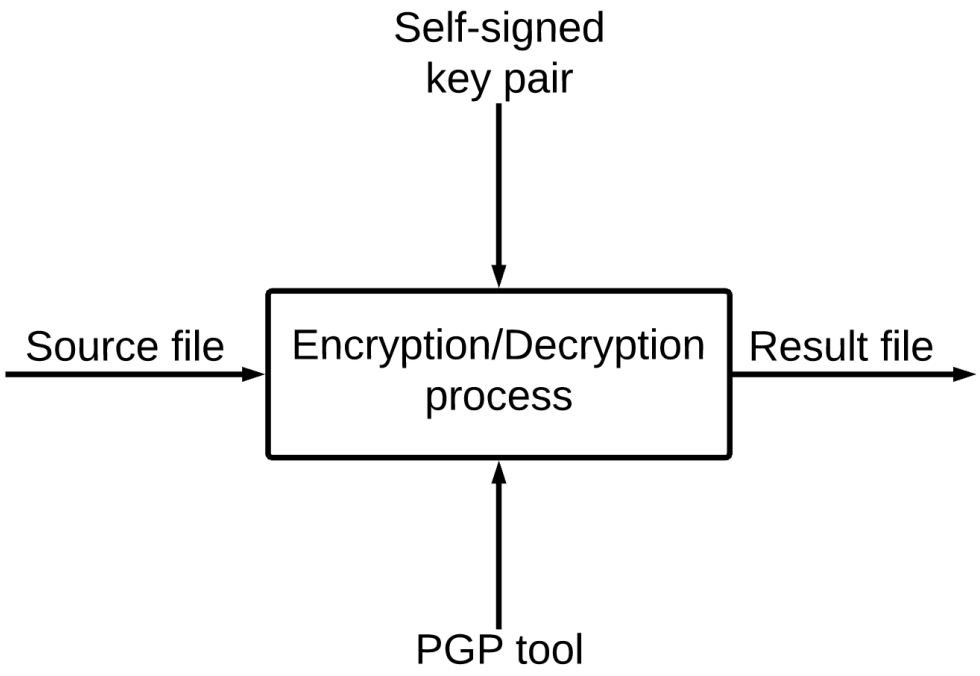

Рисунок 2.5 – IDEF0 процесу шифрування або дешифрування файлу

На даний момент процес шифрування та дешифрування файлі відбувається за наступним алгоритмом (Рис 2.6):

- компанія-партнер завантажує публічний ключ зі сховища, що належить нашій компанії;
- компанія-партнер серіалізує важливі дані до файлу або кількох та локально шифрує цей файл або файли;
- компанія-партнер зберігає цей файл або файли до нашого хмарного сховища;
- спрацьовує файловий сенсор та запускає Spark компонент, що завантажує файл та передає його до PGP компоненту;
- PGP дешифрує файл приватним ключем та віддає назад на обробку до Spark компоненту;

- Spark компонент завантажує розшифрований файл до хмарного файлового сховища.

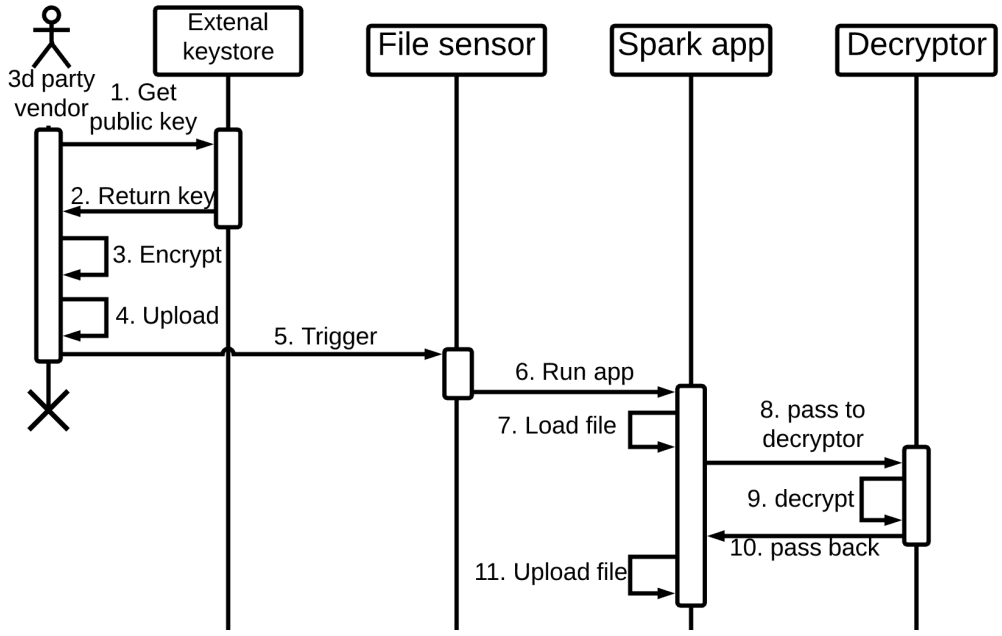

Рисунок 2.6 – Діаграма послідовності роботи існуючого шифрування та дешифрування

Токенізація – це процес заміни конфіденційного елемента даних на неконфіденційний еквівалент, що називають токеном, який не має самостійного сенсу або значення для зовнішнього або внутрішнього використання. Токен – це посилання або ідентифікатор, що зіставляється з конфіденційними даними через систему токенізації. Зіставлення вихідних даних з токеном використовує методи, які унеможливлюють зворотне перетворення токенов в вихідні дані поза системою токенізації, наприклад, з використанням токенов, що створені генератором випадкових чисел.

Доступ до конфіденційних даних зводиться до мінімуму для таких додатків, електронних магазинів, людей і процесів, де реальні дані в системах замінюються токенами, зменшуючи ризик компрометації, випадкового впливу і несанкціонованого доступу до конфіденційних даних. Додатки можуть працювати з токенами, а не з реальними даними, за винятком невеликого числа довірених додатків, яким явно дозволено зворотне перетворення токенов до чинних дані,

коли це строго необхідно для затверджених бізнес-цілей. Системи токенізації можуть бути збудовані і всередині центру обробки даних в якості ізольованого сегмента, що забезпечує безпеку, і як додатковий сервіс постачальника послуг безпеки.

В рамках існуючої системи, токенізація – це шифрування однієї колонки таблиці. Виконується за допомогою бібліотеки Protegrity. Максимально навантажує робочі вузли кластеру, бо по суті являє собою MapReduce програму з суто Map функцією (Рис 2.7).

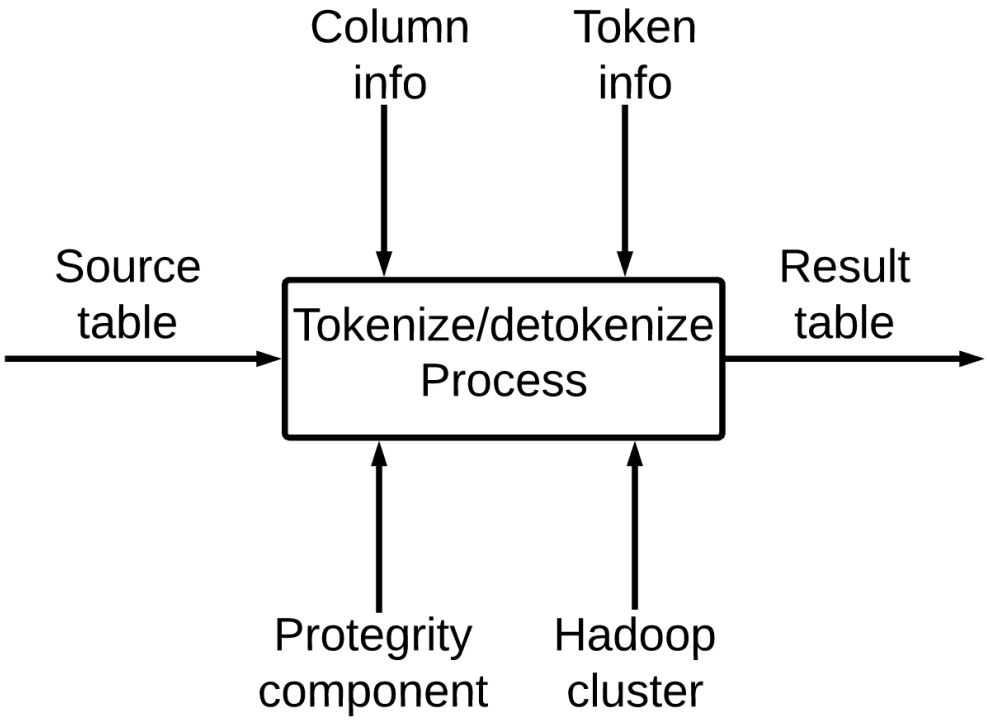

Рисунок 2.7 – IDEF0 існуючого процесу токенізації та детокенізації

На даний момент процес токенізації та детокенізації відбувається за наступним сценарієм (Рис. 2.8):

- Spark компонент починає свою роботу, завантажує таблицю та ініціює роботу MapReduce програми;
- для кожної з колонок, що треба токенізувати, виконуємо процес токенізації, тобто запитуємо токен замість реального значення у Protegrity компоненту через Java інтерфейс;

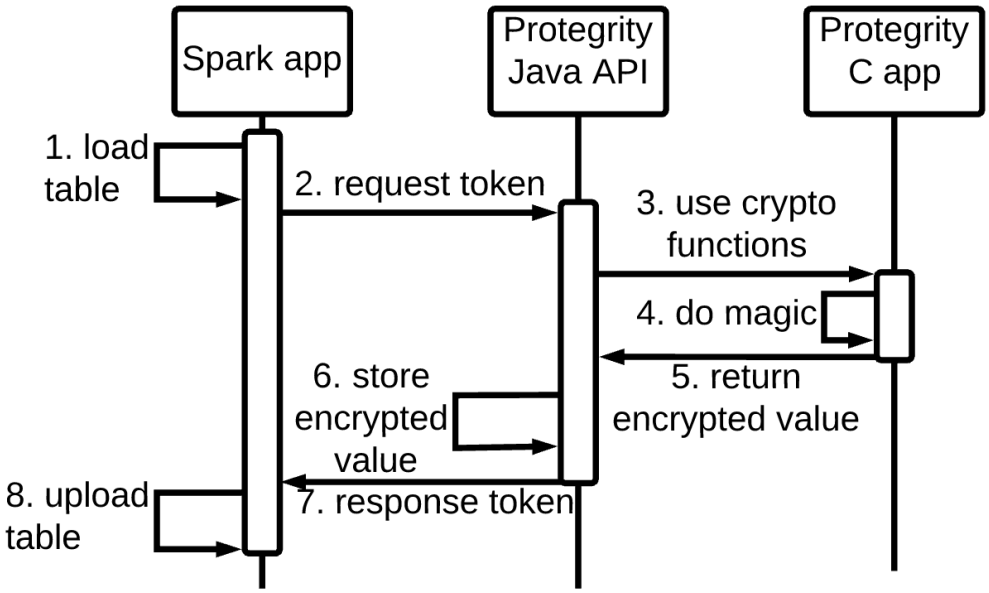

Рисунок 2.8 – Діаграма послідовності роботи існуючої токенізації та детокенізації

- Java інтерфейс звертається до компоненту, що написаний на C та передає йому незашифровані дані;
- компонент, що написаний на C, виконує ряд кріптографічних перетворень та повертає зашифроване значення;
- Java компонент зберігає зашифроване значення та генерує токен, що у подальшому буде відісланий у якості відповіді;
- Java компонент відповідає до Spark компоненту токеном;
- таблиця з новими токенізованими значеннями записується у BigQuery.

2.3 Формування критеріїв ефективності оцінки роботи нових методів

Задля того, щоб мати можливість оцінити та дослідити результаті дослідження, а саме ефективність методу хмарного шифрування та дешифрування файлів великих об’ємів з використанням пари асиметричних ключів та метод хмарної токенізації та детокенізації табличних великих даних з використанням

симетричного ключа, були виділені наступні показники ефективності, які зображені в таблиці 1.1.

Таблиця 1.1 – Показники ефективності нових методів виконання обчислення великих даних

Показник	Опис
T	Загальний час обчислення
Tq	Час знаходження програми у черзі кластеру
Tnet	Час передачі даних за інтернет каналом
Tmap	Час виконання однієї Map операції
Treduce	Час виконання однієї Reduce операції
Nreduce	Кількість Reduce операцій
Nw	Кількість робочив вузлів кластеру
Tpersist	Час запису даних до файлової системи
B	Кінцевий прибуток
Bcredit	Витрати
Bdebit	Прибуток

Головною проблемою усіх обчилень великих даних є час роботи програми задля досягнення кінцевого результату. Адже кількох даних, що обробляються є достатньо великою, розрахунки та обробка даних завжди займають значний час. Саме тому всі оптимізації та нові архітектурні рішення у цій сфері направлені на зменшення часу роботи програми та пришвидшення обробки даних. Отже, головний критерій ефективності при оцінюванні ефективності роботи нових методів – час виконання програм, що виражається як $T \rightarrow min$. Розрахунок цього показника відбувається за наступною формулою:

$$T = Tq + \frac{\sum Tmap}{Nw} + Tnet + max(\frac{Nreduce * \sum Treduce}{Nw}) + Tpresist \quad (1)$$

Підвищити ефективність, тобто зменшити показник *T,* планується за рахунок зменшення показників *Tq, Tmap* та *Treduce,* тобто за рахунок зменшення знаходження програми у черзі, часу обчислення однієї Map функції та часу обчислення однієї Reduce функції.

$$B = Bdebit - Bcredit \quad (2)$$

Не менш важливим критерієм ефективності є кошти, що розразовуються за наступною формулою:

За рахунок зменшення показника T також буде відбуватися підвищення витрат, тобто $Bcredit \rightarrow max$, а також значне підвищення показника Bdebit, тобто $Bdebit \rightarrow max$, за рахунок чого виходимо на оптимальний показник $B \rightarrow optim$.

2.4 Опис нових технологій хмарного шифрування та дешифрування без використання Hadoop кластеру

Головна мета впровадження нової технології шифрування та дешифрування – відмова від використання Hadoop кластеру. Дана відмова змушує використовувати сторонні сервіси для зберігання пари асиметричних ключів, адже раніше ці ключі зберігалися на головному сервері кластеру у сховищі ключів. У якості стороннього сервісу для зберігання ключів було обране рішення від Google – Key Management Service.

Google KMS підтримує інший формат ключів, аніж той, що використовується на даний момент у існуючій системі. Головна відмінність – використання сертифікату X.509, що засвідчує довіренність ключів та зменшує ризики на витік інформації.

X.509 – це стандарт для інфраструктури відкритого ключа та інфраструктури управління привілеями. X.509 визначає стандартні формати даних і процедури розподілу відкритих ключів за допомогою відповідних сертифікатів з цифровими підписами. X.509 має наступну структуру:

- версія;
- серійний номер;
- ідентифікатор алгоритму підпису;
- ім’я того, хто видав сертифікат;
- період дії;
- ім’я суб’єкту;
- алгоритм відкритого ключа;
- відкритий ключ суб’єкта;
- унікальний ідентифікатор видавця;
- унікальний ідентифікатор суб’єкта;
- алгоритм підпису сертифікату;
- підпис сертифікату.

Не менш важливою відмінністю у роботі з KMS є автоматична ротація ключів. Суть така, що після того, як у ключів закінчився термін дії, Google KMS автоматично замінює старі ключів на нові без втручання користувача. Дана технологія підвищує ступінь автоматизації та зменшує кількість рухів та ресурсів, необхідних для підтримки механізмів шифрування.

Також суттєвих змін зазнав сам процес шифрування та дешифрування (Рис. 2.8). Шифрування файлі усе ще відбувається локально. Це вимушена міра безпеки, адже передавати незашифровані дані за інтернет каналом небезпечно. Для шифрування файлів використовується утіліта SSL.

SSL – це криптографічний протокол, що забезпечує безпечний шифрований зв’язок [11]. Він використовує асиметричну криптографію для аутентифікації ключів обміну, симетричне шифрування для збереження конфіденційності, коди аутентифікації повідомлень для цілісності повідомлень. Протокол широко використовувався для обміну миттєвими повідомленнями і передачі голосу через IP в таких додатках, як електронна пошта, інтернет-факс та інші.

Протокол SSL надає «безпечний канал», який має три основні властивості:

- канал є приватним, бо шифрування використовується для всіх повідомлень після простого діалогу, який служить для визначення секретного ключа;
- канал є аутентифікований, адже серверна сторона діалогу завжди аутентифікується, а клієнтська сторона робить це опціонально;
- канал є надійним через те, що транспортування повідомлень включає в себе перевірку цілісності.

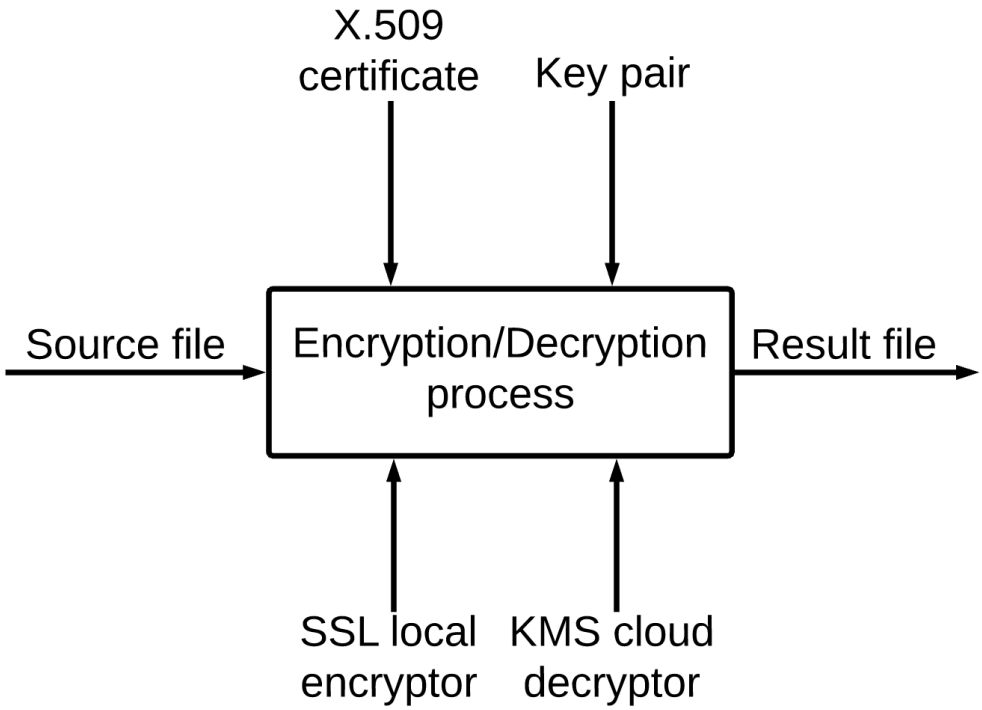

Рисунок 2.8 – IDEF0 нового процесу шифрування та дешифрування

Дешифрування файлів, навпроти, працює на хмарному сервісі на потужностях Google Cloud. Для роботи з хмарним дешифрувальник потребується лише аутентифікуватися на Google Cloud, аби сервіс мав змогу визначити, чи має даний користувач відповідні превілеї. Зазвичай при роботі використовується JSON файл, що містить необхідний токен.

Алгоритм шифрування та дешифрування теж зазнав деяких змін. З використанням нової технології хмарного шифрування та дешифрування, потребується виконати значно меншу кількість рухів для досягнення необхідного результату.

З використанням нової технології процес шифрування та дешифрування файлів відбувається за наступним алгоритмом (Рис. 2.9):

- компанія-партнер завантажує публічний ключ зі сховища, що належить нашій компанії;
- компанія-партнер серіалізує важливі дані до файлу або кількох та локально шифрує цей файл або файли;
- компанія-партнер зберігає цей файл або файли до нашого хмарного сховища;
- спрацьовує файловий сенсор та запускає Python компонент, що відправляє HTTP запит на Google Cloud для розшифрування файлу;
- Google Cloud повертає розшифрований текст назад до Python компоненту, що завантажує файл до хмарного сховища.

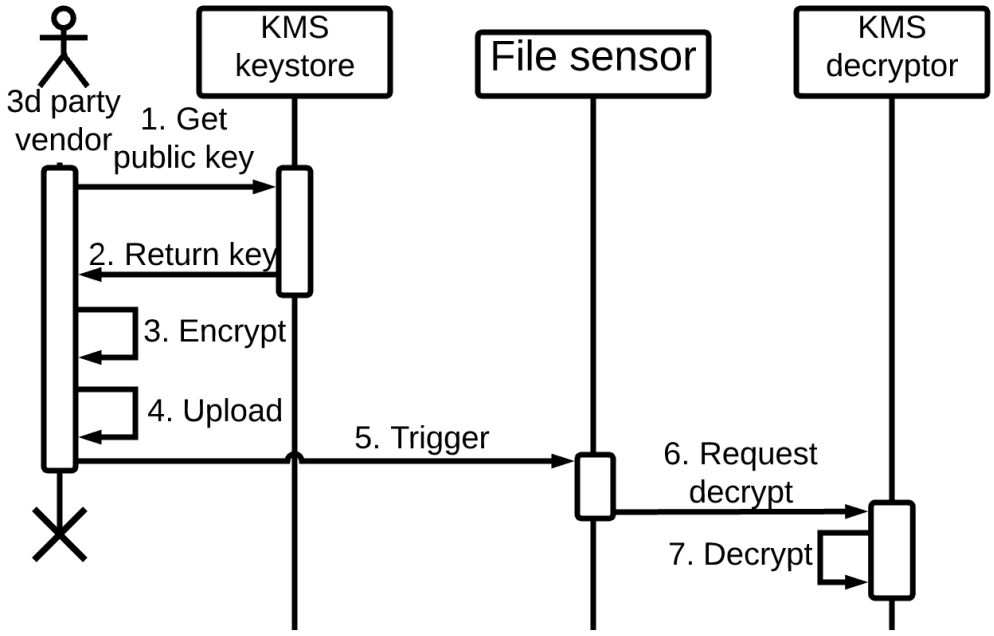

Рисунок 2.9 – Діаграма послідовності роботи нового шифрування та дешифрування

2.5 Опис нового методу хмарного шифрування та дешифрування даних

Для роботи метод потребує виконання деяких передумов:

- потрібен Google-акаунт з підключеним білінгом до Google Cloud сервісів та активованою криптографією;
- у криптографічному модулі Google Cloud має бути хоча б одне сховище ключів.

Якщо дані передумови не виконані, необхідно їх виконати за наступним алгоритмом:

- створити новий Google акаунт чи авторизуватись під існуючим;
- зайти у Google Cloud;
- підключити білігнову інформацію, наприклад, банківську карту для оплати послуг хмарних сервісів;
- активувати криптографічний модуль у Google Cloud;
- створити сховище ключів у KMS (Рис 2.10).

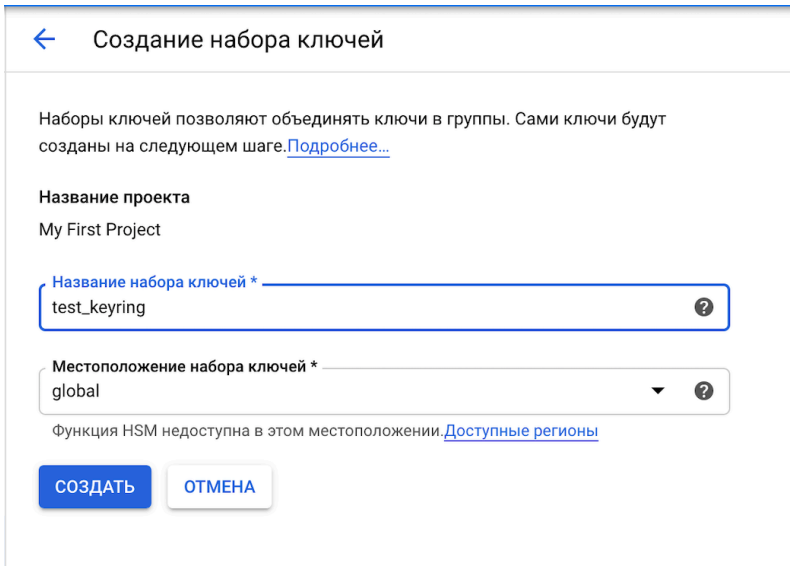

Рисунок 2.10 – Створення ключів у KMS

Якщо усі передумови виконано, можна переходити власне до роботи з новим методом, а саме до шифрування та дешифрування файлів.

Невід'ємним компонентом шифрування та дешифрування завжди є ключі. Так як в нас використовується асиметричне шифрування з сертифікатими, що підписані третьою особою, нам потребуються пара з публічного та приватного ключів, а також сертифікат, що підтверджує походження та цілісність цих ключів.

$$K = \{ K_{cert}, K_{pub}, K_{priv} \}, \quad (3)$$

де K – множина усіх ключів та сертифікатів,

K_{cert} – сертифікат за стандартом X.509,

K_{pub} – публічний (відкритий) ключ,

K_{priv} – приватний (закритий) ключ.

Надзвичайно важливим показником з точки зору безпеки є етап генерації ключей. Користувач має змогу опосередковано керувати цим процесом шляхом вибору довжини ключа та алгоритму генерації ключа:

$$\{ K_{pub}, K_{priv} \} = gen(A_{gen}, L_{key}), \quad (4)$$

де K_{pub} – публічний (відкритий) ключ,

K_{priv} – приватний (закритий) ключ,

gen – процедура генерації пари ключів,

A_{gen} – алгоритм генерації пари ключів,

L_{key} – довжина ключа.

Google дає змогу не вибирати алгоритм генерації ключа та використовувати стандартний. За замовчуванням завжди використовується алгоритм генерації на базі еліптичних кривих. Довжину ключа завжи доводиться обирати самостійно. Не рекомендується нехтувати вибором параметрів генерації ключів, особливо довжиною, адже ці вихідні дані впливають на кінцевий результат, а саме на

швидкість процесу шифрування та дешифрування, а також на криптографічну стійкість отриманих результатів.

Google рекомендує встановлювати довжину ключів не менше ніж 3072 байти, адже дані, що зашифровані ключем довжиною 2048 байт, мають недостатню криптографічну стійкість на сьогоднішній день. Але остаточний вибір довжини ключа все одно залишається за користувачем.

Ще один показник, на який може впливати користувач – це вибір алгоритму шифрування та дешифрування. Стандартний алгоритм шифрування, що пропонує Google – RSA у декількох його варіаціях. Шифрування відбувається за наступною формулою:

$$T_{enc} = A_{enc}(T_{raw}, K_{pub}), \ (5)$$

де T_{enc} – кінцевий результат шифрування у вигляді шифротексту,

K_{pub} – публічний (відкритий) ключ,

A_{enc} – алгоритм шифрування,

T_{raw} – початковий (незашифрований) текст.

Дешифрування відбувається за аналогічною схемою:

$$T_{raw} = A_{enc}(T_{enc}, K_{priv}), \ (6)$$

де T_{enc} – початковий аргумент дешифрування у вигляді шифротексту,

K_{priv} – приватний (закритий) ключ,

A_{enc} – алгоритм шифрування,

T_{raw} – вихідний (розшифрований) текст.

Таким чином, метод хмарного шифрування та дешифрування файлів без використання Hadoop кластеру поділяється на наступні кроки:

Крок 1. Переконатися, що виконані усі передумови, а саме є авторизований Google-акаунт з підключеним білінгом до Google Cloud сервісів та активованою

криптографією, а у криптографічному модулі Google Cloud є хоча б одне сховище ключів.

Крок 2. Вибрати ключі, що будуть використовуватися для шифрування та дешифрування даних. Якщо необхідні ключі вже у наявності, переходимо до кроку 4.

Крок 3. Вибрати довжину ключів L_{key} та алгоритм генерації ключей A_{gen}. Згенерувати пару асиметричних ключів, що будуть підписані сертифікатом X.509. $K = \{ K_{cert}, K_{pub}, K_{priv} \}$.

Крок 4. Експортувати публічний ключ K_{pub} з Google KMS разом із сертифікатом K_{cert}. Сертифікат одразу містить інформацію про алгоритм шифрування A_{enc}.

Крок 5. Зашифрувати цінні дані T_{raw} за допомогою утіліти SSL, що автоматично перевірить сертифікат K_{cert} та дістане усю необхідну інформацію про алгоритм. На виході отримуємо шифротекст. $T_{enc} = A_{enc}(T_{raw}, K_{pub})$.

Крок 6. Розшифрувати зашифровані дані T_{enc} за допомогою хмарних сервісів Google. Для цього треба лише відправити HTTP запит з шифротекстом T_{enc} та ідентифікатором ключа. На виході отримуємо розшифровані дані. $T_{raw} = A_{enc}(T_{enc}, K_{priv})$.

2.6 Опис нових технологій токенізації та детокенізації без використання Hadoop кластеру

Головна мета впровадження нової технології токенізації та детокенізації – відмова від використання Hadoop кластеру. Планувалось реалізувати аналогічний функціонал на Google BigQuery через User-defined functions. Для цього був проведений реверс-інжиніринг існуючих компонентів, в процесі якого було виявлено, що:

- бібліотека Protegrity, яку використовують Big Data розробники – це лише зовнішнє Java API [12];

- зовнішнє Java API використовує shared memory для контакту з іншим модулем, який містить у собі усю логіку, яка відповідна за усі основні криптографічні операції;
- інший модуль реалізований на низьорівневій мові програмовуання, імовіно на C, саме тому подальший реверс інженіринг вважається недоцільним.

Для реалізації аналогічного функціоналу був запропонований варіант з використанням AEAD шифрування у BigQuery.

AEAD-режими блочного шифрування – це клас блокових режимів шифрування, при якому частина повідомлення шифрується, частина залишається відкритою, і все повідомлення повністю аутентифікується. Блоковий шифр – це різновид симетричного шифрування, який оперує групами біт фіксованої довжини, які називають блоками, характерний розмір яких змінюється в межах від 64 до 256 біт. Якщо вихідний текст, або його залишок, менше розміру блоку, перед шифруванням його доповнюють. Фактично, блоковий шифр являє собою підстановку на алфавіті блоків, яка, як наслідок, може бути моно- або поліалфавітною. Блоковий шифр є важливою компонентою багатьох криптографічних протоколів і широко використовується для захисту даних, що передаються по мережі.

Функції шифрування AEAD у BigQuery дозволяють створювати набори ключів, які містять ключі для шифрування та розшифровки, використовувати ці ключі для шифрування та дешифрування окремих значень у таблиці та обертання ключів у межах набору ключів [13].

На практиці це позначає навність інструментів у BigQuery, що потенційно можуть замінить існуючу токенізацію та детокенізацію колонок. Даний підхід достатньо суттєво відрізняється від існуючого рішення (Рис 2.11). Головним плюсом цієї технології є швидкість роботи. Готовний недолік – це необхідність перетокенізовуати усі поточні дані. Також не можна не відмітити заміну Scala чи Java кодування на SQL синтаксис, що робить використання цієї технології значно простіше та доступніше.

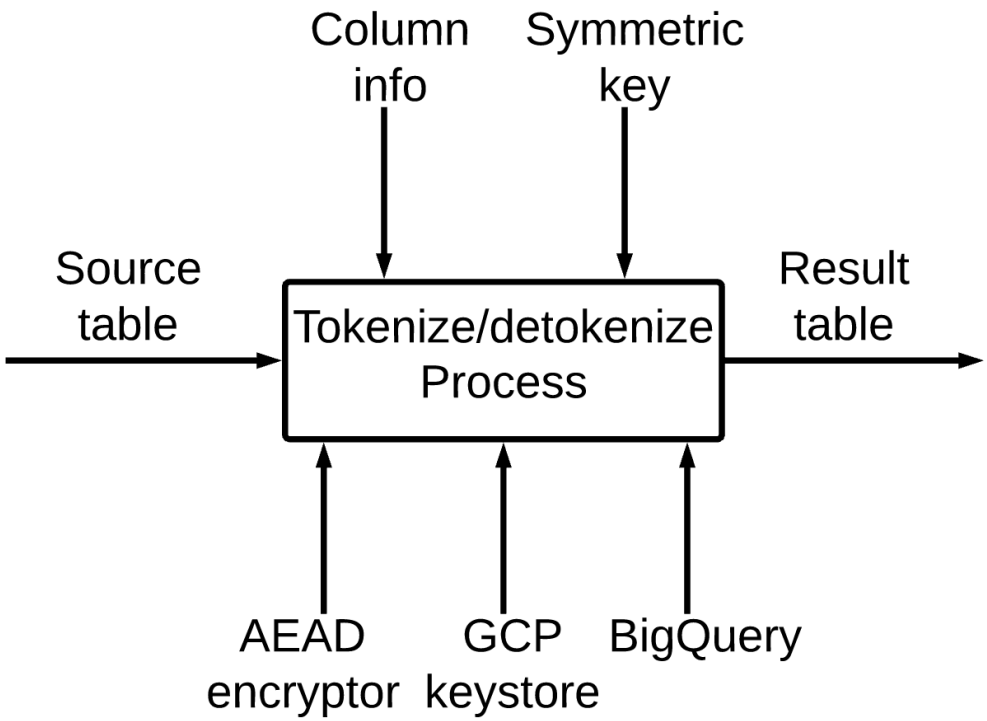

Рисунок 2.11 – IDEF0 нової технології токенізації та детокенізації табличних даних

Головною і найбільш суттєвою відмінністю є засіб зберігання та менеджменту ключей. Раніше це питання віддавалось повністю на обробку компоненту Protegrity, при чому ключі шифрування завжди були постіними. Відтепер ключі можна зберігати у будь-якому місці. Тобто це може бути:

- локальне сховище;
- хмарне сховище;
- база даних;
- BigQuery таблиця.

Для кожного з варіантів зберігання потребується ручне налаштування прав доступу до ключів, що, певно, створює деякі труднощі у порівнянні з існуючим рішенням. З іншої сторони, такий підхід дає більше свободи для керування процесом шифрування та дешифрування колонок, тобто при правильному рівні автоматизації, підтримка такого нового рішення не буде суттєво відрізнятися від того, що ми маємо зараз.

З використанням нової технології процес токенізації та детокенізації табличних даних відбувається за наступним алгоритмом (Рис. 2.12):

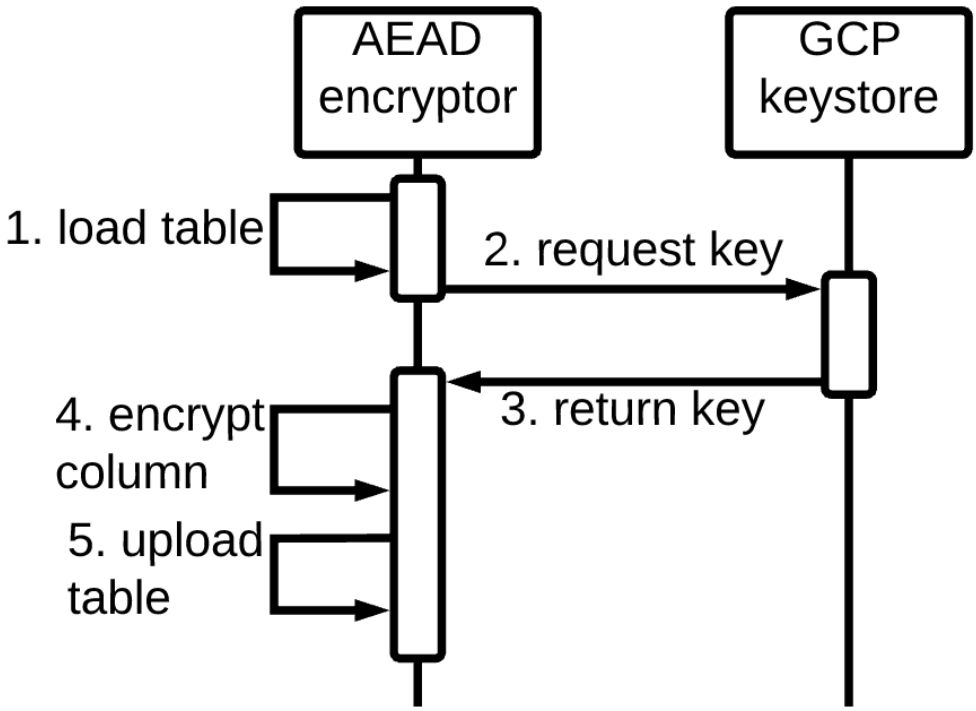

Рисунок 2.12 – Діаграма послідовності роботи нової токенізації та детокенізації

- скрипт, що трансформує дані та використовує AEAD функції, завантажує необхідну таблицю, тобто робить до неї запит;
- для кожної колонки, що треба зашифрувати, запитуємо необхідний ключ з відповідного джерела;
- шифруємо чи дешифруваємо дані отриманим симетричним ключем;
- зберігаємо результати до BigQuery таблиці.

Під AEAD encryptor маємо на увазі будь-яку програму, чи SQL скрипт, що робить запит до BigQuery таблиці та використовує AEAD функції для шифрування чи дешифрування табличних колонок.

2.7 Опис нового методу хмарної токенізації та детокенізації даних без використання Hadoop кластеру

Для роботи метод потребує виконання деяких передумов:

- потрібен Google-акаунт з підключеним білінгом до Google Cloud сервісів та активованим модулем аналізу великих даних;
- користувач потребує необхідних прав доступу, а саме права на читання та на запис;
- BigQuery має містити дані, над якими потребується виконання операцій шифрування чи дешифрування колонок.

Якщо дані передумови не виконані, необхідно їх виконати за наступним алгоритмом:

- створити новий Google акаунт чи авторизуватись під існуючим;
- зайти у Google Cloud;
- підключити білігнову інформацію, наприклад, банківську карту для оплати послуг хмарних сервісів;
- активувати BigQuery;
- імпортувати чи сгенерувати дані, над якими потребується виконання операцій шифрування чи дешифрування колонок.

Якщо усі передумови виконано, можна переходити власне до роботи з новим методом, а саме до шифрування та дешифрування колонок табличних даних.

Невід’ємним компонентом шифрування та дешифрування завжди є ключі. В даному випадку ми будемо використовуємо ключі, що були згенеровані самим BigQuery, тому довжина ключа фіксована, 16 байт, тобто 128 біт. Так як ми використовуємо AEAD шифрування, нам також потребується додадковий блок тексту, що не буде шифруватися та залишиться незмінним. Отже, набір інструментів для шифрування або дешифрування складається з двох частин – ключ довжиною 16 байт та блок додадкової інформації.

$$K = \{ K_{sym}, K_{add} \},$$

(6)

де K – множина інструменів, необхідних для шифрування та дешифрування,

K_{sym} – симетричний ключ,

K_{add} – блок додадкової інформації.

Надзвичайно важливим показником з точки зору безпеки є етап ствоення ключів. Нажаль, користувач не має можливості у повному обсязі керувати процесом генерації ключів, адже на даний момент в наявності для генерації лише алгоритм лічильнику з аутентифікацією Галуа. У будь-якому випадку, Google дає змогу імпортувати власний симетричний ключ. Для такої нагоди в наявності є два алгоритми шифрування – вже відомий нам лічильник з аутентифікацією Галуа з довжиною ключа 16 або 32 байти, а також режим зчеплення блоків шифротекста відомий як CBC. Для CBC довжина ключів може варіюватися у діапазоні 16, 24 або 32 байти.

$$\{ K_{sym} \} = gen(A_{gen}, L_{key}), \ (7)$$

де K_{sym} – симетричний ключ,

gen – процедура генерації ключа,

A_{gen} – алгоритм генерації ключа та шифрування,

L_{key} – довжина ключа.

Ще один показник, на який може впливати користувач – це вибір алгоритму шифрування та дешифрування. Якщо ми користуємось функцією, що генерує ключі, вибір відсутній та ми вимушені користуватися ключами, що згенеровані для алгоритму лічильника з аутентифікацією Галуа. У раз, якщо ми імпортували власний ключ, вибір алгоритму шифрування залишається за нами. Шифрування відбувається за наступною формулою:

$$T_{enc} = A_{enc}(T_{raw}, K_{sym}, K_{add}), \ (8)$$

де T_{enc} – кінцевий результат шифрування у вигляді шифротексту,

K_{sym} – симетричний ключ,

A_{enc} – алгоритм шифрування,

T_{raw} – початковий (незашифрований) текст,

K_{add} – блок додадкової інформації.

Дешифрування відбувається за аналогічною схемою, але без використання блоку додадкової інформації:

$$T_{raw} = A_{enc}(T_{enc}, K_{sym}), \ (9)$$

де T_{enc} – початковий аргумент дешифрування у вигляді шифротексту,

K_{sym} – симетричний ключ,

A_{enc} – алгоритм шифрування,

T_{raw} – вихідний (розшифрований) текст.

Таким чином, метод хмарної токенізації та детокенізації табличних даних без використання Hadoop кластеру поділяється на наступні кроки:

Крок 1. Переконатися, що виконані усі передумови, а саме є авторизований Google-акаунт з підключеним білінгом до Google Cloud сервісів та активованим модулем аналізу великих даних, даний акаунт має достатньо прав доступу, а саме права на читання та запис до таблиць, а також ми маємо необхідні дані для токенізації або детокенізації у BigQuery.

Крок 2. Вибрати ключі, що будуть використовуватися для шифрування та дешифрування даних. Якщо необхідні ключі вже у наявності, переходимо до кроку 4.

Крок 3. Імпортувати до BigQuery необхідний симетричний ключ K_{sym}, що призначений для шифрування алгоритмом зчеплення блоків шифротекста з

довжиною 16, 24 або 32 байти, або лічильником з аутентифікацією Галуа довжиною 16 або 32 байти.

Крок 4. Створити блок додадкової інформації K_{add}, що буде додаватися до кожного шифротексту.

Крок 5. Зашифрувати цінні дані T_{raw} за допомогою функції AEAD.ENCRYPT та ключа K_{sym} з використанням блоку додадкової інформації K_{add}. Ключ K_{sym} містить дані про алгоритм A_{enc}, яким був зашифрований запис. На виході отримуємо шифротекст. $T_{enc} = A_{enc}(T_{raw}, K_{sym}, K_{add})$.

Крок 6. Розшифрувати зашифровані дані T_{enc} за допомогою симетричного ключа K_{sym} та виклику функції AEAD.DECRYPT_STRING або AEAD.DECRYPT_BYTES. Ключ K_{sym} містить дані про алгоритм A_{enc}, яким був зашифрований запис. На виході отримуємо розшифровані дані. $T_{raw} = A_{enc}(T_{enc}, K_{sym})$.

3 ДОСЛІДЖЕННЯ НОВИХ ТЕХНОЛОГІЙ ТА МЕТОДІВ ХМАРНОЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ БЕЗ ВИКОРИСТАННЯ HADOOP КЛАСТЕРУ

У результаті досліджень існуючої інфраструктури для аналізу великих даних, які були проведені згідно з описаними методологіями організації та проведення досліджень у рамках поставленої наукової задачі, були отримані дані, алгоритми та опис існуючих методів та технологій. На їх основі були впроваджені нові технології та розроблені нові методи хмарного шифрування та дешифрування файлів з без Hadoop кластеру та токенізації і детокенізації великих табличних даних без Hadoop кластеру. Результати роботи нових методів будуть досліджені та проаналізовані у цьому розділі наукової атестаційної роботи.

3.1 Методика проведення досліджень нових методів хмарного шифрування і дешифрування файлів та хмарної токенізації і детокенізації великих табличних даних

Задля того, щоб коректно обґрунтувати актуальність та практичну вигоду від нових методів, отримані результати необхідно ретельно дослідити та проаналізувати. У першому розділі атестаційної роботи були визначені критерії ефективності – зменшення часу роботи програми та пришвидшення видачі результатів. Вибір методик для проведення дослідження результатів цієї наукової роботи небагато – це тестування продуктивності.

Тестування продуктивності – це тестування, яке проводиться з метою визначення, як швидко працює обчислювальна система або її частина під певним навантаженням [14]. Також може служити для перевірки і підтвердження інших атрибутів якості системи, таких як масштабованість, надійність і споживання ресурсів.

У тестуванні продуктивності розрізняють наступні напрямки:

- навантажувальний тест;
- стрес тестування;
- тестування стабільності;
- конфігураційне тестування.

Для оцінки ефективності роботи нового методу було прийнято рішення проводити навантажувальне тестування на вибірках даних різних розмірів. Навантажувальне тестування – це найпростіша форма тестування продуктивності. Тестування навантаження зазвичай проводиться для того, щоб оцінити поведінку програми під заданим очікуваним навантаженням. Цим навантаженням може бути, наприклад, очікувана кількість одночасно працюючих користувачів додатка, що здійснюють вказану кількість транзакцій за інтервал часу. Такий тип тестування зазвичай дозволяє отримати час відгуку всіх найважливіших бізнес-транзакцій. У разі спостереження за базою даних, сервером додатків, мережею цей тип тестування може також ідентифікувати деякі вузькі місця додатку.

3.2 Аналіз ефективності нового методу хмарного шифрування та дешифрування файлів

Головна задача тестування – визначити, наскільки новий метод відповідає критеріям ефективності. Для цього нам необхідне порівняння старих результатів з новими.

Для об’єктивності результатів були проведені навантажувальні тести за старим та за новим методом. Використовувались файли розміром 1МБ, 100МБ та 1ГБ. Для шифрування використовувався ключ довжиною 2048 біт для обох методів. Для старого шифрування було проведено по 10 тестів із заміром часу на операцію. Було розраховано середнє значення, результати занесені у таблицю 3.1.

Таблиця 3.1 – Результати тестування старого методу шифрування та дешифрування файлів з використанням Hadoop кластеру

Розмір файлу	Шифрування (старе)	Дешифрування (старе)
1 МБ	менше 1с	менше 1с
100 МБ	15с	15с
1 ГБ	3 хв 40 с	3 хв 20 с

Для нового шифрування було проведено по 3 тести із заміром часу на операцію. Було розраховано середнє значення, результати занесені у таблицю 3.2.

Таблиця 3.2 – Результати тестування нового методу шифрування та дешифрування файлів без використання Hadoop кластеру

Розмір файлу	Шифрування (нове)	Дешифрування (нове)
1 МБ	менше 1с	менше 1с
100 МБ	17с	12с
1 ГБ	3 хв 50 с	1 хв 17 с

На базі результатів тестування можна зробити наступні висновки:

- швидкість шифрування майже не відрізняється в обох методах, що пов’язано з тим, що в обох випадках файл шифрується локально;
- маленьку різницю в результатах шифрування у пользу старого методу можна пояснити тим, що старий метод не витрачає зайвий час на валідацію сертифікату, на відміну від нового;
- хмарне дешифрування не дає значного виграшу при роботі з маленькими файлами;

- хмарне дешифрування дає значний приріст швидкості при роботі з великими файлами.

З цього всього можна зробити висконовок, що використання нового методу токенізації і детокенізації великих табличних даних без Hadoop кластеру дає значний виграш часу.

3.3 Аналіз ефективності нового методу хмарної токенізації і детокенізації великих табличних даних

Головна задача тесутвання – визначити, наскільки новий метод відповідає критеріям ефективності. Для цього нам необхідне порівняння старих результатів з новими.

Для об’єктивності результатів були проведені навантажувальні тести за старим та за новим методом. Використовувались вибірки даних розміром 100 тисяч строк, 1 мільйон строк та 10 мільйонів строк. Для тестів використовувалась відкриті публічні дані з BigQuery public data, а саме інформація про работу таксі у Нью-Йорку за 2018 рік.

Тести старої токенізації проводились у час-пік на Dev кластері. Було розраховано середнє значення, результати занесені у таблицю 3.3.

Таблиця 3.3 – Результати тестування старого методу токенізації і детокенізації великих табличних даних

Розмір датасету	Spark+Protegrity 1 column tok	Spark+Protegrity 3 column tok
100 000	Queued 2 min Map+net 1 min	Queued 4 min Map+net 2 min
1 000 000	Queued 2 min Map+net 3 min	Queued 20 sec Map+net 4 min
10 000 000	Queued 16 min Map+net 4 min	Queued 2 min Map+net 6 min

Тести нових методів токенізації проводились у BigQuery. Було розраховано середнє значення, результати занесені у таблицю 3.4.

Таблиця 3.4 – Результати тестування нового методу токенізації і детокенізації великих табличних даних

Розмір датасету	BigQuery AEAD 1 column tok	BigQuery AEAD 3 column tok
100 000	Queued 0 sec Map+net 4.3 sec	Queued 0 sec Map+net 4.4 sec
1 000 000	Queued 0 sec Map+net 24.2 sec	Queued 0 sec Map+net 24.1 sec
10 000 000	Queued 0 sec Map+net 1 min 26 sec	Queued 0 sec Map+net 1 min 26 sec

На базі результатів тестування можна зробити наступні висновки:

- різниця у часі роботи самої програми майже не змінюється від об’єму даних, бо операція токенізації повністю паралельна, адже вона представляє собою суто функцію Map, тому більшість затрат часу припадає на очікування у черзі кластеру;
- час очікування у черзі кластеру – дуже нестабільний показник, що залежить від поточного навантаження кластеру;
- BigQuery одразу виділяє квоту на обробку даних, тому не витрачається час на очікування у черзі;
- зі збільшенням кількості колонок для токенізації час обчислення не збільшується.

З цього всього можна зробити висконовок, що використання нового методу токенізації і детокенізації великих табличних даних без Hadoop кластеру дає значний виграш часу.

3.4 Аналіз підвищення навантаження на BigQuery

Аналіз підвищення навантаження на BigQuery виконувався шляхом виконання запитів з шифруванням двох стовпців та запитів без шифрування. Дані тести були виконані для того щоб зрозуміти зростання навантаження на BigQuery з додаванням функцій шифрування у запит.

Використовувались вибірки даних розміром 100 тисяч строк, 1 мільйон строк та 10 мільйонів строк. Для тестів використовувалась відкриті публічні дані з BigQuery public data, а саме інформація про работу таксі у Нью-Йорку за 2018 рік. Тести проводились у BigQuery.

Було розраховано середнє значення, результати занесені у таблицю 3.5.

Таблиця 3.5 – Порівняння навантаження на BigQuery з шифруванням та без

Розмір датасету	Select без шифрування	Select з шифруванням
100 000	2.2 sec	4.3 sec
1 000 000	11.5 sec	24.3 sec
10 000 000	25.5 sec	1 min 26 sec

З цього всього можна зробити висконовок, що використання нового методу токенізації і детокенізації великих табличних даних без Hadoop кластеру підвищує навантаження на BigQuery приблизно у 2-3 рази, коли мова йдеться про звичайний select без операцій агрегації. Тим не менш, використання нового методу вважається більш як доцільним по причині виского приросту швидкості обробки даних.

4 ПРАКТИЧНЕ ВИКОРИСТАННЯ НОВИХ ТЕХНОЛОГІЙ ТА МЕТОДІВ ХМАРНОЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ БЕЗ ВИКОРИСТАННЯ HADOOP КЛАСТЕРУ

Використання нових методів хмарного шифрування та дешифрування файлів без Hadoop кластеру та токенізації і детокенізації великих табличних даних без Hadoop кластеру на практиці є не менш важливою частиною, бо саме практичне використання має приносити переваги. Для демонстрації роботи нових методів, були впроваджені нові технології, а також розроблено спеціальне прогармне забезпечення, яке служить провідником між хмарними сервісами та сервером, хоча і не є повноцінною інформаційною системою.

4.1 Опис програмного забезпечення для тестування та демонстрації роботи методу шифрування та дешифрування файлів без Hadoop кластеру

Для демонстрації роботи нового методу шифрування та дешифрування файлів без Hadoop кластеру був розроблений консольний додаток, що має наступний функціонал:

- авторизація у Google Cloud;
- створення сховища ключів;
- створення пари асиметричних ключів за допомогою хмарного сервісу (Рис 4.1);
- локальне шифрування публічним ключем, яких завантажується з хмарного сервісу разом із сертифікатом, за допомогою SSL (Рис 4.2);
- дешифрування приватним ключем за допомогою хмарного сервісу (Рис. 4.3).

```
>>> asymmetric.create_asymmetric_key("rational-cat-260800", "global", "test_keyring", "test_asym_key1")
Created CryptoKey projects/rational-cat-260800/locations/global/keyRings/test_keyring/cryptoKeys/test_asym_key1.
name: "projects/rational-cat-260800/locations/global/keyRings/test_keyring/cryptoKeys/test_asym_key1"
purpose: ASYMMETRIC_DECRYPT
create_time {
  seconds: 1575283019
  nanos: 682928614
}
version_template {
  protection_level: SOFTWARE
  algorithm: RSA_DECRYPT_OAEP_2048_SHA256
}
```

Рисунок 4.1 – Створення пари асиметричних ключів у консольному додатку

Комунікація з хмарними сервісами виконується за допомогою Google Cloud API. Програмний інтерфейс Google Cloud є ключовою частиною Cloud Cloud Platform, що дозволяє легко отримувати доступ до всьогоспектру послег Google Cloud, від доступу до пам’яті, до аналізу зображень на основі машинного навчання, а також до додатків Cloud Platform.

```
>>> asymmetric.encrypt_rsa(b"secret_message", "projects/rational-cat-260800/locations/global/keyRings/test_keyring/cryptoKeys/test_asym_key1/cryptoKeyVersions/1")
b'h\x9f"\xeag\xa8\x8a\x0c(\xe56\xb7\xb5#feD\xd3N\xb1\x01\x9d\x83\xbc\xf7[\x87S\x9d=\x14]#]\x99\n\xceQ\xbf\xfc\xad\xc2,\xde6VHN\x8b\xba\x06\xc74\xb1\xbdv\xd1|\xfb\xb2\xdc\xe7Y\xcf)\nw\xf9O;\xce|B\xca\xee\xc5\xd1\x0c\x19\x0bw;E\xe1\x0f\x89;\xbc0\xc3\x85\x84\xde\x92n\xe7|\xba\xc4\x8b\xd6\xf8\xdb\xbb|\xf0\'\xeb\xbf3\xf6jU\x91\xc7\xcb\x89s\x96b\x91\xeb\xc8Y\xc0\xc6H,\xcf:\xa4#A\xeaN\xd6q\x89G$Y[\x12\x81\xad\x04\xc6q\xfb\xe7+&\x82~B\xd2\xd40\xd9\xc9?\xb8\x87\xb4\x8d\xb4\x1fe(\xd9\x88\xf5\xed\xc0.-\x96\x94i\x05\xea\xf9\xde\xdc\xba\x97\x82\x9d\x98J\x1e\xca\xecK\xd1\xd4\x9e\xd9\xfd%\xec%\x04N\x87\x83\xef3\x8b\x96APL-\x97\xc8\t\xd66\x01M\xf8\xff\xa08\x0e\xd7\xcaM\x1b{7\xa1\x1e\xf1\x91\x82\xac b\xf1{\xa1_\x94\xeaA\x16\xbd\xa9\xf3h0\xc4u\x94'
```

Рисунок 4.2 – Шифрування у консольному додатку

Більшість хмарних API надають вам детальну інформацію про використання цього API вашого проекту, включаючи рівні трафіку, частоту помилок і навіть затримки, що допомагає вам швидко триажувати проблеми з програмами, які використовують служби Google. Цю інформацію можна переглянути на інформаційній панелі API консолі Cloud Platform.

```
>>> asymmetric.encrypt_rsa(b"secret_message", "projects/rational-cat-260800/locations/global/keyRings/test_keyring/cryptoKeys/test_asym_key1/cryptoKeyVersions/1")
b'h\x9f"\xeag\xa8\x8a\x0c(\xe56\xb7\xb5#feD\xd3N\xb1\x01\x9d\x83\xbc\xf7[\x87S\x9d=\x14]#]\x99\n\xceQ\xbf\xfc\xad\xc2,\xde6VHN\x8b\xba\x06\xc74\xb1\xbdv\xd1|\xfb\xb2\xdc\xe7Y\xcf)\nw\xf9O;\xce|B\xca\xee\xc5\xd1\x0c\x19\x0bw;E\xe1\x0f\x89;\xbc0\xc3\x85\x84\xde\x92n\xe7|\xba\xc4\x8b\xd6\xf8\xdb\xbb|\xf0\'\xeb\xbf3\xf6jU\x91\xc7\xcb\x89s\x96b\x91\xeb\xc8Y\xc0\xc6H,\xcf:\xa4#A\xeaN\xd6q\x89G$Y[\x12\x81\xad\x04\xc6q\xfb\xe7+&\x82~B\xd2\xd40\xd9\xc9?\xb8\x87\xb4\x8d\xb4\x1fe(\xd9\x88\xf5\xed\xc0.-\x96\x94i\x05\xea\xf9\xde\xdc\xba\x97\x82\x9d\x98J\x1e\xca\xecK\xd1\xd4\x9e\xd9\xfd%\xec%\x04N\x87\x83\xef3\x8b\x96APL-\x97\xc8\t\xd66\x01M\xf8\xff\xa08\x0e\xd7\xcaM\x1b{7\xa1\x1e\xf1\x91\x82\xac b\xf1{\xa1_\x94\xeaA\x16\xbd\xa9\xf3h0\xc4u\x94'
```

Рисунок 4.3 – Дешифрування у консольному додатку

Додаток створено за допомогою мови програмування Python (Рис. 4.4). Python – це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника та читабельність коду. Синтаксис ядра Python мінімальний. При цьому стандартна бібліотека включає велику кількість корисних функцій.

```
def decrypt_rsa(ciphertext, key_name):
    """
    Decrypt the input ciphertext (bytes) using an
    'RSA_DECRYPT_OAEP_4096_SHA256' private key stored on Cloud KMS
    """
    client = kms_v1.KeyManagementServiceClient()
    response = client.asymmetric_decrypt(key_name, ciphertext)
    return response.plaintext


def encrypt_rsa(plaintext, key_name):
    """
    Encrypt the input plaintext (bytes) locally using an
    'RSA_DECRYPT_OAEP_4096_SHA256' public key retrieved from Cloud KMS
    """
    # get the public key
    client = kms_v1.KeyManagementServiceClient()
    response = client.get_public_key(key_name)
    key_txt = response.pem.encode('ascii')
    public_key = serialization.load_pem_public_key(key_txt, default_backend())

    # encrypt plaintext
    pad = padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),
                       algorithm=hashes.SHA256(),
                       label=None)
    return public_key.encrypt(plaintext, pad)
```

Рисунок 4.4 – Частина вихідного коду програми

Python підтримує структурне, об'єктно-орієнтоване, функціональне, імперативне та орієнтоване на аспекти програмування [15]. Основними архітектурними особливостями є динамічне введення тексту, автоматичне управління пам'яттю, повна самоаналіз, механізм обробки винятків, підтримка багатопотокових обчислень та структури даних високого рівня. Підтримується поділ програм на модулі, які, в свою чергу, можна об'єднати в пакети.

Python – дуже популярна пова програмування у Big Data. Вона дозволяє створювати швидкі та ефективні рішення, а програмні інтерфейси виходять

простими та зрозумілими. Дуже популярна практика – транслювати Python код у машинний код, аби виконувати його беспосередньо на ядрі операційцної системи, або навіть транслювати у байт-код іншої мови програмування аби запускати його на сторонній віртуальній машині. Так, наприклад, Spark API має Python інтерфейс, але усі пітонівські вихідні фали транслюються у Java байт-код та виконується на віртуальній машині Java. Еталонна реалізація Python – це інтерпретатор CPython, який підтримує найбільш часто використовувані платформи. Він розповсюджується під безкоштовною ліцензією програмного забезпечення Python Software Foundation, що дозволяє використовувати його без обмежень у будь-якому додатку, включаючи власні. Існує реалізація інтерпретатора для JVM з можливістю компіляції, а також інших незалежних реалізацій. Проект PyPy використовує компіляцію Just In Time, що значно збільшує швидкість програм Python.

4.2 Опис програмного забезпечення для тестування та демонстрації роботи методу токенізації і детокенізації великих табличних даних без Hadoop кластеру

Для демонстрації роботи нового методу токенізації і детокенізації великих табличних даних без Hadoop кластеру потребується розробка специфічних SQL-запитів для Big Query. Для перевірки працездатності методу було створено два типи SQL-скриптів:

- імпорт симетричних ключів шифрування (Рис 4.4);
- використання імпортованих ключів шифрування для перетворення даних (Рис. 4.5).

Big Query використовує SQL для керування усіма операціями перетворення. SQL – це перш за все декларативна мова програмування, що використовується для створення, зміни та управління даними у реляційній базі даних, керованій відповідною системою управління базами даних.

SQL – це, перш за все, інформаційно-логічна мова, призначена для опису, зміни та отримання даних, що зберігаються у реляційних базах даних [16]. SQL вважається мовою програмування, в загальному випадку, тобто без ряду сучасних розширень, він не є Тюрінг-повним, але в той же час існують мовні стандарти, що передбачають можливість його процедурного розширення.

В нашому випадку BigQuery підтримує наступні види SQL діалектів:

- Legacy SQL;
- Standart SQL.

Розглянемо кожен із них більш детально.

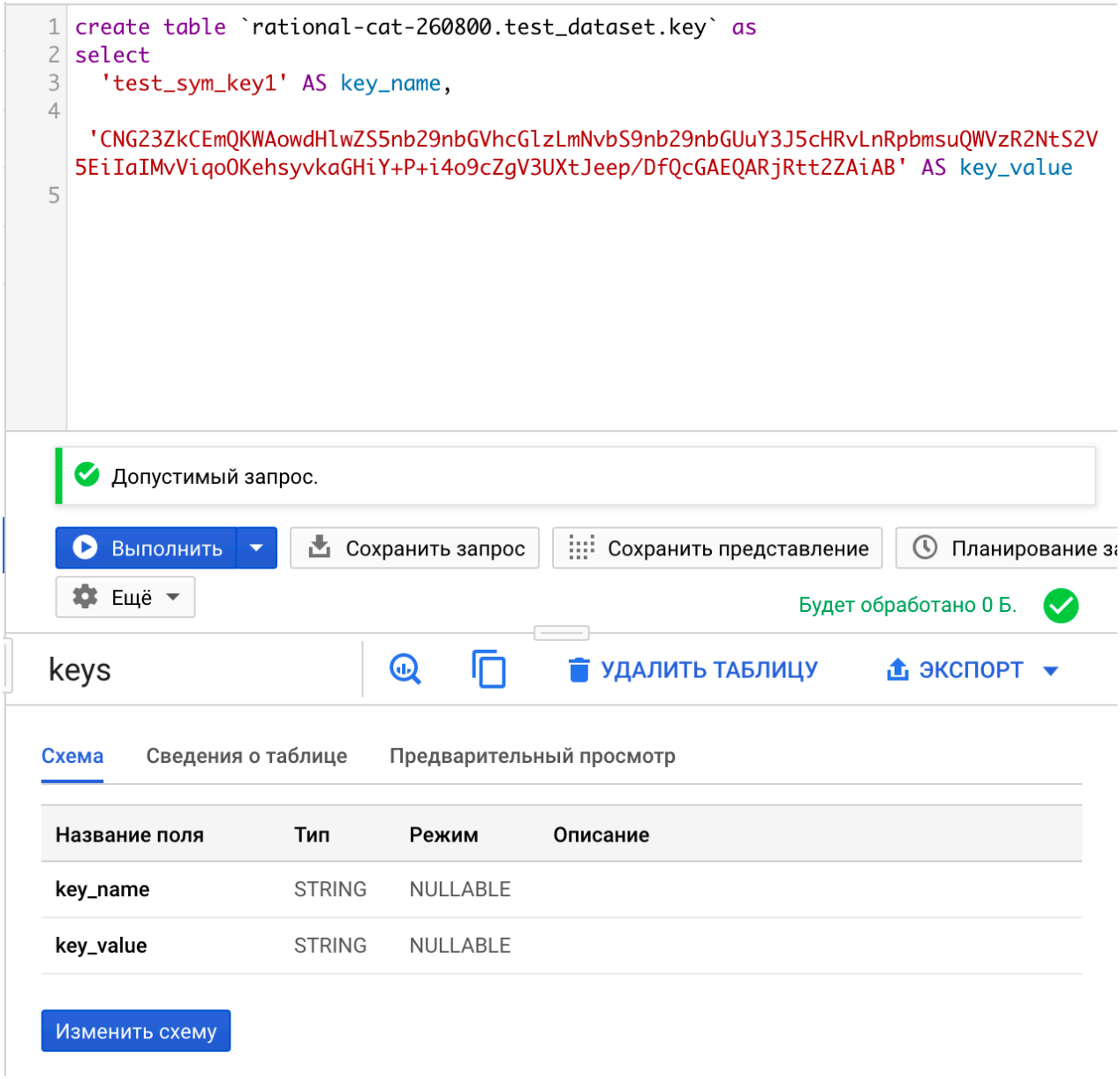

Рисунок 4.4 – Імпорт симетричних ключів шифрування

Legacy SQL – це SQL-діалект, зо був основним для роботи з Big Query до 2016 року. До цього часу у сервісу була власна версія мови структурованих запитів – BigQuery SQL, яка зараз називається Legacy. На перший погляд, між Legacy і Standard SQL немає великої різниці: трохи по-іншому пишуться назви таблиць, у стандарту трохи жорсткіші вимоги до граматики, наприклад, не можна ставити кому перед FROM, і більше типів даних. Але якщо придивитися, за невеликими відмінностями стоять зміни синтаксису, які дають маркетологам багато переваг.

```
with keyv as (
    select key_value as token
    from `rational-cat-260800.test_dataset.keys`
    where key_name = 'test_sym_key1'
),
encrypted as (
select
    visitId,
    channelGrouping,
    AEAD.ENCRYPT((select * from keyv), channelGrouping, cast(visitId as string)) as encrypted_channelGrouping
from `bigquery-public-data.google_analytics_sample.ga_sessions_20170801` limit 10
)
select
    *,
    AEAD.DECRYPT_STRING((select * from keyv), encrypted_channelGrouping, cast(visitId as string)) as decrypted_channelGrouping
from encrypted
```

Допустимый запрос.

Выполнить | Сохранить запрос | Сохранить представление | Планирование запроса | Ещё | Будет обработано 52 КБ.

Результаты запроса СОХРАНИТЬ РЕЗУЛЬТАТЫ ОТКРЫТЬ В СТУДИИ ДАННЫХ

Запрос выполнен за 0,9 сек. (обработано 52 КБ)

Сведения о задании | Результаты | Данные в формате JSON | Сведения о выполнении

Строка	visitId	channelGrouping	encrypted_channelGrouping	decrypted_channelGrouping
1	1501583974	Organic Search	AeF/2ai3iT4WD6Y0xFaq1dXuhcliXJozPV75nwk+lD6Af7sIV0HqhpCndNDhknw=	Organic Search
2	1501616585	Organic Search	AeF/2ahdwqd/0wo9JO0RQA9Io9dBUbKDlCJn3/qonLznAuSJXg0ojXdkZN4PgBA=	Organic Search
3	1501583344	Organic Search	AeF/2ahWnucu11r61HX2fVY4h2eY/YPRmNmFE6ZbS3qOBm/x5sn0wHYKODKZq98=	Organic Search
4	1501573386	Direct	AeF/2ajRD3eteQCfvhx75HpZ0SG6sAVXwwWMhDIzFRZkNM7Y6bLK	Direct
5	1501651467	Organic Search	AeF/2agcKuSKWHGSnNS3jVraRhDShvY9UIs1ndFEhrsdD+UgQnHVCdDvsnAKDwY=	Organic Search
6	1501611552	Referral	AeF/2ahYu7pXBvFxSFgVcUvVnoYVCsMAYofLueLUYXV7xBhIIrsrqi8=	Referral
7	1501600400	Organic Search	AeF/2agrqR+yY6vfgQyXpVEH4EU+azdrl5vsY+YyMO1SuBEetngRU+a0GdSQXC0=	Organic Search
8	1501640178	Organic Search	AeF/2aimlOTvpGuw4DbI13fZgIE2F6/WfRhP2xzrxvsLF1Qrb4SAwbK4oo+iH0M=	Organic Search
9	1501585492	Referral	AeF/2ahPrQznbyub57CXQb45qD26fSU4A1dClGwzTewcoBaPdT/XzJo=	Referral
10	1501635646	Organic Search	AeF/2ahRv5mF71zTgAxJCj62f7OIWR5CBIbbCoxgbpP9RW4aTZktybTEUcwCcrE=	Organic Search

Рисунок 4.5 – Використання імпортованих ключів шифрування для трансформації даних

Standart SQL – це новий SQL-діалект, що максимально наближений до стандартного SQL. Підтримка цього діалекту з'явилась у 2016 році. Стандартний SQL підтримує нові типи даних – ARRAY і STRUCT, тобто масиви і вкладені поля. Це означає, що в BigQuery стало простіше працювати з таблицями, що завантажуються з файлів JSON та Avro, дані в яких часто містять багаторівневі вкладення та об'єкти. За допомогою Standard SQL можна звертатися з BigQuery безпосередньо до таблиць Google Bigtable, Google Cloud Storage, Google Drive і Google Sheets. Тобто замість того, щоб завантажувати в BigQuery таблицю цілком, ви можете одним єдиним запитом очистити дані, вибрати ті параметри, які вам потрібні, і завантажити їх у хмарне сховище.

У стандартному діалекті призначені для користувача функції можна писати на SQL або JavaScript, а в Legacy підтримується тільки JavaScript. В якості аргументів функції використовуються колонки, а значення, які вона приймає – це результати дій з колонками. На стандартному діалекті функції можна писати в тому ж віконці, що і запити.

Якщо вам потрібно використовувати формулу, якої немає в документації, вам допоможуть призначені для користувача функції (User Defined Functions) [17]. У нашій практиці це дуже рідкісний випадок, оскільки документація Standard SQL покриває майже всі завдання digital-аналітики. Так самой й у нашому випадку Standart SQL містить необхідні нам функції, а саме:

- AEAD.ENCRYPT;
- AEAD.DECRYPT_STRING;
- AEAD.DECRYPT_BYTES.

Саме тому для реалізації специфічних SQL-запитів був обраний діалект Standart SQL. Головним критерієм вибору діалекту стала наявність необхідних функцій для шифрування колонок.

Не менш важливим є питання генерації та імпорту симетричних ключів шифрування. У BigQuery існують два способи створення таблиці із ключами, а саме:

- генерація ключів засобами BigQuery за допомогою функції KEYS.NEW_KEYSET;

- імпорт існуючих ключів у таблицю через SQL-запит.

Для реалізації був обраний другий варіант, адже він максимально наближений до реальності у зв'язку з тим, що генераія кілючів третьою стороною виглядає більш надійою.

Для цього був створений Java-додаток (Рис. 4.6), що генерує AES ключ довжиною блока 256 байт алгоритмом Galois Counter Mode.

```
import java.security.SecureRandom;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;

public class AES_GCM_Example
{
    private static final int AES_KEY_SIZE = 256;
    private static final int GCM_IV_LENGTH = 12;

    public static void main(String[] args) throws Exception
    {
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(AES_KEY_SIZE);

        // Generate Key
        SecretKey key = keyGenerator.generateKey();
        byte[] IV = new byte[GCM_IV_LENGTH];
        SecureRandom random = new SecureRandom();
        random.nextBytes(IV);

        System.out.println("Generated key : " + key);
    }
}
```

Рисунок 4.6 – Java-додаток для генерації AES ключів

Galois Counter Mode – це широко використовуваний симетричний блок-шифрний режим роботи з високою ефективністю та продуктивністю. Це режим автентифікованого шифрування AEAD, що забезпечує як конфіденційність, так і автентифікацію переданих даних, гарантуючи їх цілісність.

Режим GCM визначений для блокових шифрів з розміром блоку 128 біт. Існує опція GCM під назвою GMAC, яка забезпечує лише автентифікацію даних. Він

може використовуватися як інкрементальний код автентифікації повідомлення. І GCM, і GMAC приймають як вхідний вектор ініціалізації довільної довжини. Алгоритм не обмежений патентами, що дозволяє його вільне використання.

Завдяки наявності коду аутентифікації, тобто вставки для імітації, цей автентифікований режим шифрування дозволяє одержувачу легко виявляти будь-які зміни у повідомленні, зашифровані та доповнені інформацією, що передається відкрито, перш ніж розшифрувати його, що значно покращує захист від спотворень та криптоатак.

На наступному етапі ключ, що був згенерований Java-додатком, завантажується до BigQuery таблиці з обмеженим доступом. У подальшому доступ на запис та читання для цієї таблиці обмежується тільки до сервісних акаунтів, з-під яких виконується Big Data додаток, що потребує цих клюів для виконання розшифрування колонки.

ВИСНОВКИ

У результаті виконання атестаційної роботи було проведено дослідження методів обробки великих даних без використання Hadoop кластеру.

У рамках атестаційної магістерської роботи була доведена актуальність дослідження, проаналізовані існуючі методи та технології обробки великих даних, обґрунтована мета розробки нових методів та впровадження нових технологій, були сформовані критерії ефективності, поставлена задача дослідження.

На основі цих досліджень був виконано впровадження нових технологій та розроблені нові методи хмарного шифрування та дешифрування файлів без Hadoop кластеру та токенізації і детокенізації великих табличних даних без Hadoop кластеру, а саме: описані етапи нового методу, а також розроблений алгоритм реалізації нового методу.

Було також проведене дослідження отриманих наукових результатів, а саме дослідження методів хмарного шифрування та дешифрування файлів без Hadoop кластеру та токенізації і детокенізації великих табличних даних без Hadoop кластеру, було проведено навантажувальне тестування нових методів та технологій для аналізовану ефективності цих методів на базі критеріїв ефективності.

Задля демонстрації нового методу та його практичного використання було розроблене демонстраційне програмне забезпечення для взаємодії з хмарними сервісами Google Cloud.

У майбутньому планується продовження досліджень на тему обробки великих даних, а саме способів вдосконалення методів, отриманих у результаті досліджень в рамках магістерської атестаційної роботи. За результатами дослідження можна зробити висновок що поставлена задача була успішно виконана, були впроваджені нові технології, нові методи на базі нових технологій довели свою ефективність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Методичні вказівки щодо розробки та оформлення магістерської атестаційної роботи за спеціальністю 122 – „Комп'ютерні науки” програма «Інформаційні управляючі Методичні вказівки щодо розробки та оформлення магістерської атестаційної роботи за спеціальністю 122 Комп’ютерні науки (освітня програма «Інформаційні управляючі системи та технології» освітньо-кваліфікаційного рівня «магістр» / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2019. – 24 с.

2. В.Саенко, А. Шилин. Методы и технологии повышения эффективности информационных Cloud систем с Big Data сегментами.\ Информационные системы и технологии ИСТ2019: материалы 8-ой Международ. науч.-техн. конф., Коблево, 9-15 сентября 2019 г.: тезисы докладов/[редкол.: А.Д. Тевяшев (отв. ред.)]. – Х.: ДРУКАРНЯ МАДРИД, 2019. – 3 стр

3. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA (2004), pp. 137-150

4. T. White, “Hadoop: The definitive guide”, 4th ed. Sebastopol, ORLY, 2015, ch. 2, pp. 19-43

5. M. Zaharia, B. Chambers, “Spark: The definitive guide: Big Data Processing Made Simple”, 1st ed. Sebastopol, ORLY, 2018, ch. 2, pp. 13-31

6. What is cloud computing [Online]. Available: https://aws.amazon.com/ru/what-is-cloud-computing/

7. Top cloud providers 2018: How AWS, Microsoft, Google, IBM, Oracle, Alibaba stack up [Online]. Available: https://www.zdnet.com/article/top-cloud-providers-2018-how-aws-microsoft-google-ibm-oracle-alibaba-stack-up/

8. Xen Project Software Overview [Online]. Available: https://wiki.xen.org/wiki/Xen_Project_Software_Overview

9. Cloud products [Online]. Available: https://aws.amazon.com/ru/products/

10. Products and Services [Online]. Available: https://cloud.google.com/products/

11. P. Karlton. “The Secure Sockets Layer (SSL) Protocol Version 3.0”, 1st ed. – RTFM, Inc., August 2011, № 1, p. 67

12. Big Data Protector | Protegrity [Online]. Available: https://www.protegrity.com/products/protegrity-protectors/big-data-protector/

13. https://cloud.google.com/bigquery/docs/reference/standard-sql/aead-encryption-concepts

14. Microsoft Corporation, “Performance Testing Guidance for Web Applications, 1st ed.”, Microsoft Press 2007, p. 17

15. David Beazley, “Python Essential Reference (4th edition)”, Addison-Wesley Professional 2009, p. 117

16. Alan Beaulieu, “Learning SQL, Second edition”, Sebastapol, CA, USA, O’Reilly 2009, p. 7

17. Standart SQL User-Defined Functions | BigQuery | Google Cloud [Online]. Available: https://cloud.google.com/bigquery/docs/reference/standard-sql/user-defined-functions