ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота

на тему: «**ДОСЛІДЖЕННЯ МОЖЛИВОСТІ СТВОРЕННЯ РОЗПОДІЛЕНОЇ ОБЧИСЛЮВАЛЬНОЇ ІНФРАСТРУКТУРИ НА ОСНОВІ ТЕХНОЛОГІЙ DATA ENGINEERING, BIG DATA ТА CLOUD COMPUTING ДЛЯ АВТОМАТИЗОВАНОЇ ОБРОБКА ТА ІНТЕГРАЦІЯ ДАНИХ»**

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Євгеній ГОРБАТЮК
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. _____

Євгеній ГОРБАТЮК
(Ім'я, ПРІЗВИЩЕ)

Керівник: к.т.н., доцент Микола Гніденко
Науковий ступінь, вчене звання *(Ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
Науковий ступінь, вчене звання *(Ім'я, ПРІЗВИЩЕ)*

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально–науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук
Ступінь вищої освіти − «Магістр»
Спеціальність підготовки − «122 Комп'ютерні науки»
Освітньо-професійна програма − «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
_____Віктор Вишнівський
"____" _____2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Горбатюк Євгеній Юрійович
(прізвище, ім'я, по батькові)

1.Тема кваліфікаційної роботи: «Дослідження можливості створення розподіленої обчислювальної інфраструктури на основі технологій Data Engineering, Big Data та Cloud Computing для автоматизованої обробка та інтеграція даних»
Керівник кваліфікаційної роботи __Микола Гніденко, к.т.н., доцент._____,
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 19.10.2023 № 145
2. Строк подання студентом роботи 20.12.2023 р.
3. Вихідні дані до роботи: Аспекти розподіленої інфраструктури в контексті обробки та інтеграції даних, GCP, AWS, Azure. Науково-технічна література з питань, пов'язаних з Data Engineering, Big Data, Cloud Computing.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд предметної області.
2. Дослідження технологій Data Engineering, Big Data та Cloud Computing.
3. Дослідження реалізації розподіленої обчислювальної інфраструктури.

5. Перелік ілюстративного матеріалу: *презентація*

1. Мета роботи, об`єкт та предмет дослідження
2. Опис актуальності теми
3. Огляд областей Data Engineering, Big Data, Cloud Computing
4. Розподільні обчислення
5. Стек даних (Data Stack)
6. Підходи до обробки даних ETL та ELT
7. Підход ETL
8. Підход ELT
9. OLAP та OLTP системи обробки даних
10. Етапи обробки даних
11. Технології обробки даних
12. Реалізація хмарної та локальної системи обробки даних
13. Реалізація розподіленої обчислювальної інфраструктури локально
14. Реалізація розподіленої обчислювальної інфраструктури хмарно AWS
15. Реалізація розподіленої обчислювальної інфраструктури хмарно Azure
16. Реалізація розподіленої обчислювальної інфраструктури хмарно GCP
17. Проведення аналітики між локальним та хмарним базуванням
18. Висновки

6. Дата видачі завдання 19.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.10-27.10.23	вик.
2	Сучасні рішення щодо застосування Data Engineering, Big Data, Cloud Computing	28.10-10.11.23	вик.
3	Варіанти реалізації системи обробки даних	11.11-30.11.23	вик.
4	Вибір належного релевантної архітектури	01.12-16.12.23	вик.
5	Вступ, висновки, реферат	17.12-18.12.23	вик.
6	Розробка демонстраційних креслень	19.12-20.12.23	вик.
7	Розробка методологічної бази знань	21.12-26.12.23	вик.

Здобувач вищої освіти _____ Євгеній Горбатюк
(підпис) (Ім'я, ПРІЗВИЩЕ)

Керівник
кваліфікаційної роботи _____ Микола_Гніденко
(підпис) (Ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 92 стор., 17 рис., 6 додатків, 57 джерел.

Мета роботи – дослідити та проаналізувати архітектурні рішення та засоби для аналітики великих даних у розподіленій обчислювальній інфраструктурі з використанням технологій Data Engineering, Big Data та Cloud Computing.

Об`єкт дослідження – аналіз методів та засобів обробки даних в контексті систем, що використовують локальні та хмарні середовищі.

Предмет дослідження – інструментальні, технологічні засоби та архітектури для процесів аналізу та обробки великих даних.

Короткий зміст роботи:

Робота присвячена дослідженню можливостей створення розподіленої обчислювальної інфраструктури на основі технологій Data Engineering, Big Data та Cloud Computing для автоматизованої обробки та інтеграції даних.

Перший розділ надає огляд сучасного стану предметної області, визначає ключові аспекти розподільної обробки даних. Другий розділ досліджує технології Data Engineering, Big Data та Cloud Computing, аналізує можливості та обмеження в контексті створення інфраструктури. У третьому розділі проводиться дослідження реалізації локальної та хмарної обчислювальної системи на основі набору технологій стеку обробки даних.

Отримані результати можуть бути використані у розробці ефективних архітектур обробки та аналітики великих даних, а також враховані при проектуванні та впровадженні розподіленої обчислювальної інфраструктури.

КЛЮЧОВІ СЛОВА: Data Engineering, Big Data, Cloud Computing, розподілена обчислювальна інфраструктура, аналітика великих даних.

ABSTRACT

Text part of the master's thesis: 92 pages, 17 figures, 6 appendices, 57 sources.

The purpose of the work – the aim of the work is to investigate and analyze architectural solutions and tools for big data analytics in distributed computing infrastructure using Data Engineering, Big Data, and Cloud Computing technologies.

Object of research – the research object is the analysis of methods and tools for processing large volumes of data in the context of infrastructure systems used at the local and cloud levels.

Subject of research – the research subject is the instrumental, technological tools, and architectures for the processes of analysis and processing of big data.

Summary of the work:

The paper is devoted to exploring the possibilities of creating a distributed computing infrastructure based on Data Engineering, Big Data, and Cloud Computing technologies for automated data processing and integration.

The first section provides an overview of the current state of the subject area and identifies key aspects of distributed data processing. The second section explores Data Engineering, Big Data, and Cloud Computing technologies, analyzing their possibilities and limitations in the context of infrastructure creation. The third section investigates the implementation of local and cloud computing systems based on a set of data processing technologies.

The obtained results can be utilized in the development of effective architectures for big data processing and analytics, as well as considered in the design and implementation of distributed computing infrastructure.

KEYWORDS: Data Engineering, Big Data, Cloud Computing, distributed computing infrastructure, big data analytics.

ЗМІСТ

Стор.

ВСТУП

У сучасному світі обробка, аналіз та інтеграція даних стали критично важливими завданнями для підприємств, наукових досліджень і багатьох інших сфер. Інформація, яка оточує нас, росте з неймовірною швидкістю, створюючи великі обсяги даних, які вимагають спеціалізованих підходів для їх обробки та аналізу. Від коректності та ефективності цих процесів залежить успіх організацій та можливість отримання нових знань.

Великі обсяги і різноманітність даних вимагають ефективних технологій для їх обробки, інтеграції та аналізу. Однак це завдання стає все більш складним і вимагає нових підходів. У цьому контексті технології Data Engineering, Big Data та Cloud Computing стають ключовими компонентами для автоматизованої обробки та інтеграції даних.

Взаємозв'язок цих областей полягає в тому, що Data Engineering забезпечує створення інфраструктури для збирання та зберігання даних, Big Data дозволяє аналізувати ці дані великого обсягу, а Cloud Computing надає масштабні обчислювальні ресурси для виконання аналізу та обробки даних. Всі ці технології разом відкривають нові можливості для автоматизованої обробки та інтеграції даних.

Дана магістерська робота присвячена аналізу можливостей створення розподіленої обчислювальної інфраструктури на основі вищезазначених технологій та їх впливу на оптимізацію процесів обробки та інтеграції даних у сучасних умовах. Дослідження в цій області спрямоване на розв'язання викликів, пов'язаних з обробкою даних великого обсягу та забезпеченням доступу до них у режимі реального часу.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд понять та технологій

Data Engineering. Data Engineering є важливою складовою сучасного інформаційного ландшафту. Він включає в себе процеси збору, обробки, зберігання та передачі даних. Data Engineering допомагає створити інфраструктуру для зберігання великих обсягів даних та забезпечує їх доступність для аналітики та досліджень. Важливою частиною Data Engineering є розробка потужних інструментів для автоматизації цих процесів.

Big Data. Big Data відноситься до великих обсягів даних, які не можуть бути ефективно оброблені традиційними методами обробки даних. Big Data включає структуровані та неструктуровані дані, і їх аналіз може надати цінний інсайт для прийняття рішень. Використання Big Data вимагає спеціалізованих інструментів та технологій, які дозволяють ефективно аналізувати великі масиви даних.

Cloud Computing. Cloud Computing полягає в наданні доступу до обчислювальних ресурсів та послуг через Інтернет. Ця технологія дозволяє підприємствам збільшити масштаби обчислення, зменшити витрати на IT-інфраструктуру та покращити доступність послуг. Використання хмарних ресурсів може значно полегшити впровадження та масштабування проектів Data Engineering та Big Data.

Розподільні обчислення – це спосіб об’єднань декількох комп’ютерів для рішення загальних проблем. Це перетворює комп’ютерну мережу в потужний єдиний комп’ютер, який надає широкомасштабні ресурси для вирішення складних задач.

Розподілені обчислення можуть шифрувати великі обсяги даних, вирішувати фізичні та хімічні рівняння з багатьма змінними та виводити високоякісну тривимірну анімацію. Розподілені системи, розподілене програмування та розподілені алгоритми – це ще кілька термінів , що відносяться

до розподільних обчислень.

Переваги розподілених обчислень:

- *Можливість масштабування.* Розподільні системи можуть зростати зі зростанням робочих навантажень та вимог. При необхідності можливо додавати у обчислювальну мережу нові вузли, тобто додавати обчислювальні пристрої.
- *Доступність.* Розподілена обчислювальна система не дасть збою, якщо дасть збій один з комп'ютерів. Ця система характеризується стійкістю до відмови, оскільки вона може продовжувати працювати, навіть якщо дасть збій окремий комп'ютер.
- *Узгодженість.* Комп'ютери у розподільній системі спільно використовують інформацію та зберігають дублікати даних, також система автоматично керує цілісністю даних на всіх комп'ютерах. Тому забезпечуються перевага у вигляді стійкості до відмови без порушення цілісності даних.
- *Прозорість.* Розподілені обчислювальні системи забезпечують логічний поділ між користувачем та фізичними пристроями. Можливо взаємодіяти з системою, як окремим комп'ютером, не дбаючи про налаштування та конфігурацію окремих комп'ютерів. Можливо використовувати різні апаратні та програмні забезпечення, проміжне програмне забезпечення та операційні системи, які спільно працюють над тим, щоб забезпечити безперебійну роботу системи.
- *Ефективність.* Розподілені системи забезпечують підвищену продуктивність та оптимальне використання ресурсів базового програмного забезпечення. В результаті можливо керувати будь-яким робочим навантаженням, не дбаючи про збої екосистеми через сплески обсягів або недостатнє використання дорогого обладнання.

Приклади використання розподілених обчислень:

Розподілені архітектури сьогодні використовуються всюди. Мобільні та інтернет-програми є прикладами розподілених обчислень, оскільки кілька комп'ютерів спільно працюють над тим, щоб надати правильну інформацію. Однак при вертикальному масштабуванні розподілених систем нагору вони

можуть вирішувати складніші проблеми. Розглянемо, як різні галузі використовують високопродуктивні розподілені програми.

Охорона здоров'я та медико-біологічні розробки. У галузі охорони здоров'я та медико-біологічних досліджень розподілені обчислення використовуються для моделювання та імітації складних біологічних даних. Аналіз зображень, дослідження медичних препаратів та аналіз генетичної структури стали швидшими завдяки розподіленим системам. Нижче наведено кілька прикладів.

- прискорення проектування медичних препаратів на основі структури за допомогою візуалізації молекулярних моделей в трьох вимірах.
- зменшення строків обробки геномних даних для ранньої виявлення раку, муковісцидозу та хвороби Альцгеймера.
- розробка інтелектуальних систем, які допомагають лікарям ставити діагнози пацієнтам, обробляючи великі обсяги складних зображень, таких як МРТ, рентгенограми та КТ.

Інженерні дослідження. За допомогою розподілених систем інженери можуть імітувати складні фізичні та механічні концепції. Вони використовують ці рішення для покращення дизайну продуктів, створення складних структур та проектування швидших транспортних засобів. Нижче наведено кілька прикладів.

- обчислювальна гідродинаміка досліджує поведінку рідин та реалізує ці концепції у конструкціях літальних апаратів та гоночних автомобілів.
- для машинного моделювання потрібні інструменти імітації з розширеними обчислювальними ресурсами, щоб тестувати новинки у галузі виробничого обладнання, електроніки та споживчих товарів.

Фінансові послуги. Компанії у галузі фінансових послуг використовують розподілені системи для виконання високошвидкісних економічних імітацій щодо оцінки ризиків портфелів, прогнозування ринкових тенденцій та підтримки прийняття фінансових рішень. Вони можуть створювати інтернет-додатки, які мають переваги розподілених систем для досягнення наступних цілей:

- надання персоналізованих страхових внесків за низьких витрат.

- використання розподілених баз даних для безпечної підтримки дуже великого обсягу фінансових транзакцій.
- автентифікація користувачів та захист клієнтів від шахрайства.

Енергетика та екологія. Компаніям з енергетичної галузі необхідно аналізувати великі обсяги даних для покращення операцій та переходу до сталого розвитку та безпечних для клімату рішень. Вони використовують розподілені системи для аналізу великих потоків даних, що надходять від багатьох датчиків та інших інтелектуальних пристроїв. Нижче наведено деякі завдання, які вони можуть виконувати:

- потокова передача та консолідація сейсмічних даних для структурного проектування електростанцій;
- моніторинг нафтових свердловин як реального часу для активного управління ризиками.

Типи розподіленої обчислювальної архітектури. При розподілених обчисленнях ви проектуєте програми, які можуть виконуватися на кількох комп'ютерах, а не лише на одному. Це досягається за рахунок такого проектування програмного забезпечення, при якому різні комп'ютери виконують різні функції та зв'язуються між собою для одержання остаточного рішення. Існує чотири основні типи розподіленої архітектури:

1. Клієнт-серверна архітектура

Клієнт-серверна архітектура - це найпоширеніший спосіб організації розподіленої системи, зображено на Рисунку 1.1. Комп'ютери виконують дві ролі: клієнти та сервери.

Клієнти. Кількість інформації та обчислювальні можливості клієнтів обмежені. При цьому вони надсилають запити серверам, які управляють здебільшого даних та інших ресурсів. Запити можливо надсилати до клієнтів, і ті будуть зв'язуватись із серверами від клієнтського імені.

Сервери. Сервери синхронізують ресурси та керують доступом до них. Вони відповідають на запити клієнтів, надсилаючи дані або інформацію про стан.

Зазвичай, один сервер може обробляти запити від декількох комп'ютерів.

Переваги та обмеження. Перевагами клієнт-серверної архітектури є безпека та простота поточного обслуговування. Потрібно приділяти основну увагу лише захисту серверів. Крім того, будь-які зміни в системах баз даних потребують змін лише на сервері.

Обмеженням клієнт-серверної архітектури є те, що сервери можуть стати вузьким місцем у зв'язку, особливо коли кілька комп'ютерів надсилають запити одночасно.

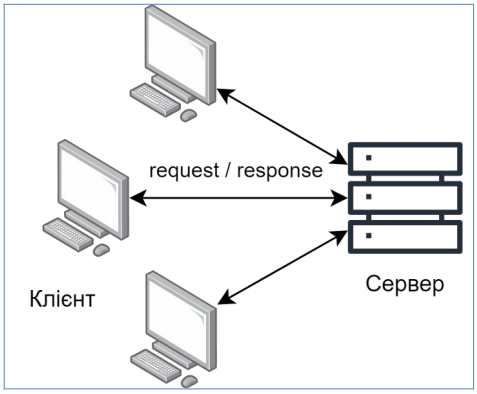

Рисунок 1.1 - Проста схема клієнт-серверної архітектури

2. Трирівнева архітектура

У трирівневих розподілених системах клієнтські комп'ютери залишаються першому рівні доступу, відповідно Рисунку 1.2. З іншого боку, серверні комп'ютери поділяються на дві категорії.

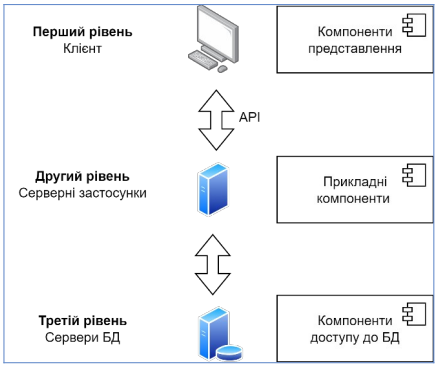

Рисунок 1.2 - Схема трирівневої архітектури

Сервери програм. Сервери програм – це середній рівень зв'язку. Вони містять логіку додатків чи основні функції, до виконання яких призначено розподілена система.

Сервери баз даних – це третій рівень, у якому здійснюється зберігання даних та управління ними. Вони відповідають отримання даних та його цілісність.

Завдяки розподілу відповідальності на сервері трирівнева розподілена система скорочує кількість вузьких місць у зв'язку та підвищує продуктивність розподілених обчислень, що абстрактно показано на Рисунку 1.3.

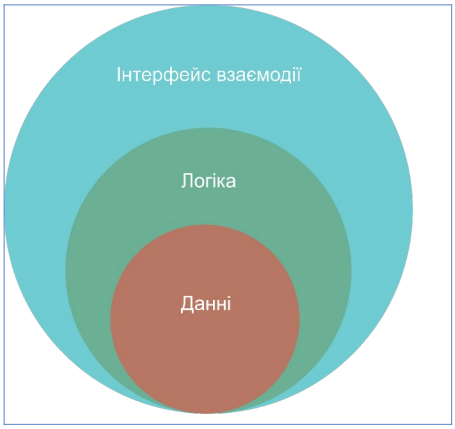

Рисунок 1.3 - Діаграма логіки трирівневої архітектури

3. N-рівнева архітектура

N-рівневі моделі складаються з декількох різних клієнт-серверних систем, які взаємодіють між собою для вирішення однієї і тієї ж проблеми. Більшість сучасних розподілених систем використовують n-рівневу архітектуру, в якій різні корпоративні програми спільно працюють за лаштунками як одна система.

4. Пірингова архітектура

Пірингові розподілені системи покладають попри всі комп'ютери у мережі однакові обов'язки. Поділ на клієнтські та серверні комп'ютери відсутній, і будь-який комп'ютер може виконувати всі функції, як вказано на Рисунку 1.4. Пірингова архітектура стала популярною у сфері спільного використання контенту, потокової передачі файлів та мереж блокчейн.

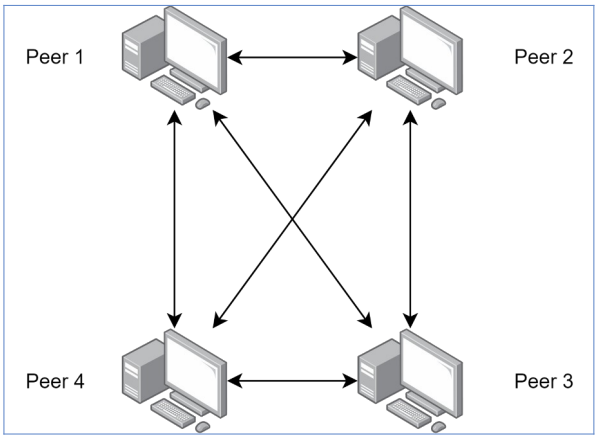

Рисунок 1.4 - Схема піринговой мережі (P2P)

Робота розподілених обчислень. У розподілених обчисленнях комп'ютери взаємодіють, надсилаючи один одному повідомлення в архітектурі розподіленої системи. Протоколи зв'язку або правила створюють залежності між компонентами розподіленої системи. Ця взаємодія називається взаємозалежністю. Існують два основних типи взаємозалежності.

Слабка взаємозалежність. При слабкій взаємозалежності компоненти мають слабке зв'язок, і зміни в одному з них не впливають на інший. Наприклад, між клієнтом і сервером може бути слабка взаємозалежність за часом. Повідомлення від клієнта додаються до черги на сервері, і клієнт може продовжувати виконувати інші функції, поки сервер не відповість на його повідомлення.

Сильна взаємозалежність. У високопродуктивних розподілених системах часто використовується сильна взаємозалежність. Швидкі локальні мережі, як правило, охоплюють кілька комп'ютерів, які утворюють кластер. У кластерних обчисленнях всі комп'ютери виконують однакове завдання. Центральні системи управління, відомі як кластерне посередництво, керують виконанням завдань, планують їх і координують комунікацію між різними комп'ютерами.

Паралельні обчислення. Паралельні обчислення – це тип обчислень, у якому один або кілька комп'ютерів у мережі проводять багато обчислень або виконують

багато процесів одночасно. Незважаючи на те, що терміни паралельні обчислення та розподілені обчислення часто використовуються як взаємозамінні, між ними є деякі відмінності.

Паралельні обчислення та розподілені обчислення. Паралельні обчислення – це форма розподілених обчислень із особливо сильною взаємозалежністю. При паралельній обробці всі процесори мають доступом до спільної пам'яті обмінюватись інформацією друг з одним. З іншого боку, при розподілених обчисленнях кожен процесор користується власною пам'яттю (розподіленою пам'яттю). Процесори використовують для обміну інформацією передачу повідомлень.

Мережеві розподілені обчислення. У мережевих розподілених обчисленнях географічно розподілені комп'ютерні мережі працюють над виконанням загальних завдань. Відмінною рисою розподілених мереж є те, що їх можна сформувати з обчислювальних ресурсів, що належать кільком особам чи організаціям.

Мережеві розподілені обчислення та розподілені обчислення. Мережеві розподілені обчислення – це розподілені обчислення з високим ступенем масштабування, у яких акцент зроблено на продуктивності та координуванні кількох мереж. На внутрішньому рівні кожна розподілена мережа працює як обчислювальна система із сильною взаємозалежністю. Однак на зовнішньому рівні взаємозалежність між мережами слабша. Кожна розподілена мережа виконує окремі функції та повідомляє результати іншим мережам.

Стек даних (Data Stack) відноситься до набору технологій та інструментів, які організації використовують для збору, зберігання, обробки, аналізу та керування своїми даними. Стек даних можна розглядати як «інфраструктуру», яка дозволяє організаціям перетворювати необроблені дані на практичні ідеї.

Стек даних (Data Stack) зазвичай включає технології та інструменти для керування даними, сховища даних, управління даними, аналітики даних, розробки даних, науки про дані, безпеки даних і бізнес-аналітики. Ці компоненти можуть включати різноманітне програмне забезпечення, платформи та технології, такі як:

- *Ведення даними (Data management):* бази даних, озера даних (data lakes), конвеєри даних (data pipelines), інструменти інтеграції даних.
- *Сховища даних(Data warehousing):* платформи сховищ даних, інструменти ETL (extract, transform, load) та ELT, стовпчасті бази даних, вітрини даних.
- *Управління даними (Data governance):* інструменти якості даних, каталоги даних, інструменти походження даних.
- *Аналітика даних (Data analytics):* засоби візуалізації даних, програмне забезпечення для інтелектуального аналізу даних, програмне забезпечення для прогнозної аналітики, платформи машинного навчання.
- *Інженерія даних (Data engineering):* інструменти інтеграції даних, конвеєри даних, інфраструктури обробки даних, платформи сховищ даних.
- *Наука про дані (Data science):* бібліотеки машинного навчання, бібліотеки обробки природної мови, бібліотеки візуалізації даних.
- *Безпека даних (Data security):* інструменти шифрування даних, інструменти маскування даних, засоби контролю доступу до даних, інструменти моніторингу та аудиту даних.
- *Бізнес-аналітика (Business intelligence):* платформи бізнес-аналітики, засоби візуалізації даних, програмне забезпечення інтелектуального аналізу даних.

Класифікація стейків даних (Data Stack). Для зазначених компонентів також можуть бути певні стеки, які наведені в Таблиці 1.1.

Таблиця 1.1 - Типи стеків даних (Data Stack)

Основний тип	**Опис**	**Звичайні компоненти**
Big Data Stack	Технології та інструменти, що використовуються для управління, зберігання та аналізу великих обсягів даних	Hadoop, Spark, NoSQL бази даних, інструменти візуалізації даних та аналітики

Продовження Таблиці 1.1 - Типи стеків даних (Data Stack)

Cloud Data Stack	Технології та інструменти, що використовуються для управління, зберігання та аналізу даних у хмарі	Хмарні служби для зберігання та обробки даних, інструменти візуалізації даних та аналітики, що можуть бути запущені в хмарі
Data Governance Stack	Технології та інструменти, що використовуються для забезпечення точності, безпеки та відповідності даних	Інструменти якості даних, каталоги даних, інструменти лінійної передачі даних, контроль доступу до даних, інструменти моніторингу та аудиту даних
Data Analytics Stack	Технології та інструменти, що використовуються для отримання інсайтів з даних	Інструменти візуалізації даних, програмне забезпечення для видобування даних, програмне забезпечення для прогнозування, платформи для машинного навчання
Data Warehousing Stack	Технології та інструменти, що використовуються для управління та аналізу великих обсягів даних	Платформи для дітей даних, інструменти ETL, колоночні бази даних, даних магазини
Data Engineering Stack	Технології та інструменти, що використовуються для збору, зберігання та обробки даних у великих масштабах	Інструменти інтеграції даних, потоки даних, фреймворки обробки даних, платформи для дітей даних
Data Science Stack	Технології та інструменти, що використовуються в сфері науки про дані	Бібліотеки машинного навчання, бібліотеки обробки природної мови, бібліотеки візуалізації даних
Data Security Stack	Технології та інструменти, що використовуються для захисту даних від кіберзагроз та забезпечення відповідності галузевим регуляціям	Інструменти шифрування даних, інструменти маскування даних, контроль доступу до даних, інструменти моніторингу та аудиту даних
Business Intelligence Stack	Технології та інструменти, що використовуються для перетворення даних в інсайти та покращення бізнес-рішень	Платформи бізнес-аналітики, інструменти візуалізації даних, програмне забезпечення для видобування даних

Порівняння спадкових (legacy) та сучасних (modern) стеків даних. Застарілі стеки даних стосуються старих систем або технологій, які використовувалися для

керування даними в минулому. Ці системи можуть базуватися на застарілих технологіях або архітектурі та не в змозі працювати з обсягом, різноманітністю та швидкістю даних, які генерують і обробляють сучасні організації. Їм також може бракувати масштабованості, гнучкості та безпеки, необхідних для задоволення потреб сучасного бізнесу.

Сучасні стеки даних, з іншого боку, побудовані з використанням нових технологій і архітектури, які розроблені для роботи з масштабом і складністю сучасних даних. Вони часто використовують хмарні сервіси, розподілені системи та технології з відкритим кодом, щоб забезпечити масштабованість, гнучкість і економічну ефективність. Сучасні стеки даних також розроблені, щоб бути більш безпечними та підтримувати обробку та аналітику даних у реальному часі.

Сучасний стек даних також використовує технології з відкритим кодом, які часто дозволяють створювати та налаштовувати стек відповідно до потреб, включаючи інтеграцію даних, обробку даних, зберігання даних, керування даними, виявлення даних, візуалізацію даних і платформи машинного навчання. Вони також дають змогу приймати рішення на основі даних і отримувати інформацію. В Таблиці 1.2 порівняння застарілих та сучасних стеків даних.

Таблиця 1.2 - Порівняння старих та сучасних Data Stack

Ознака	**Старі стеки даних**	**Сучасні стеки даних**
Архітектура	Монолітна	Розподілена, хмарно-нативна
Масштабованість	Обмежена	Висока
Обробка даних	Партійна (батчева)	В реальному часі, на основі потоків
Зберігання даних	Реляційні бази даних	Багатомодельні бази даних, сховище даних (data lake)
Управління даними	ad hoc, ручний	Автоматизований, політико-орієнтований

Продовження Таблиці 1.2 - Порівняння старих та сучасних Data Stack

Інтеграція даних	Користувацькі, ручні	Автоматизовані, на основі API
Пошук та візуалізація даних	Базовий, статичний	Інтерактивний, динамічний
Безпека	Базовий, реактивний	Високий, проактивний
Гнучкість	Обмежена	Висока
Дані, наука та машинне навчання	Базовий	Розширений

Важливо зауважити, що різниця між застарілими та сучасними стеками даних не завжди є чіткою, і межа між ними може відрізнятися залежно від організації. Деякі організації можуть модернізувати частини свого стеку даних, зберігаючи застарілі системи в інших частинах, а інші можуть перебувати в процесі переходу від застарілого стеку даних до сучасного.

Переваги сучасних стеків даних. Сучасні стеки даних, створені з використанням новітніх технологій, мають кілька переваг перед традиційними стеками даних. 10 переваг сучасних стеків даних включають:

1. Хмарні (Cloud-native): сучасні стеки даних розроблені для хмарних обчислень, що означає, що вони створені для роботи на платформах хмарних обчислень. Це забезпечує легку масштабованість, а також заощаджує кошти, сплачуючи лише за ті ресурси, які потрібні.

2. Автоматизація: багато сучасних стеків даних включають засоби автоматизації, які можуть допомогти оптимізувати обробку даних і полегшити керування великими обсягами даних.

3. Обробка даних у режимі реального часу (Real-time): сучасні стеки даних часто включають технології, спеціально розроблені для обробки даних у режимі реального часу, такі як потокові платформи та аналітичні бази даних у режимі реального часу, які дозволяють виконувати швидший і точніший аналіз даних.

4. Великі дані Big Data: сучасні стеки даних призначені для обробки

великих даних, що є терміном для наборів даних, які є настільки великими або складними, що традиційні інструменти обробки даних не підходять.

5. Мультиструктуровані дані: сучасні стеки даних створено для обробки різноманітних типів даних, у тому числі структурованих, напів структурованих і неструктурованих даних, що дає змогу зберігати й аналізувати дані з широкого діапазону джерел.

6. Простота використання: інтерфейс користувача, абстракція конвеєра даних та інші набори інструментів у сучасних стеках даних розроблені так, щоб бути зручними для користувача, що полегшує роботу з ними для аналітиків даних, інженерів і вчених. Масштабованість: сучасні стеки даних призначені для роботи з великими обсягами даних, і їх можна легко масштабувати вгору або вниз відповідно до мінливих потреб бізнесу. Це часто досягається за рахунок використання розподілених систем і хмарних сервісів.

7. Багатомодульне зберігання даних: сучасні стеки даних підтримують різні типи моделей зберігання даних, такі як реляційні бази даних, бази даних документів, бази даних графів, бази даних ключ-значення, бази даних об’єктів. Це дозволяє організаціям вибирати найкращий варіант зберігання своїх даних залежно від конкретного випадку використання.

8. Автоматизоване керування даними: сучасні стеки даних забезпечують автоматизовані можливості керування даними, такі як походження даних, каталогізація даних, походження даних, керування метаданими, що дозволяє організаціям ефективно керувати та контролювати свої дані.

9. Розширена аналітика даних: сучасні стеки даних надають розширені інструменти та методи аналітики, такі як машинне навчання та обробка природної мови, які дозволяють організаціям отримувати цінну інформацію зі своїх даних.

10. Розширена безпека: сучасні стеки даних мають вбудовані розширені функції безпеки, такі як шифрування даних, автентифікація, контроль доступу, виявлення загроз і керування інцидентами. Це допомагає організаціям захистити свої дані від несанкціонованого доступу та злому.

Компоненти сучасного стеку даних. Шість основних компонентів стека даних: інтеграція даних (Data Integration), зберігання даних (Data Storage), обробка даних (Data Processing), аналіз даних (Data Analysis), візуалізація даних (Data Visualization), управління даними (Data Governance). У Таблиці 1.3 описано та представлено приклади для рівнів стеку.

Таблиця 1.3 - Рівні стеків даних (Data Stack)

Рівень стеку даних	**Опис**	**Приклади**
Інтеграція даних (Data Integration)	Технології та інструменти, що використовуються для збору та інтеграції даних з різних джерел	Daton, AWS Kinesis, Logstash
Зберігання даних (Data Storage)	Бази даних та інші системи зберігання, використовувані для зберігання даних у структурованому або неструктурованому форматі. Моделювання даних тісно пов'язане з цим рівнем, оскільки модель даних визначає структуру даних, яка зберігається в цих системах.	MySQL, PostgreSQL, MongoDB, Cassandra, AWS S3, Google Cloud Storage
Обробка даних (Data Processing)	Технології та інструменти, що використовуються для обробки та очищення даних	Apache Spark, Hadoop
Аналіз даних (Data Analysis)	Інструменти та технології, що використовуються для аналізу та вилучення інсайтів з даних	Платформи для машинного навчання, такі як TensorFlow і PyTorch, або мова програмування Python. SQL
Візуалізація даних (Data Visualization)	Інструменти та технології, що використовуються для відображення даних у легкорозумному форматі	Power BI, Excel, Google Data Studio
Управління даними (Data Governance)	Технології та інструменти, які допомагають організаціям керувати та регулювати їх дані	Collibra, Informatica, Alation

Рівень збору даних (Data Collection Layer). Це включає в себе технології та

інструменти, які використовуються для збору даних із різних джерел, таких як інструменти ELT, API, пристрої IoT, веб-скрейпінг(web scraping) і бази даних. У Таблиці 1.4 наведено технології та інструменти збору даних.

Таблиця 1.4 - Технології та інструменти збору даних

Метод збору даних	**Головні аспекти**	**Інструменти ELT/ETL**
Веб-скрапінг (Web scraping)	Автоматизований збір даних з веб-сайтів	BeautifulSoup, Scrapy, Parsehub
APIs	Програмний доступ до даних з зовнішніх систем	Daton, RapidAPI, Talend
Експорт з бази даних	Добуття даних з бази даних та їх експорт у певному форматі	MySQL, SQL Server Management Studio, Oracle SQL Developer
Файли Excel/CSV	Добуття даних з файлів електронних таблиць	Microsoft Excel, OpenOffice Calc, Google Sheets
Файли журналів	Добуття даних з файлів журналів, створених різними системами	Logstash, Flume, Fluentd
Дані з соціальних мереж	Добуття даних з платформ соціальних мереж (наприклад, твіти, повідомлення тощо)	Hootsuite Insights, Brandwatch, Crimson Hexagon

Рівень зберігання даних (Data Storage Layer). Це включає технології та інструменти, що використовуються для зберігання даних, як-от реляційні бази даних (наприклад, MySQL, PostgreSQL), нереляційні бази даних (наприклад, MongoDB, Cassandra), сховище даних (наприклад, Amazon Redshift, Google BigQuery) і хмарні рішення для зберігання даних (наприклад, Amazon S3). , Google Cloud Storage). У Таблиці 1.5 наведено технології та інструменти зберігання даних.

Таблиця 1.5 - Технології та інструменти зберігання даних

Варіант зберігання	**Переваги**	**Недоліки**
Реляційні бази даних (наприклад, MySQL, PostgreSQL)	Підтримка структурованих запитів за допомогою SQL, розроблені для забезпечення цілісності і послідовності даних.	Може бути менш продуктивним при масштабуванні і може вимагати більше складної настройки та обслуговування.
Нереляційні бази даних (наприклад, MongoDB, Cassandra)	Більш продуктивні при масштабуванні та можуть бути ефективнішими для певних випадків, таких як зберігання великих обсягів неструктурованих даних.	Позбавлені робустних можливостей запитів реляційних баз даних і можуть бути менш ефективними в забезпеченні цілісності та послідовності даних.
Сховище даних (наприклад, Amazon Redshift, Google BigQuery)	Розроблені для сховища даних та роботи бізнес-аналітики (BI), дозволяють зберігати та запитувати великі обсяги історичних даних і підтримують складні агреговані запити.	Дорожчі у відношенні до вартості ліцензування та обслуговування та можуть бути менш продуктивними з високими навантаженнями на запис.
Хмарне сховище (наприклад, Amazon S3, Google Cloud Storage)	Може бути високо масштабованим та дозволяє легкий доступ до даних з будь-якого місця.	Може бути дорожчим, ніж інші варіанти зберігання, та може вимагати більше складних розгляду з питань безпеки та відповідності.
Розподілені файлові системи (наприклад, HDFS, GlusterFS)	Висока доступність та реплікація даних, підтримка дуже великих файлів і каталогів, добре підходять для великих обсягів даних та пакетної обробки робочих навантажень.	Вимагають більше складної настройки та обслуговування і можуть не підтримувати доступ до даних в режимі реального часу або робочі навантаження з транзакціями.

Рівень обробки даних (Data Processing Layer). Це включає технології та інструменти, що використовуються для обробки та перетворення даних, наприклад Apache Hadoop і Apache Spark. У Таблиці 1.6 наведено технології та інструменти обробки даних.

Таблиця 1.6 - Технології та інструменти обробки даних

Технологія обробки даних (Data Processing Technology)	**Основні пункти**
Hadoop	Розподілена система обробки даних для великих обсягів даних.
Spark	Система обробки даних у пам'яті для великих обсягів даних.
Storm	Система обробки даних в режимі реального часу для поточних даних.
Flink	Розподілена система обробки даних для поточних і пакетних даних.
Kafka	Розподілена платформа потокової передачі даних.
NiFi	Платформа для управління потоком даних та інтеграції даних.
SQL	Декларативна мова програмування для взаємодії та управління реляційними базами даних.
Dataflow	Повністю керований сервіс для створення конвеєрів обробки даних.
Airflow	Відкрита платформа для створення, планування та моніторингу конвеєрів даних.
AWS Glue	Серверний сервіс для вилучення, перетворення та завантаження (ETL) даних.
Azure Data Factory	Хмарний сервіс інтеграції даних.
Google Cloud Dataflow	Хмарний сервіс обробки даних.

Рівень аналізу даних (Data Analysis Layer). Сюди входять технології та інструменти, які використовуються для аналізу та отримання інформації з даних, наприклад SQL, бібліотеки Python для аналізу даних (наприклад, Pandas, NumPy) та інструменти бізнес-аналітики (BI) (наприклад, Tableau, Looker). У Таблиці 1.7 наведено технології та інструменти аналізу даних.

Таблиця 1.7 - Технології та інструменти аналізу даних

Технологія аналізу даних (Data Analysis Technology)	**Основні пункти**
R	Відкрита мова програмування для аналізу даних та візуалізації.
Python	Універсальна мова програмування для аналізу даних та машинного навчання.
SAS	Набір програм для аналізу даних, бізнес-інтелекту та прогнозування.
MATLAB	Мова програмування та середовище для числових обчислень та візуалізації.
Tableau	Інструмент візуалізації даних, який дозволяє користувачам створювати інтерактивні інформаційні панелі та графіки.
Excel	Програмне забезпечення для роботи з електронними таблицями, яке може використовуватися для базового аналізу даних та візуалізації.
SQL	Декларативна мова програмування, яка використовується для видобування, аналізу та запиту даних з реляційних баз даних.
Power BI	Інструмент візуалізації даних та бізнес-інтелекту від Microsoft.
Looker	Платформа для візуалізації та дослідження даних.
Google Analytics	Веб-аналітична служба, яка відстежує та звітує про трафік на веб-сайтах.
BigQuery	Хмарний веб-сервіс аналізу великих обсягів даних від Google.

Рівень візуалізації даних (Data Visualization Layer). Це включає в себе технології та інструменти, які використовуються для створення візуалізацій і інформаційних панелей, наприклад Tableau, D3.js, matplotlib, ggplot2 та інші. У Таблиці 1.8 наведено технології та інструменти візуалізації даних.

Таблиця 1.8 - Технології та інструменти візуалізації даних

Технологія	Опис
Matplotlib	Бібліотека для створення графіків для мови програмування Python. Часто використовується для базових графіків і діаграм.
Seaborn	Бібліотека візуалізації даних, побудована на основі Matplotlib. Надає більше розширених можливостей візуалізації та більш привабливий стандартний стиль.
Plotly	Бібліотека для створення інтерактивних графіків та діаграм для веб-середовища. Може використовуватися з Python, R або JavaScript.
Bokeh	Бібліотека для створення інтерактивних графіків і діаграм для веб-середовища, схожа на Plotly. Зосереджена на створенні зручного для користувача інтерфейсу.
ggplot2	Бібліотека для створення графіків для мови програмування R, заснована на граматиці графіки. Надає високорівневий інтерфейс для створення графіків та діаграм.
D3.js	Бібліотека для створення інтерактивних візуалізацій даних для веб-середовища на JavaScript. Часто використовується для більш складних візуалізацій, таких як мережеві діаграми (графи) та карти.
Tableau	Комерційний інструмент для візуалізації даних, який дозволяє користувачам створювати інтерактивні візуалізації для веб-середовища без кодування.
Power BI	Комерційний інструмент для візуалізації даних і бізнес-аналітики, розроблений Microsoft. Дозволяє легко створювати інтерактивні інформаційні панелі та звіти.
Looker	Інструмент для бізнес-інтелекту та візуалізації даних, який надає простий спосіб створення та обміну інтерактивними і пізнавальними візуалізаціями даних.
Apache Superset	Відкрите додаток бізнес-аналітики для створення та обміну візуалізацій даних. Має простий та інтуїтивний користувацький інтерфейс, SQL-лабораторію та підтримку різних баз даних.

Рівень управління та керування даними (Data Governance & Management Layer). Сюди входять технології та інструменти, що використовуються для керування та управління даними, наприклад каталогізація даних, походження даних, якість даних і керування метаданими. У Таблиці 1.9 наведено технології та інструменти управління та керування даними.

Таблиця 1.9 - Технології та інструменти управління даними

Компонент	**Опис**	**Застереження**
Комплекс управління даними (Data Governance Framework)	Набір правил і процесів, які регулюють збір, зберігання і використання даних в організації.	– Узгоджуйте з загальною стратегією і цілями бізнесу. – Чітко визначайте ролі та відповідальності для управління даними. – Регулярно оглядайте та оновлюйте комплекс, щоб відповідати найкращим практикам та регулятивним вимогам галузі.
Команда управління даними (Data Governance Team)	Спеціалізована група осіб, відповідальних за впровадження та підтримку комплексу управління даними.	– Складається з представників різних відділів та рівнів в організації. – Переконайтеся, що члени команди мають необхідні навички та експертизу. – Надайте членам команди регулярні навчання та можливості для розвитку.
Політика управління даними (Data Management Policy)	Набір правил і процедур зі збору, зберігання і використання даних в організації.	– Чітко визначте тип даних, які збираються та способи їх використання. – Розгляньте питання забезпечення безпеки та конфіденційності даних. – Регулярно оглядайте та оновлюйте політику, щоб відповідати найкращим практикам та регулятивним вимогам галузі.
Якість даних (Data Quality)	Ступінь відповідності даних вимогам, визначеним в комплексі управління даними та політиці управління даними.	– Встановіть процеси для відстеження та покращення якості даних. – Впроваджуйте процедури валідації та очищення даних для забезпечення їх точності та повноти. – Регулярно оглядайте та оновлюйте процедури щодо якості даних.
Захист даних (Data Security)	Заходи, призначені для захисту даних від несанкціонованого доступу, використання або розкриття.	– Впроваджуйте відповідні засоби безпеки, такі як шифрування та контроль доступу, для захисту даних у спокої та під час передачі. – Регулярно моніторьте та оглядайте безпеку даних для виявлення та реагування на можливі порушення безпеки. – Навчайте співробітників кращим практикам забезпечення безпеки даних.

Продовження Таблиці 1.9 - Технології та інструменти управління даними

Конфіденційність даних (Data Privacy)	Процедури для захисту особистих даних та забезпечення відповідності відповідним регулюванням, таким як GDPR.	– Регулярно оглядайте та оновлюйте процедури конфіденційності даних для відповідності найкращим практикам та регулятивним вимогам галузі. – Навчайте співробітників кращим практикам конфіденційності даних. – Впроваджуйте відповідні технічні та організаційні заходи для захисту особистих даних, такі як псевдонімізація та контроль доступу.

Варто зазначити, що для кожного рівня стеку даних доступно багато інших інструментів і технологій, і конкретні компоненти стеку даних залежатимуть від конкретних потреб організації.

Побудова сучасного стеку даних. Зазвичай побудова сучасного стеку даних включає декілька етапів, включаючи збір даних, зберігання, обробку та візуалізацію. Ось загальний огляд процесу побудови сучасного стеку даних:

1. Визначити джерела даних, які потрібно зібрати та зберігати. Це може включати файли журналів, дані додатків, дані сенсорів та інші джерела.

2. Вибрати рішення для зберігання даних, яке може впоратися з масштабом, продуктивністю та надійністю інформації. Спільні варіанти включають реляційні бази даних, бази даних NoSQL, рішення для роботи з даними (data warehousing) та data lake.

3. Розробити ефективний конвеєр даних, який може збирати та обробляти інформацію в реальному чи близькому до реального часу. Зазвичай це передбачає використання інструментів, таких як Apache Kafka, Apache NiFi або AWS Kinesis для збору даних, та Apache Spark, Apache Storm або Apache Flink для обробки інформації.

4. Вибрати інструмент чи платформу візуалізації даних, яка допоможе досліджувати та аналізувати інформацію. Деякі популярні варіанти включають Tableau, Power BI, Looker та Grafana.

5. Впровадити надійні механізми управління даними та засоби безпеки, щоб забезпечити захист ваших даних та відповідність відповідним нормативам.

6. Моніторинг та вирішення проблеми стеку даних, і постійно оптимізуйте його продуктивність та ефективність.

Варто відзначити, що вибір технологій залежить від конкретного випадку використання, бюджету, команди та екосистеми, яка використовується, а також потреби в масштабованості.

Необхідні технічні та архітектурні компетенції для побудови стеку даних включають:

1. Досвід роботи з системами управління базами даних (СУБД) та SQL. Професійне володіння роботою з різними СУБД та мовою SQL для ефективного управління та взаємодії з базами даних.

2. Знання концепцій та технік роботи з даними. Розуміння основних концепцій та технік для ефективної роботи з даними в різних контекстах.

3. Ознайомленість із моделюванням даних та процесами ETL. Здатність моделювати дані та розуміння процесів видобутку, трансформації та завантаження (ETL) для оптимальної обробки інформації.

4. Розуміння розподілених систем та конвеєрів даних. Навички роботи з розподіленими системами та розуміння принципів конвеєрів даних для забезпечення неперервного потоку інформації.

5. Знання хмарних обчислювальних платформ. Експертиза в роботі з хмарними платформами, такими як AWS, GCP або Azure, та їх сервісами для зберігання та обробки даних.

6. Ознайомленість із технологіями великих обсягів даних та базами даних NoSQL. Знання технологій, таких як Hadoop і Spark, а також баз даних NoSQL для ефективної обробки великих обсягів різноманітних даних.

7. Вміння програмування. Професійне володіння принаймні однією мовою програмування, такою як Python або Java, для написання скриптів та автоматизації процесів ETL та конвеєрів даних.

8. Ознайомленість із найкращими практиками управління даними. Знання та дотримання найкращих практик у сфері управління даними, безпеки та дотримання стандартів для забезпечення якості та конфіденційності даних.

Приклади стеків даних в різних галузях:

- Електронна комерція: Стек даних для компанії електронної комерції може включати технології, такі як склад даних (такий як Amazon Redshift або Google BigQuery), інструменти ETL (такі як Daton) для вилучення даних з різних джерел та їх перетворення в єдиноформатний вид, і інструменти бізнес-аналітики (такі як Tableau або Looker) для аналізу та візуалізації даних.
- Охорона здоров'я: Стек даних для компанії в галузі охорони здоров'я може включати технології, такі як data lake (такий як Amazon S3 або Microsoft Azure Data Lake) для зберігання та обробки великих обсягів медичних даних, платформу медичних зображень (таку як Horos або OsiriX) для обробки та аналізу медичних зображень, та систему управління клінічними даними (таку як OpenClinica або Medidata Rave) для збору та управління даними клінічних випробувань.
- Реклама: Стек даних для компанії в галузі реклами може включати технології, такі як платформа обробки даних в реальному часі (така як Apache Kafka або Google Cloud Dataflow) для приймання та обробки великих обсягів даних в реальному часі, склад даних (такий як Amazon Redshift або Google BigQuery) для зберігання та запитування даних, і платформу передбачення (таку як TensorFlow або H2O.ai) для створення та впровадження моделей машинного навчання.
- Фінанси: Стек даних для фінансової компанії може включати технології, такі як data lake для зберігання та обробки великих обсягів фінансових даних, платформу обробки даних в реальному часі для приймання та обробки потоків фінансових даних та платформу виявлення шахрайства (таку як Kount або Feedzai) для ідентифікації та запобігання обману.
- Автомобільна галузь: Стек даних для автомобільної компанії може

включати технології, такі як data lake для зберігання та обробки великих обсягів даних сенсорів, засоби обробки даних в реальному часі, такі як Apache Kafka або Google Cloud Dataflow, і платформу машинного навчання, таку як TensorFlow або H2O.ai, для створення моделей та обробки прогнозів на льоту.

Слід відзначити, що це не вичерпний перелік, і різні компанії в одній галузі можуть використовувати різні технології в залежності від їх конкретних потреб та ресурсів. Давайте розглянемо подробиці стеку даних для електронної комерції та роздрібної торгівлі. У Таблиці 1.10 розглянуто компонування сучасного стеку даних для електронної комерції та роздрібної торгівлі

Таблиця 1.10 - Сучасний стек даних для електронної комерції

Компонент	**Інструмент/Платформа**	**Сфери використання**
Зберігання даних (Data Warehousing)	Redshift, BigQuery, Snowflake	Зберігання та аналіз великих обсягів даних про клієнтів, продажів та продуктів для розуміння патернів покупок та виявлення ключових тенденцій та можливостей.
Проведення даних (Data Pipeline)	Daton	Збір даних в реальному часі з різних джерел, таких як журнали веб-сайтів, соціальні медіа та системи реєстрації продажів, їх трансформація та очищення, а потім завантаження в хранилище даних для аналізу.
Візуалізація даних (Data Visualization)	Tableau, Looker, Power BI	Створення інтерактивних інструментів для відстеження ключових метрик, таких як відвідуваність веб-сайту, продажі та поведінка клієнтів, а також ідентифікація областей для покращення.

Продовження Таблиці 1.10 - Сучасний стек даних для електронної комерції

Моделювання даних (Data Modeling)	ERD, Зіркова схема (Star schema), Схема сніжинка (Snowflake schema)	Структурування даних в системі зберігання для підтримки ефективного запитування та аналізу, таких як розкладання даних про продажі за продуктом, місцем розташування та періодом часу.
Бізнес-інтелект (Business Intelligence)	Tableau, PowerBI, QlikView, SAP BusinessObjects	Аналіз даних про клієнтів для сегментації та спрямування на конкретні групи клієнтів, прогнозування продажів та потреб у товарах на складі та виявлення можливостей для хрест-продажу та упродажу.

ETL і ELT — найпоширеніші способи доставки даних з одного або кількох джерел до централізованої системи для зручності доступу та аналізу. Обидві ці методики складаються з етапів extract (вилучення), transform (перетворення) та load (завантаження). Різниця полягає у послідовності дій. Хоча можна подумати, що невелика зміна порядку етапів ніяк не впливає, насправді для потоку інтеграції це змінює все.

У цьому розділі розглядається процеси ETL і ELT, а також порівнюється їх за важливими критеріями, щоб для розуміння, який найкраще підходить для різних конвеєра даних. На Рисунок 1.5 та Рисунок 1.6 зображена схема процесу ETL та ELT.

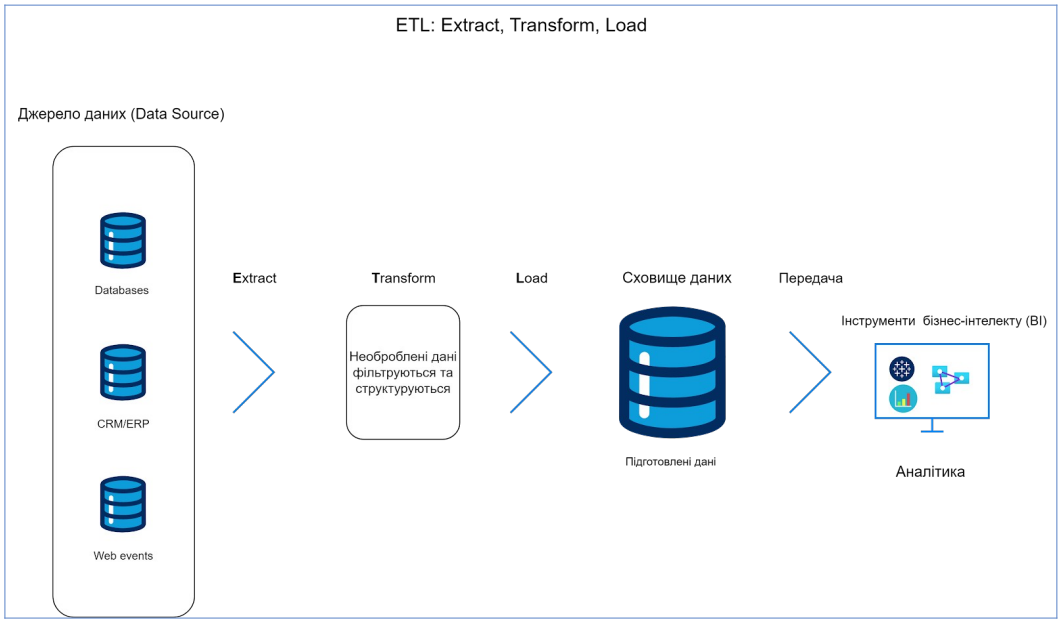

Рисунок 1.5 - Схема процесу ETL

Історія розвитку ETL та ELT. Процес ETL відомий з 1970-х років і став дуже популярним з розвитком сховищ даних. Але традиційні сховища даних вимагали налаштування всього ETL процесу для кожного джерела даних окремо. Еволюція хмарних технологій змінила всі доступні можливості. Тепер компанії можуть зберігати необмежену кількість необроблених даних та аналізувати їх у міру потреби. Процес ELT став сучасним методом інтеграції даних отримання ефективної аналітики.

Ключові етапи процесів ETL та ELT. Потік даних і в ETL, і ELT використовує три базові етапи. Незважаючи на однакову назву, кожен із етапів цих методик відрізняється не тільки за порядком їх виконання, а й за тим, як вони виконуються.

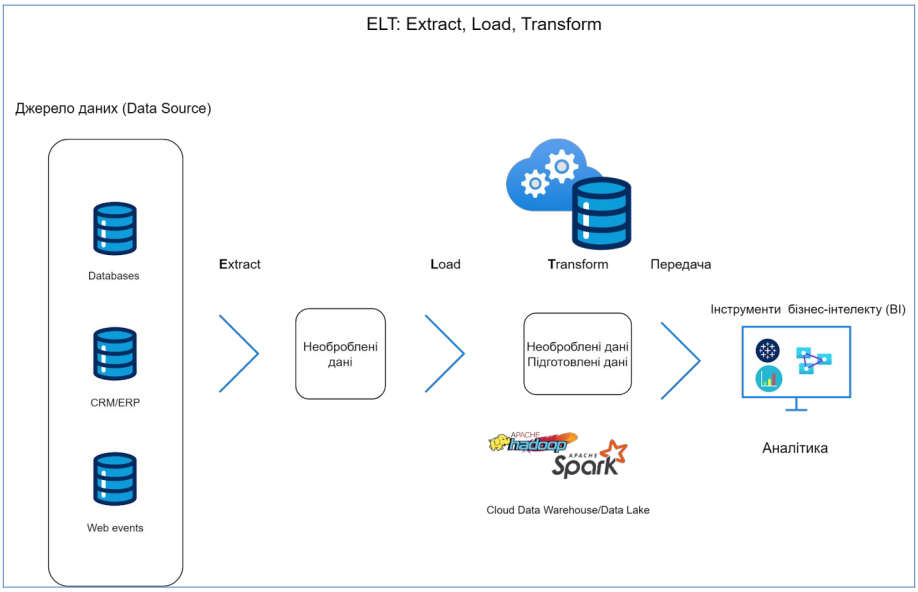

Рисунок 1.6 Схема процесу ELT

Вилучення (Extract). Перший етап в обох процесах. Подорож даних завжди починається з їх вилучення та копіювання з пулу джерел – систем ERP та CRM, баз даних SQL та NoSQL, додатків SaaS, веб-сторінок, неструктурованих файлів, електронних листів, мобільних додатків тощо. Через складнощі кожної із систем-джерел перша фаза може бути досить хитромудрою.

Дані зазвичай витягуються одним із трьох способів.

Повне вилучення застосовується до систем, які можуть відрізняти нові чи змінені записи. У разі єдиним способом отримання даних із системи є вилучення всіх записів, як старих, і нових.

Часткове вилучення з повідомленнями про оновлення - найзручніший спосіб вилучення даних із систем-джерел. Він можливий, якщо в системах є сповіщення про зміни записів, завдяки чому відсутня необхідність завантаження всіх даних.

Інкрементне вилучення або часткове вилучення без повідомлень про оновлення – це спосіб вилучення лише тих записів, які були змінені.

У разі використання методу ETL користувачам потрібно заздалегідь планувати, які елементи даних необхідно витягувати для подальшого перетворення та завантаження. З іншого боку, ELT дозволяє отримувати всі дані

миттєво. Користувачі можуть пізніше вирішити, які дані потрібно перетворювати та аналізувати.

Перетворення (Transform). Другий етап у ETL та третій етап у ELT.

У фазу перетворення входить послідовність дій, націлених на підготовку даних для зміни їх під параметри іншої системи або досягнення потрібного результату.

Перетворення можуть включати такі дії:

1. сортування та фільтрування даних для позбавлення від нерелевантних елементів.
2. усунення дублікатів та очищення.
3. транслювання та перетворення.
4. видалення або шифрування для захисту вразливої інформації.
5. з'єднання або поділ таблиць.

У ETL ці операції відбуваються поза цільової системи, на етапі підготовки. За реалізацію цих процесів відповідають дата-інженери. Наприклад, сховища даних Online Analytical Processing (OLAP) допускають зберігання лише реляційних структур даних, тому дані попередньо повинні перетворюватися на SQL-читаний формат. Всі перетворення можуть відбуватися лише один раз, через що ETL виявляється досить негнучким. У разі, якщо необхідно застосувати до вже перетворених даних новий тип аналізу, може знадобитися модифікація всього конвеєра даних з нуля.

Метод ELT є гнучким і зручним для перетворень, оскільки дані передаються безпосередньо в сховище даних, озеро даних або data lakehouse, де вони можуть валідуватися, структуруватися і перетворюватися по-різному і в будь-який момент. Більш того, «сирі» дані можуть зазнавати незліченних перетворень, оскільки вони зберігаються необмежено. Так як все відбувається всередині цільової системи, аналітики даних можуть допомагати дата-інженерам у виконанні перетворень, використовуючи для цього SQL.

Завантаження (Load). Другий етап у ELT/третій етап у ETL Цей етап

полягає у завантаженні даних у цільову систему зберігання даних, щоб до них могли отримувати доступ користувачі. Потік процесу ETL передбачає імпорт раніше вилучених і вже підготовлених даних із проміжної бази даних до цільового сховища даних або бази даних. Це виконується шляхом фізичної вставки окремих записів як нових рядків таблиці сховища за допомогою команд SQL, або за допомогою сценарію пакетного завантаження великого обсягу даних.

ELT, своєю чергою, направляє масив «сирих» даних у цільове місце зберігання, минаючи проміжний рівень. Це сильно заощаджує час циклу «витяг-доставка». Як і у разі вилучення, дані можуть завантажуватися або повністю, або частково.

ELT і ETL детальне порівняння, наведено в Таблиці 1.11. Щоб допомогти отримати переваги та обмеження всіх підходів до інтеграції даних, виділимо найбільш важливі критерії, якими порівнювати ETL і ELT.

Таблиця 1.11 - Порівняння ETL та ELT

Атрибут	**ETL**	**ELT**
Розгортання	Хмарний / локальний	Хмарний
Зрілість технології	Надійний і зрілий	Відносно новий
Великі обсяги даних	ETL ускладнює роботу з великими обсягами даних	ELT полегшує роботу з великими обсягами даних
Тип даних	Переважно структуровані	Всі типи: структуровані та не структуровані
Цільова система	Локально/хмарне сховище даних	Хмарне сховище даних/озеро даних
Вартість	дорого (локально), витратоефективний (хмарно)	Витратоефективний
Обслуговування	ETL потребує більшого обслуговування, якщо воно виконується традиційним способом. Cloud ETL не потребує технічного обслуговування	ELT вимагає невеликого обслуговування

Продовження Таблиці 1.11 Порівняння ETL та ELT

Час завантаження	Завантаження даних відбувається повільніше	Завантаження даних відбувається швидше
Час трансформації	Значно повільніше, оскільки перетворення відбувається на окремому сервері	Швидше, оскільки перетворення відбувається всередині цільової системи за вимогами
Відповідність	Дозволяє захистити чутливі дані, редагувати, шифрувати перш ніж потрапити до системи	Вимагає завантаження всіх даних без редагування/видалення чутливих даних
Інструменти	Informatica, Cognos, Oracle, IBM	Kafka, Hevo Data, Talend
Необхідна експертиза	Величезний досвід роботи з джерелами даних, експортом, трансформацією. ETL спеціалісти, інженери,аналітики	Глибоке знання існуючих інструментів і сильні спеціальні навички. ELT спеціалісти
Краще використовувати	Невелика кількість даних, структуровані дані, застарілі системи, реляційні бази даних	Швидке отримання всіх необроблених даних в систему, неструктуровані дані, проекти з тенденцією до масштабності

Зрілість технологій. ELT - відносно нова методологія, тому для неї існує менше напрацювань та професійних компетенцій. Такі інструменти та системи все ще знаходяться на ранній стадії розвитку. Знаючих процес ELT фахівців складніше знайти.

З іншого боку, практика ETL вже досить доросла. Оскільки вона використовується вже довгий час, існує достатня кількість якісно розроблених інструментів, досвідчених спеціалістів та напрацювань. Ключовий висновок: ETL надійніший і дорослiший.

Тип та розмір даних. Поряд із структурованими даними, ELT дозволяє обробляти великі обсяги нереляційних та неструктурованих даних, які потрібні для аналітики big data та бізнес-аналітики.

У випадку, коли всі вихідні дані надходять з реляційних баз даних або коли

їх потрібно ретельно очищати перед завантаженням у цільову систему, часто надається перевага ETL. Ключовий висновок: гнучкий і масштабований ELT перевершує свого предка з погляду можливостей споживання великих масивів різних типів даних.

Підтримка сховищ/озер даних. ETL застосовується при роботі зі сховищами даних OLAP, легасі-системами та реляційними базами даних. Він не має підтримки озер даних. ELT - сучасний метод, який можна використовувати з хмарними сховищами та озерами даних. Ключовий висновок: різні методи підходять у різних випадках використання.

Витрати. ELT не тільки економічно ефективніший у порівнянні з високопродуктивними рішеннями на потужностях компанії, але й забезпечує нижчі початкові витрати. Багато постачальників хмарних послуг пропонують гнучкі тарифні плани.

Архітектури, в яких працюють традиційні процеси ETL, можуть бути витратними через початкові інвестиції в обладнання і витрати, пов'язані з потужністю движка перетворень. У той же час, сучасні хмарні сервіси ETL забезпечують гнучкі тарифні плани на підставі вимог щодо масштабів використання. Ключовий висновок: ELT відносно менш витратний, порівняно з ETL на потужностях компанії.

Обслуговування. ELT — це хмарне рішення з автоматизованими функціями, яке не потребує або майже не потребує обслуговування. Всі дані в ньому готові і можуть частинами трансформуватися для аналітичних цілей.

Процеси ETL вимагають вищого рівня обслуговування, коли йдеться про рішення на потужностях компанії з фізичними серверами. Що стосується хмарних рішень ETL з автоматизованими процесами, то ситуація в них не надто відрізняється від ситуації в ELT: вони не потребують особливого обслуговування. Ключовий висновок: з погляду обслуговування ELT перевершує ETL на потужності компанії.

Час завантаження. Завдяки вбудованим можливостям обробки хмарних

рішень, що дозволяє завантажувати дані у «сирих» форматах без попередніх перетворень, ELT порівняно з ETL знижує час завантаження.

При ETL процес завантаження даних повільніше через необхідність перетворення даних на окремому сервері обробки перед поставкою в цільову систему. Ключовий висновок: ELT забезпечує швидше завантаження.

Час перетворення. При ETL перетворення виконуються на окремому сервері і відбуваються значно повільніше, особливо у великих обсягах даних.

При ELT за перетворення відповідає цільова система. Завдяки розділенню зберігання та обчислень дані можна зберігати у їх вихідному форматі та перетворювати за потребою. Розмір даних не впливає на швидкість. Ключовий висновок: при ELT перетворення займають менше часу.

Комплаєнс. При ELT дані завантажуються так, без попередніх скорочень та шифрування, що може зробити дані вразливими для злому та порушувати стандарти комплаєнсу.

ETL дозволяє редагувати, шифрувати та видаляти вразливі дані перед їх передачею до сховища даних. Отже, компаніям простіше захищати дані та дотримуватись різних стандартів комплаєнсу, у тому числі HIPAA, CCPA та GDPR. Ключовий висновок: ETL має перевагу з погляду комплаєнсу.

Інструменти та компетенції. Реалізація обох процесів потребує глибоких знань існуючих інструментів та високих навичок.

Через незрілість ELT фахівців із компетенціями у цій методології знайти складно. Kafka, Hevo Data, Talend та деякі інші програми надають вичерпні можливості ELT та ETL.

Для виконання таких процесів, як передача, експорт, перетворення та міграція даних, потрібні досвідчені фахівці з ETL. На щастя, у цій сфері знайти компетентного фахівця найпростіше. Одним із прикладів традиційних інструментів ETL є Informatica, Cognos та Oracle. Ключовий висновок: інструменти та компетенції ETL на ринку представлені ширше.

Підсумок: обидва процеси мають свої переваги та обмеження. Щоб вибрати

переможця в цій сутичці (якщо він взагалі є), розгляньмо можливі способи застосування ETL і ELT.

Області застосування ETL та ELT. Хмарні сховища даних відкрили нові горизонти для інтеграції даних, проте вибір між ETL та ELT насамперед залежить від потреб компанії.

Краще використовувати ETL, потрібно забезпечити відповідність встановленим стандартам захисту вразливих даних клієнтів. Наприклад, організації у сфері охорони здоров'я мають забезпечувати відповідність вимогам HIPAA Security Rule. Тому вони можуть вибрати ETL, щоб маскувати, шифрувати або видаляти вразливі дані перед завантаженням у хмару. Робота лише зі структурованими даними та/або з невеликими частинами даних.

У компанії працює легасі-система або реляційні бази даних на власних потужностях. Прикладами використання ETL у реальному житті може бути вилучення даних електронних медичних карток (EHR) при модернізації легасі-системи EHR: дані пацієнтів спочатку мають бути селективно вилучені з легасі-системи, а потім перетворені на потрібний формат для нової системи.

Краще використовувати ELT, якщо найважливішим фактором стратегії є ухвалення рішень у реальному часі. Швидкість інтеграції даних – ключова перевага ELT, оскільки цільова система здатна виконувати перетворення та завантаження даних паралельно. Це, своєю чергою, дозволяє генерувати аналітику у реальному часі.

На прикладі, компанія працює з величезними обсягами даних як структурованих, так і неструктурованих. Наприклад, транспортної компанії, що використовує телематичні пристрої у своїх машинах, може бути потрібна обробка великих обсягів даних, що генеруються датчиками, відеореєстраторами, GPS-трекерами і так далі. Для обробки всіх цих даних потрібні величезні ресурси, а також інвестиції у ці ресурси. ELT дозволяє економити гроші та забезпечувати покращену продуктивність. Працюєте з хмарними проектами чи гібридними архітектурами. Хоча сучасний ETL відкрив свої двері хмарним сховищам даних,

він все одно вимагає окремого двигуна для виконання перетворень перед завантаженням даних у хмару. ELT усуває потребу в установці проміжних двигунів обробки, а тому краще підходить для хмарних та гібридних систем.

Збираєтеся запускати проект із Big Data. ELT якраз і розроблявся для досягнення основних завдань Big Data: обсягу, різноманітності, швидкості та достовірності.

Якщо в компанії є команда дата-саєнтистів, яка потребує доступу до всіх «сирих» даних для їх використання в проектах машинного навчання. Проект масштабуватиметься і скористатися перевагами великої масштабованості хмарних сховищ та озер даних.

Майбутнє ETL та ELT. Технології сховищ даних швидко розвиваються. Сучасні хмарні рішення поступово замінюють традиційні засоби зберігання даних. Надання хмарними платформами масштабованого сховища даних та обчислювальних ресурсів з гнучкими тарифними планами дозволяє зберігати великі обсяги даних, забезпечувати доступ і обробляти їх. Тому все більше компаній відходить від використання ETL в обслуговуванні конвеєрів даних і вибирає метод ELT. Поширення озер даних теж грає на руку ELT, тому що все більше організацій вважають за краще виконувати міграцію своїх процесів роботи з даними з власних потужностей у хмару. Управління озерами даних виконується за допомогою платформ big data на кшталт згаданої вище Apache Hadoop або системи управління базами даних NoSQL. ELT також є кращим для команд дата-саєнтистів, оскільки надає їм можливість використовувати «сирі» дані та перетворювати їх під власні унікальні вимоги.

З урахуванням всього вищесказаного, ELT здається логічним вибором майбутнього створення ефективних потоків даних, оскільки він має безліч переваг проти ETL. ELT вигідно економічно, гнучкий і вимагає меншої кількості ресурсів для обслуговування. Він підходить компаніям різного розміру в багатьох областях. ETL - це застарілий і повільний процес, що має безліч прихованих каменів, про які може спіткнутися компанія на шляху до інтеграції даних.

Одним із ключових аспектів сучасної обробки даних є використання OLAP (Online Analytical Processing) та OLTP (Online Transaction Processing) систем. Ці дві концепції визначають різні способи обробки та аналізу інформації, але є важливими складовими для розвитку сучасних бізнес-та наукових систем.

Основи OLAP. OLAP системи орієнтовані на аналітику та аналіз великих обсягів даних. Основною їхньою характеристикою є можливість вибору та агрегації даних з різних джерел для подальшого аналізу. Вони спроектовані для роботи з мультимедійними та багатовимірними даними, що надає можливість використовувати їх для складних операцій аналізу. Типовим прикладом OLAP є системи, які дозволяють виконувати сложні запити та побудову звітів, спрямовані на виявлення закономірностей та трендів в накопичених даних.

Основи OLTP. OLTP системи, навпаки, спроектовані для операцій обробки транзакцій та забезпечують ефективне виконання транзакційно-орієнтованих завдань. Їхнє призначення полягає в обробці транзакцій у реальному часі, забезпечуючи коректність та стійкість до збоїв при обміні даними. OLTP використовується для щоденних операцій бізнесу, таких як обробка замовлень, транзакції з клієнтами та управління запасами.

Переваги та недоліки використання OLAP і OLTP. OLAP і OLTP системи служать різним цілям, і їхнє використання залежить від конкретних потреб користувача чи організації. Однією з переваг OLAP є здатність виявлення та аналіз складних відносин в даних, тоді як OLTP гарантує ефективну обробку транзакцій. Однак важливо враховувати, що використання обох систем може вимагати значних ресурсів та оптимізації для досягнення максимальної продуктивності.

Застосування OLAP в сучасних системах. Однією з ключових особливостей OLAP є його здатність до роботи з даними в багатовимірному просторі. В сучасних бізнес-системах це виявляється особливо важливим, оскільки багатовимірний аналіз може допомогти виявляти приховані відносини та залежності між різними аспектами діяльності підприємства. За допомогою OLAP

можна побудувати складні звіти, графіки та інші візуалізації для ефективного представлення результатів аналізу.

OLTP у реальних бізнес-сценаріях. OLTP системи зазвичай застосовуються в реальному часі та надають можливість обробляти велику кількість транзакцій швидко та надійно. У банківській сфері, наприклад, OLTP забезпечує обробку платежів, переказів та інших транзакційних операцій. Його використання також розповсюджене в онлайн-торгівлі, де швидка обробка замовлень та платежів є критично важливою для задоволення потреб споживачів.

Взаємодія між OLAP та OLTP. У реальних бізнес-сценаріях часто виникає необхідність в одночасному використанні як OLAP, так і OLTP. Така взаємодія дозволяє підприємствам не лише ефективно обробляти транзакції, але й одночасно виконувати глибокий аналіз даних для прийняття обґрунтованих стратегічних рішень. Використання обох концепцій може вимагати складних систем інтеграції та оптимізації, але це є важливою частиною модернізації та оптимізації бізнес-процесів.

Виклики та перспективи використання OLAP і OLTP. Хоча OLAP та OLTP мають безліч переваг, вони також стикаються із викликами. Спільне використання може призводити до конфліктів використання ресурсів та може вимагати чіткої архітектурної роботи. Вирішення цих викликів визначається не лише технічними аспектами, але й потребами конкретного бізнес-сценарію.

Big Data та їх обробка. Big Data є однією з найважливіших та найактуальніших тем у світі інформаційних технологій, оскільки великі обсяги різноманітних даних стають не лише великим резервуаром інформації, але і викликом для ефективної обробки та аналізу. У цьому підрозділі магістерської роботи глибоко вдаватимемось у питання, що стосуються Big Data, розглядаючи його характеристики, виклики та стратегії обробки в розподіленій обчислювальній інфраструктурі.

Характеристики Big Data. Big Data визначається не лише об'ємом (Volume), розмаїттям (Variety) та швидкістю (Velocity), але й трьома іншими "B" - вартістю

(Value), достовірністю (Veracity) та високою швидкістю зміни (Volatility). Вартість полягає в можливості витягти значущі дані, достовірність визначає надійність інформації, а висока швидкість зміни вказує на те, як швидко дані змінюються та оновлюються.

Стратегії обробки Big Data. Однією з ключових стратегій обробки Big Data є паралельне обчислення. Це відмінно працює в розподіленій обчислювальній інфраструктурі, дозволяючи розділити завдання на менші частини та ефективно виконувати їх одночасно на різних вузлах мережі. Іншою стратегією є розподілена зберіганій та обробка даних, де дані зберігаються та обробляються на різних фізичних машинах або в хмарних обчисленнях.

Технології для обробки Big Data. Для ефективної обробки Big Data використовуються різноманітні технології, такі як Apache Hadoop, Apache Spark, Apache Flink, Apache Kafka тощо. Apache Hadoop надає фреймворк для зберігання та обробки великих обсягів даних, використовуючи концепцію розподіленої файлової системи та MapReduce для обчислень. Apache Spark, з іншого боку, пропонує швидший та більш гнучкий спосіб обробки даних, використовуючи концепцію Resilient Distributed Datasets (RDD).

Виклики обробки Big Data. Попри беззаперечні можливості, які принесло використання Big Data, існують деякі складнощі та виклики, з якими стикаються практики і дослідники у цій області.

Проблеми безпеки та конфіденційності. Однією з найбільших труднощів у сфері Big Data є забезпечення безпеки та конфіденційності даних. Великі обсяги інформації часто містять особисті дані, і їх обробка вимагає надзвичайно ефективних методів шифрування, контролю доступу та захисту від несанкціонованого використання.

Проблеми етики обробки особистих даних. Зростання обсягів даних також покладає відповідальність за етичне та відповідальне використання особистої інформації. Належна обробка та зберігання даних вимагає дотримання високих стандартів етики, а також врахування прав та свобод осіб, чиї дані обробляються.

Високі вимоги до обчислювальних ресурсів. Обробка Big Data вимагає значних обчислювальних ресурсів для забезпечення швидкості та ефективності. Створення та підтримка розподіленої інфраструктури, яка здатна справлятися з великими обсягами даних, потребує значних інвестицій та технічних знань.

Необхідність вдосконалених методів аналізу. Разом з великими обсягами даних зростає необхідність у вдосконалених методах аналізу та інтелектуальних технологіях для виявлення корисної інформації серед безмежної кількості даних. Машинне навчання та аналітичні методи грають важливу роль у витяганні цінності з Big Data.

У наступних розділах цього дослідження ретельно розглядається різноманітні аспекти, пов'язані з обробкою та аналізом великих обсягів даних, і розглянемо різні підходи та технології, які використовуються для вирішення цих викликів.

1.2 Огляд досліджень та робіт

Гіганти IT-інфраструктури активно вдосконалюють та впроваджують комплексні технологічні рішення, спрямовані на створення стеків інструментів, які забезпечують ефективну організацію робочого середовища. У цих стеках враховується велика кількість аспектів, включаючи віддалену обробку даних на спеціалізованих серверах. Такий підхід передбачає використання концепцій Cloud Computing для забезпечення гнучкості та масштабованості обчислень, що стає важливим в контексті сучасних вимог до обробки та аналізу великих обсягів даних.

На сьогоднішній день існує значна кількість наукових та прикладних досліджень, присвячених аналізу та впровадженню технологій Cloud Computing у різних сферах. Вони висвітлюють аспекти використання віртуалізованих ресурсів, послуг та інфраструктури для оптимізації обчислювальних процесів та забезпечення високої доступності. Дослідження також стосуються впливу цих

технологій на архітектуру систем, аспекти безпеки та ефективність в реальних умовах використання.

Проведений аналіз робіт у даній галузі дозволяє з'ясувати ключові аспекти та тенденції, що визначають сучасний стан використання технологій Cloud Computing у розробці інфраструктури для обробки великих обсягів даних. У подальших підрозділах буде розглянуто важливі деталі та висновки, отримані в ході цього огляду.

Сучасна архітектура даних. Сучасна архітектура даних визнає ідею, що застосування універсального підходу до аналітики зрештою призводить до компромісів. Йдеться не просто про інтеграцію озера даних зі сховищем даних, а про інтеграцію озера даних, сховища даних і спеціально створених сховищ, що забезпечує уніфіковане управління та легкий переміщення даних. Завдяки сучасній архітектурі даних на AWS клієнти можуть швидко створювати масштабовані озера даних, використовувати широку та глибоку колекцію спеціально створених сервісів даних, забезпечувати відповідність через уніфікований доступ до даних, безпеку та керування, масштабувати свої системи за низькими витратами без зниження продуктивності та легко обмінюватися даними між межами організації, дозволяючи їм приймати рішення зі швидкістю та гнучкістю в масштабі. На Рисунку 1.7 зображені технологічні рішення AWS.

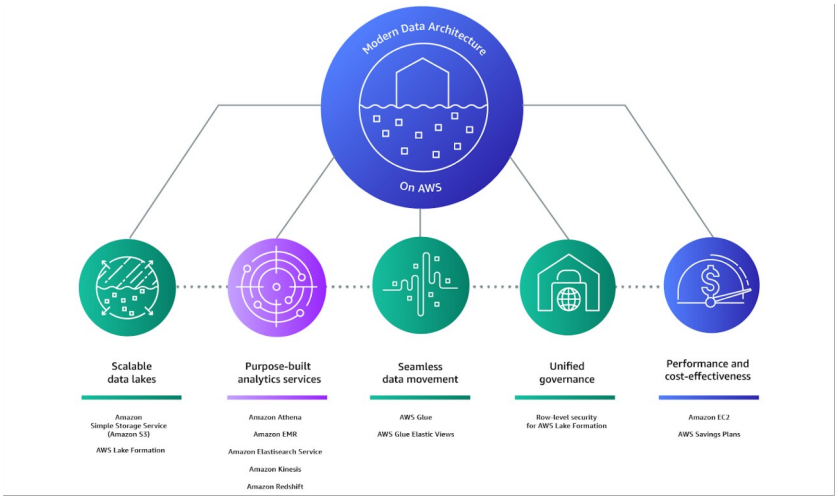

Рисунок 1.7 - Сучасна архітектура даних AWS

Потреба в сучасній архітектурі даних. Обсяги даних зростають із безпрецедентною швидкістю, вибухаючи від терабайтів до петабайтів, а іноді й ексабайтів. Традиційні підходи до локальної аналітики даних не можуть впоратися з такими обсягами даних, оскільки вони недостатньо добре масштабуються та є надто дорогими. Багато компаній беруть усі свої дані з різних силосів і агрегують усі ці дані в одному місці, яке багато хто називає озером даних, щоб проводити аналітику та ML безпосередньо на основі цих даних. В інший час ці ж компанії зберігають інші дані в спеціально створених сховищах даних, щоб аналізувати та швидко отримувати розуміння як структурованих, так і неструктурованих даних. Це переміщення даних може бути «Inside-out data movement», «Outside-in data movement», «Around the perimeter data movement» або «Sharing across data movement», оскільки дані мають силу тяжіння.

Пересування даних зсередини назовні. Клієнти зберігають дані в екосистемі даних, а потім переміщають частину цих даних в спеціалізоване сховище даних для виконання додаткового машинного навчання чи аналітики. Відповідне зображене на Рисунку 1.8.

Приклад: Дані про кліки з веб-додатків можна збирати безпосередньо в екосистемі даних, і частина цих даних може бути переміщена до сховища даних для щоденного звітування.

Рисунок 1.8 - Пересування даних зсередини назовні

Зовнішнє переміщення даних. Клієнти зберігають дані в спеціалізованих сховищах даних, таких як сховище даних або база даних, і переміщають ці дані до екосистеми даних для виконання аналізу. Відповідне зображене на Рисунку 1.9.

Приклад: копіюють результати запиту щодо продажу товарів у певному регіоні з сховища даних в екосистему даних для виконання алгоритмів рекомендацій продуктів на більшому наборі даних з використанням машинного навчання.

Рисунок 1.9 - Зовнішнє переміщення даних.

Переміщення даних навколо периметра. Безперервна інтеграція екосистему даних, сховища даних та спеціалізованих сховищ даних. Відповідне зображене на Рисунку 1.10.

Приклад: може знадобитися скопіювати дані про каталог товарів, збережені в базі даних, до сервісу пошуку для спрощення перегляду каталогу продуктів та перенаправлення запитань пошуку від бази даних.

Рисунок 1.10 - Переміщення даних навколо периметра

Обмін даними через рух даних. Клієнти використовують сучасну архітектуру даних для забезпечення управління та обміну даними через логічні або фізичні межі управління для створення Доменів Даних, які вирівнюються за лініями бізнесу. Відповідне зображене на Рисунку 1.11.

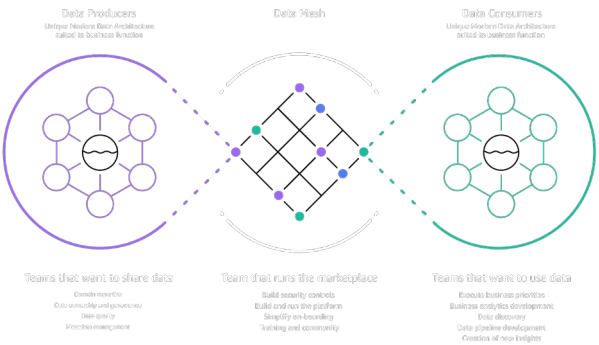

Рисунок 1.11 - Обмін даними через рух даних

Стовпи сучасної архітектури даних. Організації взяли свої дані з різних сховищ та агрегують усі ці дані в одному місці для подальших аналітичних та

машинного навчання. Щоб отримати максимальну користь від цього, їм потрібно використовувати сучасну архітектуру даних, яка дозволяє легко переміщати дані між озерами даних та спеціалізованими сховищами даних. Цей сучасний спосіб проектування вимагає:

- наразі налаштування та управління озерами даних вимагає значної кількості ручних та часомістких завдань. AWS Lake Formation автоматизує ці завдання, щоб ви могли створювати та захищати своє озеро даних за декілька днів замість місяців. Для зберігання даних в озері, Amazon S3 є найкращим місцем для створення озера даних, оскільки воно має неперевершену долю в 11 дев'яток та доступність на рівні 99,99%; найкращі можливості з безпеки, відповідності та аудиту із об'єктним веденням журналів та управлінням доступом; найбільшу гнучкість із п'ятьма рівнями зберігання; та найнижчу вартість, з цінами, що починаються з менше ніж 1 долара за терабайт на місяць.

Amazon Simple Storage Service (S3). Об'єктне сховище, створене для зберігання та отримання будь-якої кількості даних з будь-якого місця.

AWS Lake Formation. AWS Lake Formation спрощує створення безпечного озера даних за декілька днів.

Amazon Athena. Запитуйте та аналізуйте дані, збережені в озері даних, використовуючи стандартний SQL.

AWS Glue. Безсерверний сервіс інтеграції даних для аналітики, машинного навчання та розробки застосунків.

- Scalable Data Lakes. Усі ці сервіси розроблені так, щоб бути найкращими у своєму класі, що означає, що ніколи не доведеться компрометувати щодо продуктивності, масштабу чи вартості при їх використанні. Наприклад, Amazon Redshift працює у 3 рази швидше і коштує принаймні на 50% дешевше, ніж інші хмарні склади даних. Spark на Amazon EMR працює на 1,7 рази швидше, ніж стандартний Apache Spark 3.0, і ви можете проводити аналіз на петабайтному рівні за менше половини вартості порівняно з традиційними рішеннями на власних серверах.

Amazon Athena. Запитуйте та аналізуйте дані, збережені в озері даних, використовуючи стандартний SQL.

Amazon EMR. Amazon EMR - це провідна у галузі хмарна платформа великих даних для обробки великих обсягів даних за допомогою відкритих інструментів, таких як Apache Spark, Hive, Presto та інші фреймворки для великих даних.

Amazon OpenSearch Service. Amazon OpenSearch Service спрощує для вас виконання інтерактивного аналізу журналів, моніторингу за додатками в реальному часі, пошуку на веб-сайті та інших завдань.

Amazon Kinesis. Amazon Kinesis полегшує збір, обробку та аналіз потоків відео та даних в реальному часі.

Amazon Redshift. Amazon Redshift є найшвидше зростаючим хмарним складом даних з можливістю виконання складних та аналітичних запитів до петабайтів структурованих даних.

- Purpose-built analytics services. AWS полегшує для вас поєднання, переміщення та реплікацію даних між різними сховищами даних та вашим сховищем даних. Наприклад, AWS Glue надає всеосяжні можливості інтеграції даних, що полегшують виявлення, підготовку та поєднання даних для аналітики, машинного навчання та розробки програм, тоді як Amazon Redshift може легко опитувати дані в сховищі даних S3. Жоден інший постачальник аналітики не робить так легким для вас переміщення ваших даних масштабно туди, де це найбільше потрібно.

AWS Glue. Безсерверна служба інтеграції даних для аналітики, машинного навчання та розробки програм.

Amazon Kinesis Data Firehose. Підготовка та завантаження потоків даних в реальному часі в сховища даних та служби аналітики.

- Unified data access. Це може бути викликано тим, що управління безпекою, контролем доступу та слідами аудиту для всіх сховищ даних в вашій організації є складним, часом затратним та вразливим до помилок. AWS надає можливість

управління для керування доступом до всіх ваших даних у сховищі даних та спеціальних сховищах даних з єдиного місця. AWS Lake Formation дозволяє централізовано визначати та управляти політиками безпеки, управління та аудиту, що призводить до єдиної системи контролю доступу для обміну даними на рівні всієї підприємства.

AWS Lake Formation. AWS Lake Formation дозволяє легко налаштувати безпечний сховище даних за кілька днів

- Performant and cost-effective. Поміж провідними аналітичними службами, які пропонують відмінну ціноутворення, S3 Intelligent Tiering заощаджує клієнтам до 70% від вартості зберігання даних у сховищі даних, а Amazon EC2 надає доступ до вибору із понад 200 типів екземплярів, мережевої пропускної здатності до 100 Гбіт/с та можливості вибору між екземплярами з запитанням, зарезервованими та інстанціями "spot".

Amazon Simple Storage Service (S3). Об'єктне сховище, створене для зберігання та отримання будь-якої кількості даних з будь-якого місця.

Amazon EC2. EC2 надає безпечну та масштабовану потужність обчислень для підтримки практично будь-якого завдання.

Amazon EMR. Amazon EMR є провідною обчислювальною платформою у хмарі для обробки великих обсягів даних за допомогою відкритих інструментів, таких як Apache Spark, Hive, Presto та інші фреймворки великих даних.

Реалізація в AWS вимог до ETL та ELT. Аналітика в AWS - це сукупна назва широкого асортименту аналітичних сервісів Amazon Web Services (AWS), з яких ви знайдете відповідний варіант для будь-яких потреб в аналітиці даних. AWS дозволяє організаціям усіх розмірів знайти новий підхід для роботи з даними.

Нижче наведено кілька сервісів AWS, які можна використовувати в інфраструктурі ETL та ELT.

- Amazon Aurora підтримує інтеграцію з Amazon Redshift із нульовим використанням ETL. Така інтеграція забезпечить отримання аналітики та

використання машинного навчання в Amazon Redshift за декількома петабайтами даних про транзакції з Aurora в режимі, близькому до реального часу.

- Конвеєр даних AWS – це керований сервіс ETL, який дозволяє описувати правила переміщення та перетворення даних між різними сервісами AWS.
- AWS Glue – це безсерверний сервіс інтеграції даних для організації керованих подіями завдань ETL та безкодових завдань ETL.
- AWS IoT Greengrass підтримує будь-які сценарії ETL на периферійних пристроях, додаючи можливості хмарної обробки та хмарної логіки у локальне середовище.
- Amazon Redshift дозволяє налаштувати будь-які робочі процеси ELT і безпосередньо звертатися до наборів даних із різних джерел.

Реалізація OLAP та OLTP в AWS. Аналітика в Amazon Web Services (AWS) надає різні керовані хмарні послуги для аналітичної обробки онлайн (OLAP) та обробки транзакцій онлайн (OLTP). AWS пропонує спеціально розроблені сервіси, що забезпечують найкращу продуктивність, масштабованість та низьку вартість, для переміщення, зберігання, аналітики даних та багато іншого.

Приклади сервісів AWS, які можуть забезпечити підтримку ваших потреб у OLAP та OLTP:

- Amazon Redshift – це хмарне сховище даних, розроблене спеціально для OLAP.
- Служба реляційних баз даних Amazon (Amazon RDS) – це реляційна база даних із функціональністю OLAP. Її можна використовувати для виконання робочих навантажень OLTP або Oracle OLAP для виконання складних запитів до розмірних кубів.
- Amazon Aurora – це хмарна реляційна база даних, сумісна з MySQL та PostgreSQL, яка може виконувати і робочі навантаження OLTP, і складні робочі навантаження OLAP.

Для організації Data Engineering на основі технологій Google та Google Cloud Platform (GCP) можливо використовувати різні сервіси та інструменти, які

пропонує ця платформа. Нижче подано загальний огляд технологій та можливої архітектури:

Google Cloud Storage (GCS). Використовуйте GCS для зберігання великих обсягів даних, таких як растр та структуровані дані.

Google Cloud Dataprep. Для очищення та передобробки даних можна використовувати Dataprep, який надає можливість візуальної підготовки даних безпосередньо на GCP.

Google Cloud Dataprep. Використовуйте Dataprep для автоматизованої підготовки та очищення даних.

Google Cloud Composer. Для оркестрації ваших завдань обробки даних використовуйте Composer, який базується на Apache Airflow.

Google Cloud Dataflow. Dataflow дозволяє обробляти та аналізувати великі потоки даних в режимі реального часу або пакетного режиму.

BigQuery. Використовуйте BigQuery для аналізу та вивчення ваших даних. Він добре підходить для SQL-запитів та аналітики.

Google Cloud Pub/Sub. Для роботи з потоковими даними можна використовувати Pub/Sub для отримання та надсилання повідомлень між різними частинами архітектурного стеку.

Google Cloud BigTable або Firestore. Для зберігання NoSQL-даних в режимі реального часу використовуйте BigTable або Firestore.

Google Cloud ML Engine. Якщо ваші завдання обробки даних пов'язані з машинним навчанням, використовуйте ML Engine для навчання та розгортання моделей.

Архітектуру можливо реалізувати наступним чином:

1. Збір та зберігання: Дані збираються та зберігаються в GCS або BigTable, залежно від їхнього типу та обсягу.
2. Очищення та передобробка: Dataprep використовується для попередньої обробки та очищення даних.

3. Оркестрація завдань: Composer використовується для планування та виконання різних завдань обробки даних.

4. Обробка потокових даних: Dataflow використовується для обробки потокових даних у режимі реального часу.

5. Зберігання та аналіз: Дані можуть бути збережені в BigQuery для подальшого аналізу та використані для створення звітів та візуалізації.

6. Машинне навчання: ML Engine використовується для навчання та розгортання моделей машинного навчання, якщо це потрібно.

Організація Big Data на базі технологій Google та Google Cloud Platform (GCP) включає використання ряду інструментів та сервісів для зберігання, обробки та аналізу великого обсягу даних. Ось можливий набір технологій та архітектура:

Google Cloud Storage (GCS). Використовуйте GCS для зберігання великих обсягів неструктурованих даних.

Google Cloud Bigtable або Cloud Firestore. Для зберігання NoSQL-даних або документ-орієнтованих даних.

Google Cloud Dataprep. Використовуйте Dataprep для підготовки та очищення даних перед завантаженням у систему обробки.

Google Cloud Pub/Sub. Для обміну потоковими даними між різними частинами архітектури.

Google Cloud Dataproc. Для обробки великого обсягу даних в режимі батча або реального часу використовуйте Dataproc, який використовує Apache Spark та Apache Hadoop.

Google Cloud Dataflow. Для обробки потокових даних в реальному часі, використовуйте Dataflow. Він може працювати з потоковими даними, такими як трансляції з Pub/Sub.

Google Cloud BigQuery. Використовуйте BigQuery для аналізу та візуалізації великих обсягів даних. BigQuery підтримує SQL-запити та дозволяє використовувати засоби штучного інтелекту для аналізу даних.

Google Cloud AI Platform. Якщо потрібно розробляти та впроваджувати моделі машинного навчання на ваших даних.

Архітектуру можливо реалізувати наступним чином:

1. Збір та зберігання: Дані збираються та зберігаються в GCS або Bigtable/Firestore в залежності від характеру даних.

2. Очищення та підготовка: Використовуйте Dataprep для передобробки та очищення даних перед завантаженням.

3. Обробка батч-даних та потокових даних: Dataproc для батч-даних та Dataflow для потокових даних.

4. Зберігання та аналіз: Дані можуть бути збережені в BigQuery для подальшого аналізу та використані для створення звітів та візуалізації.

5. Машинне навчання: AI Platform може використовуватися для створення та розгортання моделей машинного навчання на ваших даних.

6. Моніторинг та оптимізація: Забезпечте моніторинг та оптимізацію архітектури для ефективної роботи з великим обсягом даних.

Конкретна архітектура повинна відповідати вимогам конкретного випадку використання та особливостям даних. Звертається увага на безпеку, масштабованість та ефективність системи.

Екосистема Hadoop - це комплекс взаємопов'язаних технологій та інструментів, розроблений для обробки та аналізу великих обсягів даних у розподілених середовищах. Ця екосистема створена на основі відкритих стандартів та включає різноманітні компоненти, які сприяють збору, зберіганню, обробці та аналізу даних. Назва "Hadoop" походить від проекту Apache Hadoop, який є однією з основних складових цієї екосистеми. На Рисунку 1.13 зображена популярна реалізація екосистеми HADOOP.

Основні компоненти екосистеми Hadoop включають:

1. Hadoop Distributed File System (HDFS): Розподілена файлова система, призначена для зберігання великих обсягів даних на вузлах кластера.

2. MapReduce: Модель програмування та обчислювальний фреймворк для

розподіленої обробки великих даних.

3. Apache Hive: Система для роботи з даними у форматі табличних структур, що використовує мову запитів HiveQL, схожу на SQL.

4. Apache Pig: Високорівнева платформа для програмування MapReduce-завдань, яка спрощує процес написання скриптів.

5. Apache HBase: Розподілена, розширювана система керування базами даних, спроектована для роботи з великими обсягами даних у реальному часі.

6. Apache Spark: Швидкий та універсальний фреймворк обробки даних, який підтримує багато інструментів та бібліотек для аналізу та машинного навчання.

7. Apache Kafka: Платформа для обробки та передачі потокових даних у реальному часі.

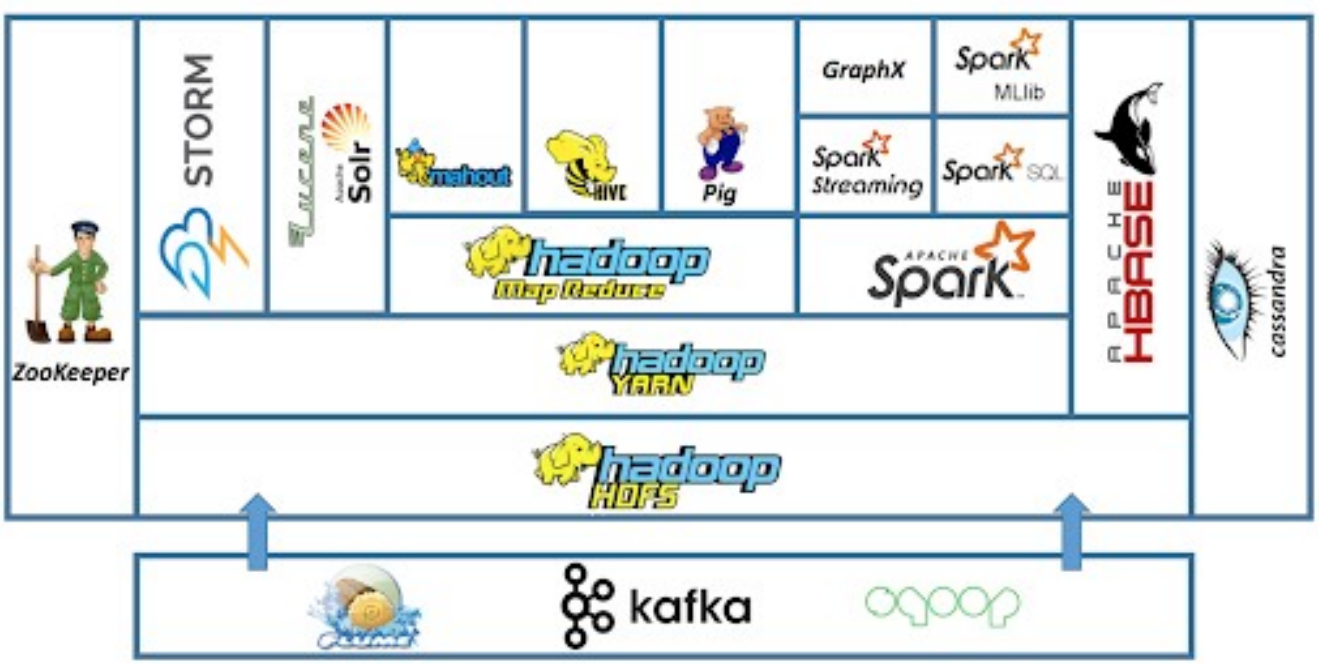

Рисунок 1.13 - Екосистема Hadoop

Це основні компоненти глибоко і тісно взаємодіють, будуючи досконалу систему для високоефективного зберігання, комплексної обробки та завдання глибокого аналізу величезних обсягів даних у вимогливих розподілених середовищах. Широка і деталізована екосистема Hadoop відчиняє широкі

горизонти перед підприємствами, надаючи їм змогу вирішувати викликові завдання, пов'язані з обробкою об'ємних даних, що виникають в різноманітних сферах інформаційної діяльності. Ця передова технологічна інфраструктура виявляється невід'ємною в різноманітних галузях, таких як фінанси, телекомунікації, охорона здоров'я та інші, надаючи не тільки необхідні інструменти, але і комплексні рішення для ефективної обробки та глибокого аналізу об'ємних наборів даних, що визначають сучасний бізнесовий пейзаж.

2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ DATA ENGINEERING, BIG DATA ТА CLOUD COMPUTING

2.1 Вибір технологічного стеку

У світі, де обсяги даних невпинно зростають, а потреба в їхньому аналізі та обробці стає все більш актуальною, створення розподіленої обчислювальної інфраструктури є ключовим завданням для підтримки ефективного управління інформацією. В рамках цього дослідження фокусуємося на розгляді можливостей використання технологій Data Engineering, Big Data та Cloud Computing для автоматизованої обробки та інтеграції даних.

В даному розділі роботи розглядається проблема вибору технологічного стеку, що визначає основні засоби та середовища, які будуть задіяні у реалізації розподіленої інфраструктури. Для цього розглядаються два основні сценарії: реалізація в хмарному середовищі з використанням технологій Google Cloud Platform та локальна реалізація, для якої обираються відповідні відкриті інструменти та фреймворки. Ці рішення мають на меті не лише забезпечити потужність та масштабованість обчислювальних процесів, але й забезпечити ефективність, надійність та гнучкість системи для оптимальної обробки та інтеграції великого обсягу різноманітних даних.

Обираючи технологічний стек, враховуємо різні аспекти, такі як продуктивність, можливості аналізу даних, масштабованість та інтеграцію з іншими системами. Результати цього вибору визначать ефективність розробленої системи, яка має на меті відповісти на виклики, пов'язані з обробкою та інтеграцією великого обсягу даних у розподіленому середовищі.

Data Engineering — це область, яка охоплює роботу з обробкою та підготовкою даних для аналізу та використання в бізнесі. Цей процес можна поділити на кілька етапів, і для кожного етапу можна використовувати різні технології. На Рисунку 2.1 зображено варіанти технологій розбитих на основні

етапи Data Engineering, представлено технології від AWS, GCP, Azure та Open Source.

Element	aws	Google Cloud	Azure	Open Source / 3rd Party
Data Collection	Cloud Watch Cloud Watch Logs Cloud Trail Config Custom agents / Scripts	Cloud Monitoring Cloud Logging Cloud Audit Logs Custom agents / Scripts	Azure Monitor Azure Activity Log Azure Policy Security Center Custom agents / Scripts	ZABBIX Prometheus fluentd logstash splunk> ELK telegraf Nagios Sensu
Data Storage	S3	Cloud Storage	Blob Storage	MINIO GLUSTER ceph
Data Analysis	CloudWatch Metrics Insights	Cloud Operations	Azure Monitor Metrics Explorer	Grafana kibana +ableau
Alerting	SNS	Cloud Monitoring Alerts	Azure Monitor Alerts	PagerDuty slack
Visualization	CloudWatch Dashboard QuickSight	Cloud Monitoring Dashboard Data Studio	Azure Monitor Dashboard Power BI	Grafana Superset Metabase redash +ableau
Reporting and Compliance	Config Rules Trusted Advisor	Security Command Center	Policy Compliance Security Center Compliance	OpenSCAP CISOfy
Automation	Lambda Step Functions	Cloud Functions	Azure Functions Azure Automation	Jenkins ANSIBLE
Integration	CloudFormation CodePipeline	Cloud Deployment Manager Cloud Build	Azure Automation Azure DevOps	Pulumi Terraform ANSIBLE Jenkins GitLab Travis CI
Feedback Loop	Well-Architected Tool	Well-Architected Framework	Well-Architected Framework	Scout APM Cloud Custodian

Рисунок 2.1 - Огляд варіативних технологій в етапах Data Engineering

Опис етапів Data Engineering:

1. Збір даних (Data Collection): використання стрімінгових платформ для ефективного та надійного збору та передачі даних з різних джерел. Застосування технологій, таких як Apache Kafka чи AWS Kinesis, для забезпечення потокового збору даних.

2. Зберігання даних (Data Storage): використання розподілених файлових систем або хмарних сховищ для ефективного та масштабованого зберігання великих обсягів даних. Використання технологій, таких як Hadoop Distributed File System (HDFS) чи Amazon S3.

3. Обробка даних (Data Processing): застосування технологій обробки великих даних, таких як Apache Spark або Apache Flink, для виконання різноманітних операцій над даними. Реалізація ETL (Extract, Transform, Load) процесів для підготовки даних до аналізу.

4. Аналіз даних (Data Analysis): використання аналітичних інструментів та мов програмування для виявлення закономірностей та отримання інсайтів з даних. Застосування алгоритмів машинного навчання для прогнозування та класифікації.

5. Оповіщення (Alerting): налаштування системи оповіщення для миттєвого сповіщення про події або стан даних, що вимагають уваги. Використання інструментів, таких як Prometheus, для моніторингу та оповіщення.

6. Візуалізація (Visualization): створення візуальних зображень та графіків для розуміння та представлення результатів аналізу даних. Використання інструментів, таких як Tableau чи Power BI, для візуалізації інформації.

7. Звітність та Дотримання (Reporting and Compliance): розробка систем звітності для представлення результатів аналізу даних, а також забезпечення відповідності стандартам та нормативам.

8. Автоматизація (Automation): впровадження автоматизованих процесів для забезпечення ефективності та регулярності виконання завдань Data Engineering. Використання інструментів автоматизації, таких як Apache Airflow чи Jenkins.

9. Інтеграція (Integration): забезпечення взаємодії між різними системами та компонентами для забезпечення єдиної інфраструктури обробки даних.

10. Цикл Зворотного Зв'язку (Feedback Loop): визначення та впровадження механізмів зворотного зв'язку для постійного вдосконалення процесів обробки даних на основі отриманих результатів та відгуків.

Етапи можуть відрізнятися в залежності від конкретних вимог та завдань

проекту, і вибір технологій буде залежати від специфіки роботи з даними в конкретному випадку.

2.2 Обчислювальні ресурси

У цьому підрозділі проводиться аналіз та вибір платформи для обчислювальних завдань. Для хмарної реалізації розглядається використання Google Compute Engine (GCE), зокрема його можливості віртуальних машин для обробки великого обсягу даних. Для локальної реалізації вивчаються можливості Apache Hadoop та Apache Spark як основних фреймворків для розподіленої обробки.

Обчислювальні ресурси. Обчислювальні ресурси визначають основний фундамент системи та впливають на її продуктивність та масштабованість. У виборі технологічного стеку для обчислювальних завдань розглядаються можливості як для хмарної, так і для локальної реалізації.

- Хмарна реалізація (Google Cloud Platform)

У хмарному середовищі обчислювальні ресурси визначаються з використанням Google Compute Engine (GCE). Цей сервіс надає гнучкі та масштабовані віртуальні машини, які можуть бути адаптовані до різних видів обчислювальних завдань. GCE дозволяє швидко розгортати та налаштовувати віртуальні машини з різними параметрами, такими як кількість процесорів, обсяг оперативної пам'яті та обсяг сховища даних.

- Локальна реалізація

У випадку локальної реалізації використовуються відкриті технології для розподіленої обробки даних. Apache Hadoop є ключовим інструментом у цьому контексті, де використовуються вузли для обчислювальних завдань. Його фреймворк MapReduce дозволяє ефективно розподіляти та виконувати обчислення на кластері вузлів.

Висновки. В обох випадках вибір обчислювальних ресурсів пов'язаний із

забезпеченням ефективності та масштабованості обчислень. У хмарній реалізації надається гнучкість та швидкість налаштувань завдяки віртуальним машинам, тоді як локальна реалізація забезпечує контроль та адаптацію під конкретні потреби системи. Обраний підхід залежить від вимог до ресурсів, готовності до використання хмарних послуг та інших факторів, які визначають оптимальність вибраного рішення.

Зберігання даних. Розглядається проблематика зберігання великих обсягів даних у хмарному середовищі та локальній інфраструктурі. Для хмарної реалізації використовується Google Cloud Storage (GCS) для управління неструктурованими даними та Bigtable для оптимізованої роботи з великомасштабними даними. Для локальної реалізації розглядається використання Hadoop Distributed File System (HDFS) та Apache Cassandra.

Вибір ефективної та масштабованої системи зберігання даних є ключовим етапом у реалізації розподіленої обчислювальної інфраструктури. У цьому підрозділі розглядається вибір технологічного стеку для забезпечення ефективного та надійного зберігання великих обсягів даних.

- Хмарна реалізація (Google Cloud Platform)

У хмарній реалізації ключовими компонентами для зберігання даних є Google Cloud Storage (GCS) та Bigtable. GCS надає масштабоване та безпечне сховище для неструктурованих даних, де дані розподілені по всьому світі для забезпечення доступності. Bigtable використовується для оптимізованої роботи з великомасштабними структурованими даними, забезпечуючи швидкий доступ та високу продуктивність.

- Локальна реалізація

У локальній реалізації використовується Apache Hadoop Distributed File System (HDFS) для забезпечення розподіленого зберігання даних. HDFS розподіляє та реплікує дані по вузлах кластера, що забезпечує високу доступність та стійкість до відмов.

Висновки. Обраний технологічний стек для зберігання даних повинен

враховувати характеристики самих даних та вимоги до швидкості доступу. У хмарній реалізації Google Cloud Platform надає широкий спектр інструментів для різноманітних потреб у зберіганні даних, в той час як локальна реалізація заснована на відкритих технологіях, забезпечуючи контроль і надійність. Вибір між цими варіантами визначається конкретними потребами проекту та його можливостями.

Обробка та аналіз даних. В даному підрозділі досліджується вибір інструментів для обробки та аналізу даних. Для хмарної реалізації звертається увага на Google Cloud Dataflow для потокової та пакетної обробки даних, а також BigQuery для аналітичних запитів. У локальній реалізації розглядається використання Apache Flink для потокової обробки та Apache Hive для виконання SQL-подібних запитів на великій кількості даних.

Ефективна обробка та аналіз великих обсягів даних є однією з ключових задач у реалізації розподіленої обчислювальної інфраструктури. У цьому підрозділі розглядаються технології, які використовуються для цих цілей.

- Хмарна реалізація (Google Cloud Platform)

У хмарній реалізації використовуються Google Cloud Dataflow та BigQuery для потокової та пакетної обробки даних відповідно. Google Cloud Dataflow надає зручний інтерфейс для створення потокових та пакетних обчислень, забезпечуючи автоматичне масштабування. BigQuery використовується для аналізу великих обсягів даних за допомогою SQL-подібних запитів.

- Локальна реалізація

У локальній реалізації для потокової обробки використовується Apache Flink, який дозволяє ефективно обробляти потокові дані у реальному часі. Для пакетної обробки використовується Apache Hive, який виконує SQL-подібні запити на великій кількості даних, розподілених по вузлах кластера.

Висновки. Обробка та аналіз даних визначається характером завдань та вимогами до швидкодії. У хмарній реалізації Google Cloud Platform забезпечує потужні та гнучкі інструменти для обчислення та аналізу даних, забезпечуючи

високу продуктивність. Локальна реалізація заснована на відкритих технологіях, що дозволяє реалізувати збалансовані та ефективні системи обробки даних в розподіленому середовищі. Вибір залежить від конкретних завдань та вимог проекту.

Інтеграція даних. Цей підрозділ присвячено аналізу та вибору механізмів інтеграції даних. У хмарній реалізації використовується Cloud Pub/Sub для системи обміну повідомленнями. У локальній реалізації розглядається використання Apache Kafka як системи потокової обробки та обміну даними між компонентами.

Ефективна інтеграція даних грає ключову роль у забезпеченні сполучення та обміну інформацією в розподіленій обчислювальній інфраструктурі. У цьому підрозділі розглядаються технології, які використовуються для забезпечення гнучкості та ефективності інтеграції даних.

- Хмарна реалізація (Google Cloud Platform)

У хмарній реалізації використовується Google Cloud Pub/Sub для побудови системи обміну повідомленнями. Цей сервіс дозволяє створювати потоки подій та передавати їх між компонентами системи в режимі реального часу. Такий підхід забезпечує асинхронний обмін даними між різними складовими інфраструктури.

- Локальна реалізація

У локальній реалізації для інтеграції даних використовується Apache Kafka. Цей інструмент служить системою потокової обробки та обміну даними між різними частинами системи. Apache Kafka гарантує надійність та стійкість до відмов, сприяючи ефективному обміну даними.

Висновки. Інтеграція даних є критично важливою для забезпечення сполучення та обміну інформацією. У хмарній реалізації Google Cloud Pub/Sub дозволяє побудувати надійну та ефективну систему обміну повідомленнями. У локальній реалізації Apache Kafka забезпечує потужний механізм для потокової обробки та обміну даними між різними компонентами інфраструктури. Вибір технології залежить від конкретних вимог до обміну даними та взаємодії між

складовими системи.

Додаткові технології. У цьому підрозділі розглядається вибір додаткових технологій, таких як TensorFlow для машинного навчання в хмарному середовищі, та Docker і Kubernetes для контейнеризації та оркестрації у локальній реалізації.

Вибір додаткових технологій визначає додаткові можливості та функціональність системи, такі як машинне навчання, контейнеризація та оркестрація.

- Хмарна реалізація (Google Cloud Platform)

У хмарній реалізації використовуються технології Google Cloud для розширення можливостей системи. TensorFlow використовується для реалізації машинного навчання та аналізу даних для виведення цінних інсайтів. Крім того, Docker і Kubernetes використовуються для контейнеризації та оркестрації, забезпечуючи гнучкість та масштабованість в управлінні контейнерами.

- Локальна реалізація

У локальній реалізації можна використовувати відкриті технології для аналогічних завдань. Наприклад, для машинного навчання використовується бібліотека scikit-learn, а Docker і Kubernetes можна використовувати для контейнеризації та оркестрації локальних компонентів системи.

Висновки. Впровадження додаткових технологій дозволяє розширити функціональність та можливості системи. У хмарній реалізації використовуються спеціалізовані сервіси для машинного навчання та оркестрації контейнерів. У локальній реалізації можна використовувати відкриті технології для досягнення подібних цілей, забезпечуючи при цьому гнучкість та контроль над реалізацією. Вибір технологій залежить від конкретних вимог та стратегії розробки проекту.

2.3 Опис архітектури розподіленої обчислювальної інфраструктури

Архітектурний підхід. Цей підрозділ визначає загальний архітектурний підхід до реалізації розподіленої інфраструктури. Розглядаються концепції мікросервісної архітектури та шарової моделі для забезпечення гнучкості та масштабованості.

Для успішного впровадження розподіленої обчислювальної інфраструктури визначається архітектурний підхід, який враховує організацію компонентів та їх взаємодію в системі.

Мікросервісна архітектура. Обрана архітектурна парадигма — мікросервісна архітектура. Система розбивається на невеликі та незалежні мікросервіси, кожен із яких виконує конкретні функції. Це сприяє гнучкості, масштабованості та простоті управління окремими компонентами системи. Мікросервіси комунікують між собою за допомогою стандартизованих протоколів, таких як HTTP або message queues.

Шарова модель. Архітектурна система реалізується за принципом шарової моделі. Вона розділяється на різні функціональні шари, такі як інтерфейс користувача, логічний шар бізнес-логіки та шар доступу до даних. Кожен шар має визначені інтерфейси, що спрощує розробку та тестування, а також дозволяє змінювати частини системи незалежно одна від одної.

Висновки. Обрана мікросервісна архітектура та шарова модель дозволяють створити гнучку та легко масштабовану систему. Мікросервіси сприяють незалежності компонентів, а шарова модель визначає чіткі інтерфейси та структуру взаємодії між шарами. Цей архітектурний підхід забезпечує зручність управління та розвитку системи, покращуючи її ефективність та масштабованість.

Забезпечення масштабованості та високої доступності. В даному підрозділі обговорюються стратегії та інструменти для забезпечення масштабованості та високої доступності в розподіленій інфраструктурі, враховуючи як особливості хмарних технологій, так і локальної реалізації.

Для забезпечення ефективності системи та відповіді на зростаюче навантаження важливим є впровадження стратегій масштабування та забезпечення високої доступності.

Масштабованість. Використання горизонтального масштабування дозволяє додавати нові ресурси або сервери для розподілу навантаження та підтримки збільшеного обсягу даних чи транзакцій. Мікросервісна архітектура сприяє легкості масштабування окремих компонентів, забезпечуючи гнучкість у виборі ресурсів для кожного сервісу.

Висока доступність. Для забезпечення високої доступності системи використовується реплікація даних та балансування навантаження. Дані реплікуються між різними вузлами для запобігання втраті даних при відмові одного з вузлів. Балансер навантаження розподіляє запити між різними серверами, що також сприяє високій доступності та уникненню однопунктової невдачі.

Висновки. Стратегії масштабування та забезпечення високої доступності визначають стійкість та продуктивність системи. Горизонтальне масштабування у поєднанні з мікросервісною архітектурою дозволяє системі ефективно адаптуватися до зростання обсягу роботи. Реплікація даних та балансування навантаження забезпечують надійність та високу доступність сервісів, зменшуючи ймовірність виникнення відмов. Це гарантує, що система залишатиметься доступною та витривалою в умовах зростаючого обсягу користувачів та завдань.

Забезпечення безпеки даних. У цьому підрозділі визначається стратегія забезпечення безпеки даних, включаючи шифрування, автентифікацію та авторизацію, яка використовується як для хмарної, так і для локальної реалізації.

Забезпечення безпеки даних є критичним елементом розподіленої обчислювальної інфраструктури, і вимагає комплексного підходу для захисту конфіденційності, цілісності та доступності даних.

Шифрування даних. Застосування шифрування для даних у спокої та під час

передачі є основною стратегією забезпечення конфіденційності. Використання алгоритмів шифрування на різних рівнях, таких як на рівні бази даних, забезпечує захист від несанкціонованого доступу до чутливої інформації.

Автентифікація та авторизація. Механізми автентифікації та авторизації використовуються для контролю доступу до різних компонентів системи. Засоби, такі як многорівнева авторизація та використання токенів, забезпечують ідентифікацію та визначення прав доступу користувачів.

Моніторинг та аудит безпеки. Системи моніторингу та аудиту безпеки дозволяють вчасно виявляти та реагувати на потенційні загрози. Це включає в себе реєстрацію та аналіз подій, виявлення аномалій та реагування на інциденти безпеки.

Висновки. Забезпечення безпеки даних — це неперервний процес, який включає в себе використання шифрування, правильні стратегії автентифікації та авторизації, а також системи моніторингу та аудиту. Це необхідно для забезпечення конфіденційності, цілісності та доступності даних в умовах зростаючого ризику кіберзагроз та викликів безпеки в розподіленому середовищі.

Оркестрація та управління. Розглядається питання оркестрації та управління компонентами системи. Визначається використання інструментів, таких як Google Cloud Composer у хмарному середовищі та Kubernetes для локальної реалізації.

Оркестрація та управління розподіленою обчислювальною інфраструктурою є ключовим елементом для забезпечення ефективності та гнучкості в умовах зростаючого обсягу завдань та компонентів.

Оркестрація контейнерів. Використання систем оркестрації контейнерів, таких як Kubernetes, дозволяє автоматизувати розгортання, масштабування та управління контейнеризованими додатками. Kubernetes забезпечує автоматичну оркестрацію та керування життєвим циклом контейнерів, спрощуючи розгортання та забезпечуючи стабільність в роботі системи.

Управління конфігурацією. Використання систем управління конфігурацією, таких як Ansible або Puppet, дозволяє автоматизувати

налаштування та розгортання компонентів системи. Це забезпечує консистентність та стабільність у роботі всіх елементів інфраструктури.

Моніторинг та логування. Системи моніторингу та логування, такі як Prometheus чи ELK Stack, дозволяють вчасно виявляти проблеми та проводити аналіз подій. Вони забезпечують візуалізацію стану системи, допомагаючи у вирішенні проблем та оптимізації продуктивності.

Висновки. Оркестрація контейнерів, управління конфігурацією та системи моніторингу та логування є важливими компонентами для ефективного управління та підтримки розподіленої інфраструктури. Вони забезпечують автоматизацію, стабільність та можливість швидкого реагування на зміни та проблеми, що дозволяє підтримувати систему в оптимальному стані та забезпечувати її надійність.

3 ДОСЛІДЖЕННЯ РЕАЛІЗАЦІЇ РОЗПОДІЛЕНОЇ ОБЧИСЛЮВАЛЬНОЇ ІНФРАСТРУКТУРИ

3.1 Реалізація розподіленої обчислювальної інфраструктури локально

Реалізація розподіленої обчислювальної інфраструктури локально вимагає використання різних технологій та компонентів для створення ефективного та масштабованого середовища обчислень. Нижче наведено загальну схему та технології, які можна використовувати:

Контейнеризація. Використання контейнерів дозволяє ізолювати обчислювальні середовища та забезпечити консистентність робочих середовищ на різних машинах. Технології: Docker або Podman.

Оркестрація контейнерів. Оркестрація контейнерів дозволяє автоматизувати розгортання, масштабування та управління контейнерами. Технології: Kubernetes або Docker Swarm.

Оркестрація робочого процесу або ж workflow. Оркестрація робочого процесу - це процес управління та координації виконання різних компонентів або етапів великого робочого завдання. У сфері програмування, аналізу даних та інших областях це може означати автоматизоване керування послідовністю дій або задач, які повинні виконуватися для досягнення конкретної мети. Оркестрація робочого процесу стає особливо важливою в умовах складних та розподілених систем. Технології: Apache Airflow, Luigi

Мікросервісна архітектура. Розробка додатків у вигляді мікросервісів для декомпозиції функціональності та полегшення розгортання.

Системи розподілених обчислень. Використання систем розподілених обчислень для обробки великих обсягів даних та виконання складних обчислень паралельно. Технології: Apache Spark, Apache Flink.

Брокери повідомлень. Забезпечення асинхронного обміну повідомленнями між різними компонентами системи для підтримки взаємодії. Технології: Apache Kafka, RabbitMQ.

Моніторинг та логування: Використання інструментів для моніторингу та логування дозволяє відслідковувати та аналізувати роботу розподіленої системи. Технології: Prometheus, ELK Stack (Elasticsearch, Logstash, Kibana).

Бази даних. Використання баз даних для зберігання та управління даними у розподіленому середовищі. Технології: MySQL, PostgreSQL, MongoDB (залежно від потреб).

Хмарні сервіси. Інтеграція з хмарними сервісами для розширення ресурсів та забезпечення високої доступності. Технології: AWS, Azure, Google Cloud.

Схема представляє загальний підхід та може змінюватися в залежності від конкретних вимог та характеристик проекту.

Відповідно до цих компонентів побудована абстрактна функціональна схема для локальної реалізації обчислень, представлена на Рисунку 3.1.

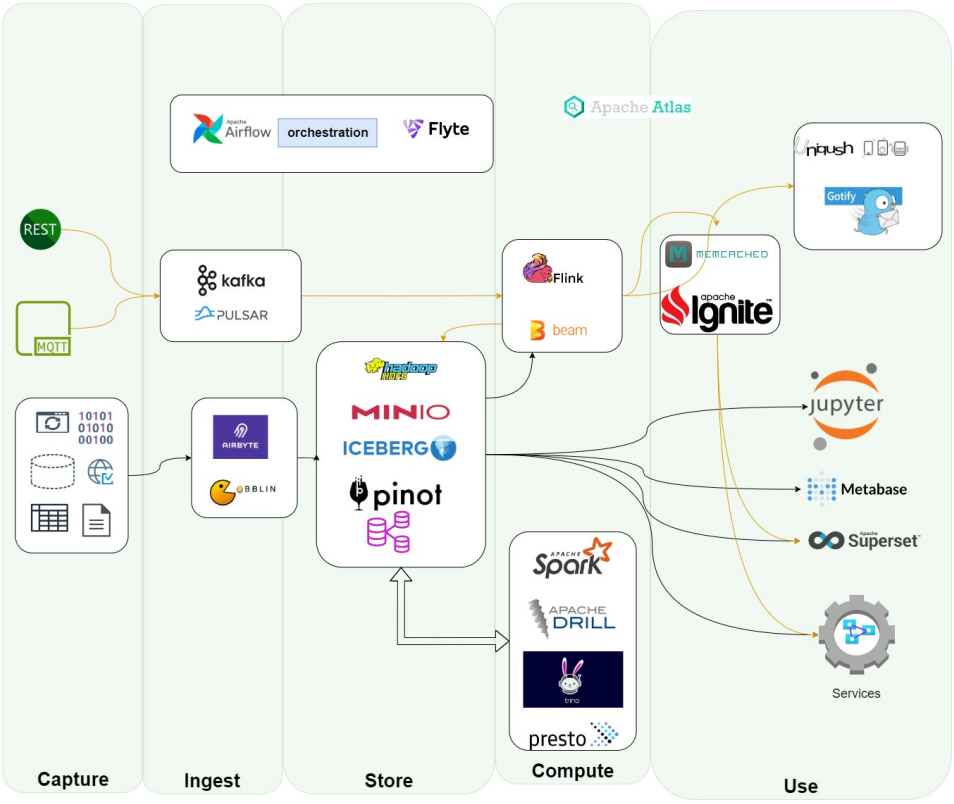

Рисунок 3.1 - Архітектура локальної обчислювальної мережі

3.2 Реалізація розподіленої обчислювальної інфраструктури хмарно

Реалізація розподіленої обчислювальної інфраструктури в хмарі може бути структурована на абстрактних п'ять основних етапів для ефективної обробки потоку даних:

1. Захоплення (Capture). На цьому етапі дані збираються від різних джерел. Це може включати в себе дані з сенсорів, журналів, додатків, веб-сайтів та інших джерел. Для забезпечення максимальної ефективності цей етап може бути налаштований для автоматичного захоплення даних в реальному часі (streaming) або пакетного режиму.

2. Поглинання (Ingest). На цьому етапі дані, зібрані на етапі захоплення, пересилаються в централізоване сховище даних. Хмарні сервіси, такі як Amazon Kinesis, Apache Kafka, або Azure Event Hubs, можуть використовуватися для потокового вводу даних в систему. Це забезпечує стійкість та гарантує високу швидкість передачі.

3. Збереження (Store). На цьому етапі дані зберігаються в хмарному сховищі даних. Amazon S3, Azure Data Lake Storage, або Google Cloud Storage можуть служити для ефективного збереження великих обсягів даних. Забезпечується висока масштабованість та доступність.

4. Обчислення (Compute). На цьому етапі відбувається обробка даних. Використовуючи хмарні сервіси обчислення, такі як Amazon EC2, Azure Virtual Machines або Google Cloud Compute Engine, можна створювати та масштабувати обчислювальні потужності в залежності від обсягу роботи.

5. Використання (Use). Цей етап включає в себе аналіз та використання оброблених даних. Відбувається вивчення та витягнення цінних знань із даних. Для візуалізації, статистичного аналізу та машинного навчання можна використовувати сервіси, такі як Amazon SageMaker, Azure Machine Learning або Google AI Platform.

Цей підхід дозволяє забезпечити повний цикл обробки даних в хмарному середовищі, забезпечуючи гнучкість, масштабованість та високу доступність для всього процесу.

Розглянемо побудування зазначеної п'ятирівневої архітектури з базуванням на різних платформах: AWS, Azure, GCP. Всі представлені платформи являються провідними постачальник хмарних послуг та надають широкий спектр інструментів та сервісів, що сприяють ефективній обробці та управлінню даними. Архітектура з базуванням на AWS представлена на Рисунку 3.2.

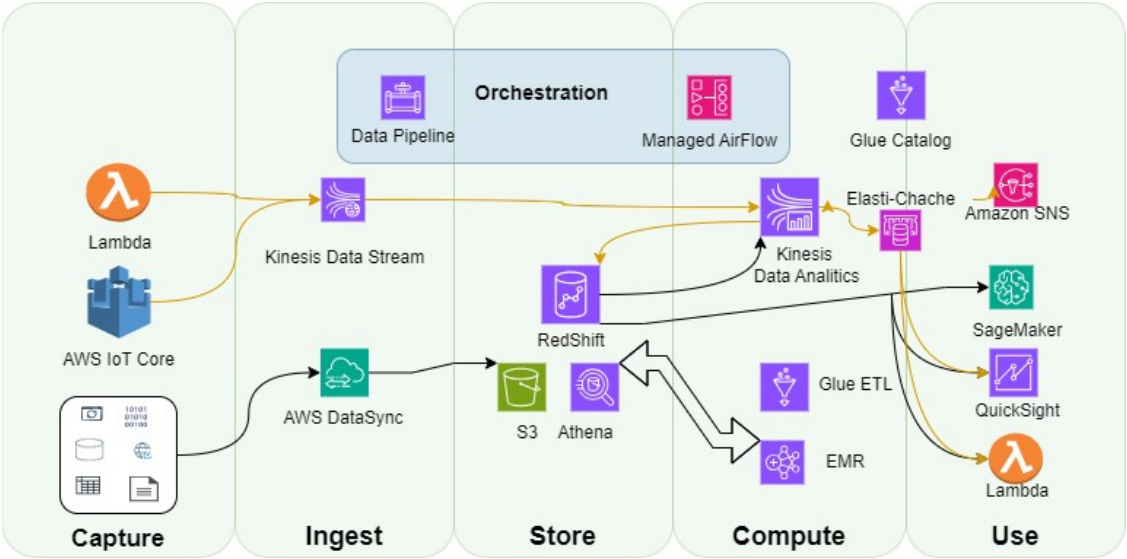

Рисунок 3.2 - Архітектура на базі AWS

Реалізації розподіленої обчислювальної інфраструктури на AWS на різних етапах:

1. Захоплення та Введення Даних (Capture & Ingest):

Amazon Kinesis Data Streams використовується для обробки великого потоку подій чи даних в режимі реального часу. Це може бути використано для аналізу логів, вимірювань IoT, аналітики в реальному часі та інших сценаріїв, де важлива обробка даних у режимі реального часу.

AWS IoT Core використовується для підключення та керування великою кількістю IoT-пристроїв, а також для обміну даними між цими пристроями та хмарною платформою. Вона забезпечує безпечну та масштабовану

інфраструктуру для розробки та управління системами Інтернету речей.

AWS DataSync використовується для автоматизації переміщення та синхронізації даних між різними репозитаріями, забезпечуючи ефективний обмін даними та зменшуючи час та зусилля, потрібні для таких операцій.

2. Збереження Даних (Store):

Amazon Redshift використовується для зберігання та аналізу великих обсягів даних, зазвичай в аналітичних додатках. Вона дозволяє виконувати складні SQL-запити та агрегації над великими наборами даних, що забезпечує швидкий доступ до результатів аналізу.

Amazon S3 використовується для зберігання різноманітних типів даних, включаючи файли, зображення, відео, архіви та інші об'єкти. Це часто використовується як основне сховище для даних в хмарних додатках та аналітичних проектах.

Amazon Athena використовується для аналізу даних, які зберігаються в Amazon S3, шляхом виконання SQL-запитів без необхідності власного створення та управління базою даних. Це дозволяє швидко та зручно аналізувати дані, не завантажуючи їх у окрему базу даних.

3. Обчислення (Compute):

Amazon Kinesis Data Analytics дозволяє виконувати SQL-запити на потокових даних та створювати аналітичні додатки. Використовується для реального аналізу та витягування цінної інформації з потокових даних.

Amazon EMR використовується для обчислення та аналізу великих обсягів даних, зазвичай застосовується в області обробки Big Data, машинного навчання, аналізу логів та інших обчислювально витратних завдань.

AWS Glue ETL використовується для автоматизації етапів вилучення, трансформації та завантаження даних, щоб забезпечити оптимальну підготовку даних для подальшого аналізу, зазвичай в системах Business Intelligence (BI) або в аналітичних платформах.

Amazon ElastiCache використовується для прискорення доступу до даних,

зберіганих у базі даних або інших джерелах, шляхом зберігання результатів часто використовуваних запитів у пам'яті. Це дозволяє покращити продуктивність та знизити затрати на ресурси для доступу до даних. ElastiCache підтримує кешування за допомогою популярних систем, таких як Redis та Memcached.

4. Використання (Use):

Amazon SNS використовується для розсилання повідомлень (SMS, email, HTTP, itd.) до різних отримувачів (кінцевих користувачів, мобільних пристроїв, електронних адрес, інших AWS-сервісів) в реальному часі. Це може бути використано для різних сценаріїв, таких як сповіщення користувачів, моніторинг або синхронізація подій.

Amazon SageMaker використовується для розробки, тренування та розгортання моделей машинного навчання безпосередньо в хмарі. Це включає в себе створення моделей, оптимізацію гіперпараметрів, тренування та впровадження моделей у виробництво.

Amazon QuickSight використовується для створення швидких та інтерактивних візуалізацій з даних, зокрема для дашбордів та звітів. Вона підтримує підключення до різних джерел даних, таких як Amazon S3, Amazon Redshift, Amazon RDS, інші бази даних та сервіси.

Архітектура з базуванням на Azure представлена на Рисунку 3.3.

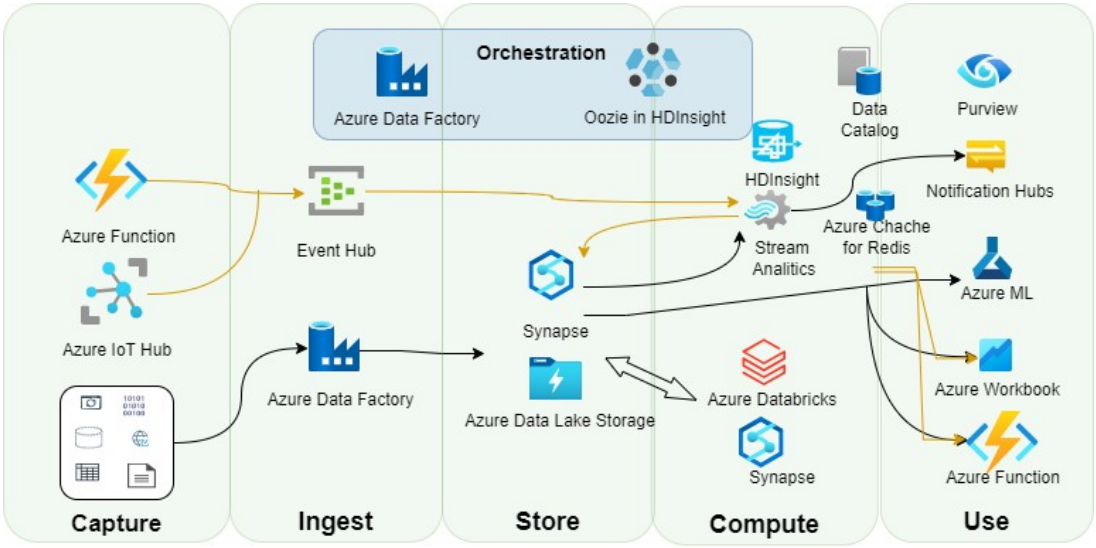

Рисунок 3.3 - Архітектура на базі Azure

Реалізація розподіленої обчислювальної інфраструктури на Azure та GCP на різних етапах:

1. Захоплення та Введення Даних (Capture & Ingest):

Azure Functions використовується для написання та виконання коду відповідно до подій, таких як тригери таймера, зміни в хмарних сховищах, HTTP-запити та інші події. Це дозволяє автоматизувати обробку подій без необхідності управління інфраструктурою.

Azure IoT Hub використовується для обміну даними між IoT-пристроями та хмарною інфраструктурою. Він надає механізми реєстрації пристроїв, керування доступом, моніторингу та реалізації комунікації за допомогою протоколів, таких як MQTT, AMQP або HTTP. Azure IoT Hub грає ключову роль в розробці та управлінні розподіленими системами IoT.

Azure Data Factory використовується для оркестрації та автоматизації завдань обробки та переміщення даних з одного місця в інше в хмарному середовищі. Вона надає можливості для інтеграції з різними джерелами даних та виконання комплексних завдань ETL (Extract, Transform, Load).

Azure Event Hub використовується для збору та обробки великого потоку подій в реальному часі. Вона надає можливості для прийому, зберігання та аналізу потокових даних від різних джерел, таких як датчики, додатки або

пристрої IoT. Це дозволяє розробникам будувати аналітичні застосунки та системи обробки потокових даних в середовищі Azure.

2. Збереження Даних (Store):

Azure Data Lake Storage використовується для зберігання та аналізу великих обсягів структурованих та неструктурованих даних. Воно підтримує роботу з данними у реальному часі та є частиною облачної аналітичної екосистеми Azure, дозволяючи розробникам та аналітикам ефективно обробляти та вивчати дані для прийняття рішень.

Azure Synapse використовується для оброблення та аналізу великих обсягів даних в реальному часі. Вона надає можливості для об'єднання даних з різних джерел, виконання аналітичних запитань та створення звітів.

3. Обчислення (Compute):

Azure HDInsight використовується для роботи з різноманітними завданнями Big Data, включаючи обробку та аналіз великих обсягів даних, машинне навчання, аналіз логів та інші сценарії.

Azure Stream Analytics використовується для обробки та аналізу великого потоку подій в реальному часі. Вона надає можливості для застосування SQL-подібних запитів до потокових даних, фільтрації та трансформації даних для витягування цінної інформації з потокових джерел.

Azure Databricks використовується для розробки, тренування та впровадження моделей машинного навчання, а також для виконання аналітики та обробки великих обсягів даних. Він дозволяє аналітикам та даним науковцям працювати в спільному середовищі, об'єднуючи екосистему Apache Spark з можливостями Azure.

4. Використання (Use):

Azure Purview використовується для створення централізованого каталогу даних, який дозволяє організаціям відстежувати, розуміти та керувати своїми даними. Вона надає можливості для пошуку та оцінки даних, автоматичної класифікації та забезпечення дотримання вимог до конфіденційності та безпеки.

Notification Hubs використовується для створення та управління розсиланнями повідомлень (повідомлення push) на мобільні пристрої, планшети, комп'ютери та інші пристрої. Це дозволяє розробникам створювати ефективні механізми сповіщення для своїх додатків.

Azure ML використовується для побудови та експериментування з моделями машинного навчання, тренування моделей за допомогою різноманітних алгоритмів, а також для розгортання моделей у виробництво. Вона надає інструменти для автоматизації процесів машинного навчання та аналізу результатів.

Azure Workbook використовується для створення інтерактивних дашбордів та звітів, які дозволяють аналізувати дані, стежити за ключовими метриками та взаємодіяти з інформацією в хмарному середовищі Azure.

Архітектура з базуванням на GCP представлена на Рисунку 3.4.

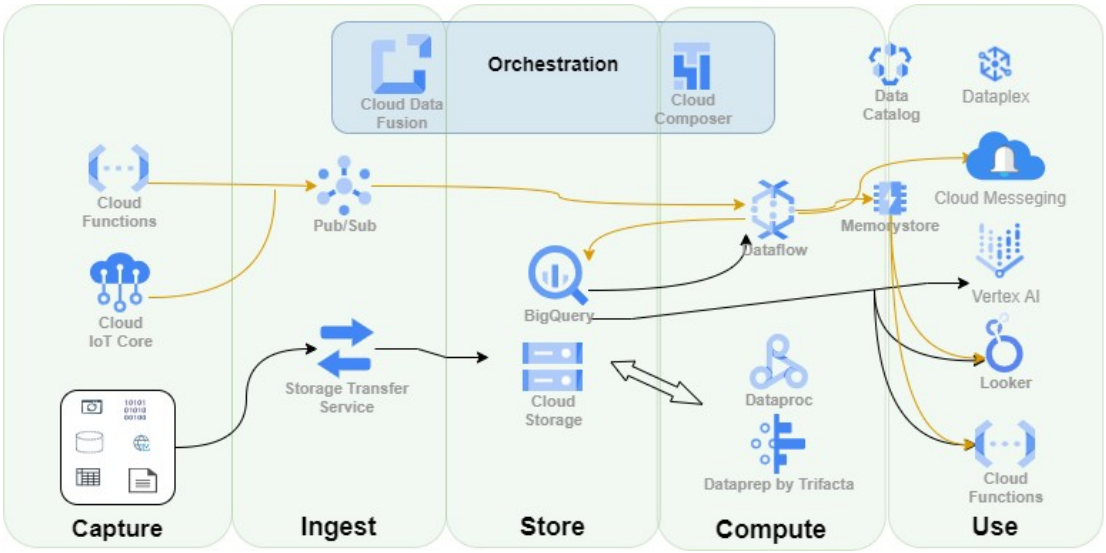

Рисунок 3.4 - Архітектура на базі GCP

Реалізація розподіленої обчислювальної інфраструктури на GCP на різних етапах:

1. Захоплення та Введення Даних (Capture & Ingest):

Google Cloud Functions дозволяє розробникам виконувати функції (код) в відповідь на події без необхідності керування інфраструктурою. Вона забезпечує

миттєву масштабованість та виконання коду без необхідності підтримки серверів.

Google Cloud IoT використовується для підключення та керування мільйонами IoT-пристроїв, збору та обробки потокових даних з цих пристроїв, а також для інтеграції з іншими хмарними сервісами для аналізу та вивчення даних.

Google Cloud Pub/Sub використовується для обміну потоковими даними між різними компонентами системи. Вона надає масштабовану та надійну інфраструктуру для передачі повідомлень між різними складовими додатків чи систем.

Google Cloud Storage Transfer Service використовується для автоматизованого керування та переміщення великих обсягів даних між різними сховищами, дозволяючи ефективно обробляти задачі резервного копіювання, архівації та синхронізації даних.

2. Збереження Даних (Store):

BigQuery використовується для виконання аналітичних запитань та вивчення великих обсягів даних у реальному часі. Вона підтримує SQL-запити та дозволяє розробникам та аналітикам взаємодіяти з великими даними без необхідності управління інфраструктурою бази даних.

Cloud Storage використовується для зберігання різноманітних типів даних, таких як файли, зображення, відео та інші об'єкти. Вона надає масштабоване та надійне сховище для розробки та розгортання додатків, аналізу даних та забезпечення доступу до даних в режимі реального часу.

3. Обчислення (Compute):

Dataprep використовується для ефективної обробки та підготовки даних для аналітичних завдань. Він надає інтерфейс з графічним користувацьким інтерфейсом, що дозволяє користувачам легко визначати та виконувати операції очищення, перетворення та стандартизації даних.

Dataproc використовується для розгортання та управління кластерами Spark та Hadoop в хмарному середовищі. Вона дозволяє виконувати розподілені обчислення та аналіз великих обсягів даних у хмарному середовищі.

Dataflow використовується для створення та виконання потокових та пакетних обчислень, таких як ETL-процеси, обробка потокових даних та інші сценарії аналізу та обробки даних. Вона підтримує роботу з Apache Beam та надає можливості для розподіленого обчислення в хмарному середовищі.

Google Cloud MemoryStore фулі-управліюча служба хмарного кешування в Google Cloud Platform (GCP). Вона використовує технології Redis та Memcached для забезпечення швидкого та надійного кешування даних в пам'яті.

4. Використання (Use):

Looker дозволяє аналізувати та візуалізувати дані з різних джерел у вигляді зручних звітів та графіків. Вона допомагає бізнес-користувачам отримувати цінні інсайти з даних, а також взаємодіяти з ними через інтерактивні панелі та звіти.

Vertex AI надає інструменти для побудови, тренування та розгортання моделей машинного навчання. Вона допомагає розробникам та даним науковцям прискорювати процеси машинного навчання та створювати ефективні моделі для різних сценаріїв.

Cloud Messaging використовується для розсилання повідомлень (повідомлень push) на мобільні пристрої, планшети та комп'ютери. Це дозволяє розробникам створювати ефективні механізми сповіщення для своїх додатків та сервісів. Повідомлення можуть використовуватися для сповіщення користувачів про нові події, оновлення та інші важливі інформаційні повідомлення.

3.3 Проведення аналітики

Реалізація розподіленої обчислювальної мережі може бути реалізована як локально, так і в хмарному середовищі. Обидва підходи мають свої переваги та недоліки, які важливо враховувати при виборі оптимального рішення. Розглянемо плюси та мінуси кожного з варіантів:

Локальна реалізація. Переваги заключаються в:

- Локальний контроль:

- Компанія має повний контроль над обчислювальною інфраструктурою, що дозволяє налаштовувати систему під конкретні вимоги та процеси.
- Забезпечує високий рівень конфіденційності, оскільки дані залишаються власній області.

- Локальний доступ:
  - Забезпечує високу швидкість доступу до ресурсів і даних завдяки локальній мережі.
- Вартість управління:
  - Може бути меншою з точки зору загальних витрат на управління та обслуговування.

Локальна реалізація. До мінусів можливо зазначити:

- Обмежені ресурси:
  - Обмежена масштабованість та обчислювальні ресурси, оскільки обсяги даних та завдань можуть перевищувати локальні можливості.
- Низька гнучкість:
  - Важко масштабувати та адаптувати інфраструктуру до змін потреб.
- Апаратні витрати:
  - Локальна інфраструктура може вимагати значних витрат на апаратне забезпечення та обслуговування, що може бути вище порівняно з хмарними альтернативами.
- Відмовостійкість:
  - Обмежена можливість забезпечити відмовостійкість порівняно з хмарними рішеннями, особливо якщо необхідно забезпечити резервне копіювання та відновлення даних.

Плюси хмарної реалізації:

- Масштабованість:

- Забезпечує високий рівень масштабованості, дозволяючи легко збільшувати або зменшувати обчислювальні ресурси залежно від потреб.

- Гнучкість:
  - Висока гнучкість та зручність управління, оскільки хмарні послуги легко адаптуються до змін у вимогах.
- Доступність:
  - Забезпечує високий рівень доступності завдяки розподіленій природі хмарної інфраструктури.
- Витрати за фактом використання:
  - Хмарні послуги зазвичай пропонують модель витрат за фактом використання, що дозволяє оптимізувати витрати в залежності від реального обсягу роботи.
- Інновації та оновлення:
  - Хмарні платформи регулярно оновлюються, надаючи доступ до нових технологій та інновацій, що може бути важливим для розвитку бізнесу.
- Легкість інтеграції:
  - Хмарні рішення часто забезпечують гнучкі механізми для інтеграції з різними іншими хмарними службами та іншими інфраструктурами.
- Географічна гнучкість:
  - Можливість розгортання хмарних рішень в різних регіонах світу для забезпечення високої доступності та зменшення латентності.

Мінуси хмарної реалізації:

- Залежність від Інтернету:
  - Робота в хмарному середовищі може бути обмеженою в разі відсутності або обмеженого доступу до Інтернету.
- Безпека:
  - Може існувати більші турботи щодо безпеки даних у хмарі порівняно з локальною реалізацією.

Обираючи між локальною та хмарною реалізацією, важливо враховувати конкретні вимоги, бюджет і стратегію компанії, а також забезпечити баланс між контролем, доступністю та масштабованістю. В Таблиці 3.1 наведено ключові аспекти різниці реалізацій.

Таблиця 3.1 - Порівняння локальної та хмарної реалізації

Аспект	**Локальна реалізація**	**Хмарна реалізація**
Витрати на обладнання	Значні апаратні витрати, потреба у великих капіталовкладеннях	Модель витрат за фактом використання, можливість оптимізації витрат
Доступність	Обмежена можливість забезпечити відмовостійкість	Забезпечення високої доступності та резервного копіювання даних
Оновлення обладнання	Витрати та складнощі з оновленням, потреба у великих трудовитратах	Регулярні оновлення та доступ до нових технологій, автоматичне оновлення
Гнучкість інтеграції	Обмежена гнучкість у виборі інтеграційних рішень, потреба у великих зусиллях для інтеграції	Гнучкі механізми для інтеграції з іншими хмарними та зовнішніми сервісами, готові API
Географічна гнучкість	Обмежена можливість розгортання на різних географічних областях	Географічна гнучкість для високої доступності та зменшення латентності
Безпека	Потреба у великих зусиллях для забезпечення високого рівня безпеки, обмежені можливості у використанні передових захисних технологій	Широкі можливості для застосування передових інструментів безпеки та використання служб безпеки хмарного провайдера
Масштабованість	Обмежена масштабованість, потреба у великих зусиллях для розширення	Висока масштабованість, можливість автоматичного розширення ресурсів
Управління витратами	Складно контролювати і передбачити витрати	Гнучкий контроль витрат, можливість моніторингу та оптимізації
Запуск проектів	Великий час на підготовку і запуск проектів, бюрократія	Швидкий старт проектів, готові шаблони та сервіси для автоматизації
Робота з даними	Обмежена пропускна здатність для обробки великих обсягів даних	Велика пропускна здатність, можливість обробки великих обсягів даних

Висновки щодо розглянутих хмарних платформ виявляють багатоаспектну природу їхніх переваг та можливостей:

- AWS:
  - Постачає широкий спектр інструментів, особливо сильний в галузі стрімів та Інтернету речей, надаючи величезні можливості для інновацій та розвитку новаторських рішень.
  - Ефективна інтеграція з масивними сховищами даних, дозволяючи легко маніпулювати та аналізувати великі обсяги інформації.
  - Гнучка шкала ресурсів, що дозволяє легко адаптуватися до змін обсягу роботи та оптимізувати витрати.
  - Забезпечує високу надійність та безпеку для зберігання та обробки конфіденційної інформації.
- Azure:
  - Має глибокий інтегрований стек, що забезпечує ефективну роботу з даними на всіх етапах їхньої обробки, від збору до аналізу.
  - Надає спеціалізовані сервіси для кожної фази обробки даних, забезпечуючи комплексність та повноту підходу до їхнього управління.
  - Відмінна інтеграція з іншими продуктами Microsoft, спрощуючи розгортання та управління комплексними інфраструктурними рішеннями.
  - Широкий вибір сертифікованих служб та висока ступінь дотримання стандартів безпеки.
- GCP:
  - Вирізняється високою продуктивністю та швидкістю обчислень, що робить її ідеальним вибором для завдань, які вимагають великої обчислювальної потужності.
  - Передові сервіси для аналітики та машинного навчання, що надають додаткові можливості для розуміння та використання даних.
  - Широкі можливості автоматизації та використання інструментів штучного інтелекту для вдосконалення ефективності роботи з даними.
  - Пропонує прозорий та простий механізм ціноутворення, що дозволяє ефективно керувати витратами та планувати бюджет.

Кожен хмарний провайдер має свої переваги, та вибір залежить від конкретних потреб та вимог проекту.

ВИСНОВКИ

1. Обробка та інтеграція даних у великому обсязі – важливе завдання. Технології Data Engineering, Big Data та Cloud Computing стали стандартом для автоматизованої обробки даних. Оптимальні технологій для розподіленої інфраструктури покращують ефективність організацій до управління інформацією. Обробка та інтеграція даних включають ETL та ELT процеси, а основні принципи OLTP та OLAP допомагають в роботі з транзакціями та аналітикою. Використання технологій Data Stack розвивається разом із збільшенням обсягу та різноманітності даних, що створює нові можливості для ефективного використання.

2. При виборі технологічного стеку ретельно враховується різноманітні аспекти, такі як продуктивність, можливості аналізу даних, масштабованість та інтеграцію з іншими системами. Результати цього вибору будуть визначати ефективність розробленої системи, яка має за мету відповісти на виклики, пов'язані з обробкою та інтеграцією великого обсягу даних у розподіленому середовищі. В цьому контексті, пріоритетним завданням є створення інфраструктури, яка не лише відповідає поточним вимогам, але й забезпечує гнучкість та готовність до майбутніх викликів в області обробки та аналізу даних.

3. Локальна інфраструктура забезпечує внутрішню ефективність та контроль, тоді як хмарне розподілене обчислення пропонує гнучкість, масштабованість та високу доступність. Обидва підходи мають унікальні переваги та виклики: локальна реалізація - більший контроль, але інфраструктурні витрати; хмарне розподілене обчислення - гнучкість та швидкість відгуку, але вимагає уважного управління безпекою та вибору оптимальних хмарних послуг. Ефективне впровадження розподіленої обчислювальної інфраструктури дозволяє покращити обробку даних, оптимізувати ресурси та підвищити продуктивність організації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Fundamentals of Data Engineering by Joe Reis, Matt Housley Released June 2022 Publisher(s): O'Reilly Media, Inc. ISBN: 9781098108304

2. Big Data: Principles and best practices of scalable realtime data systems, by Nathan Marz (Author), James Warren (Author), Publication date 2015 May 10, ISBN: 1617290343

3. Cloud Computing: Concepts, Technology & Architecture (The Pearson Service Technology Series from Thomas Erl), by Thomas Erl (Author), Ricardo Puttini (Author), Zaigham Mahmood (Author), Publisher : Pearson; 1st edition (May 10, 2013), ISBN-10 : 9780133387520

4. Modern Data Stack [Електронний ресурс] – Режим доступу. – URL: www.moderndatastack.xyz/stacks/facebook

5. Data Stack [Електронний ресурс] – Режим доступу. – URL: sarasanalytics.com/blog/modern-data-stack

6. ETL та ELT [Електронний ресурс] – Режим доступу. – URL: habr.com

7. Distributed Computing [Електронний ресурс] – Режим доступу. – URL: aws.amazon.com/ru/what-is/distributed-computing

8. Ebay Tech Blog [Електронний ресурс] – Режим доступу. – URL: innovation.ebayinc.com/tech

9. Databricks engineering Blog [Електронний ресурс] – Режим доступу. – URL: www.databricks.com/blog/category/engineering

10. Google Cloud Blog [Електронний ресурс] – Режим доступу. – URL: cloud.google.com/blog

11. Altexsoft [Електронний ресурс] – Режим доступу. – URL: www.altexsoft.com/blog/apache-airflow-pros-cons

12. Projectpro [Електронний ресурс] – Режим доступу. – URL: https://www.projectpro.io

13. BlueXP [Електронний ресурс] – Режим доступу. – URL: bluexp.netapp.com

14.Engineering at Meta - [Електронний ресурс] – Режим доступу. – URL: engineering.fb.com

15.Google Research - [Електронний ресурс] – Режим доступу. – URL: ai.googleblog.com

16.AWS Architecture Blog - [Електронний ресурс] – Режим доступу. – URL: aws.amazon.com/blogs/architecture

17.All Things Distributed - [Електронний ресурс] – Режим доступу. – URL: www.allthingsdistributed.com

18.The Netflix Tech Blog - [Електронний ресурс] – Режим доступу. – URL: netflixtechblog.com

19.LinkedIn Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: engineering.linkedin.com/blog

20.Uber Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: eng.uber.com

21.Engineering at Quora - [Електронний ресурс] – Режим доступу. – URL: quoraengineering.quora.com

22.Pinterest Engineering - [Електронний ресурс] – Режим доступу. – URL: medium.com/pinterest-engineering

23.Lyft Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: eng.lyft.com

24.Twitter Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: blog.twitter.com/engineering/en_us

25.Dropbox Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: dropbox.tech

26.Spotify Engineering - [Електронний ресурс] – Режим доступу. – URL: engineering.atspotify.com

27.Github Engineering - [Електронний ресурс] – Режим доступу. – URL: github.blog/category/engineering

28.Instagram Engineering - [Електронний ресурс] – Режим доступу. – URL: instagram-engineering.com

29.Canva Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: canvatechblog.com

30.Etsy Engineering - [Електронний ресурс] – Режим доступу. – URL: www.etsy.com/codeascraft

31.Booking.com Tech Blog - [Електронний ресурс] – Режим доступу. – URL: blog.booking.com

32.Expedia Technology - [Електронний ресурс] – Режим доступу. – URL: medium.com/expedia-group-tech

33.The Airbnb Tech Blog - [Електронний ресурс] – Режим доступу. – URL: medium.com/airbnb-engineering

34.Stripe Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: stripe.com/blog/engineering

35.Flickr's Tech Blog - [Електронний ресурс] – Режим доступу. – URL: code.flickr.net

36.Hubspot Product and Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: product.hubspot.com/blog/topic/engineering

37.Zynga Engineering - [Електронний ресурс] – Режим доступу. – URL: www.zynga.com/blogs/engineering

38.Yelp Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: engineeringblog.yelp.com/

39.Heroku Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: blog.heroku.com/engineering

40.Discord Engineering and Design - [Електронний ресурс] – Режим доступу. – URL: discord.com/blog

41.Zomato - [Електронний ресурс] – Режим доступу. – URL: blog.zomato.com/category/technology

42.Hotstar - [Електронний ресурс] – Режим доступу. – URL: blog.hotstar.com

43.Swiggy - [Електронний ресурс] – Режим доступу. – URL: bytes.swiggy.com

44.Acast Tech - [Електронний ресурс] – Режим доступу. – URL: medium.com/acast-tech

45.ASOS Tech Blog - [Електронний ресурс] – Режим доступу. – URL: medium.com/asos-techblog

46.Shopify Engineering - [Електронний ресурс] – Режим доступу. – URL: shopify.engineering

47.Microsoft Tech Blogs - [Електронний ресурс] – Режим доступу. – URL: devblogs.microsoft.com/

48.MongoDB Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: www.mongodb.com/blog/channel/engineering-blog

49.Slack Engineering - [Електронний ресурс] – Режим доступу. – URL: slack.engineering

50.Engineering at Depop - [Електронний ресурс] – Режим доступу. – URL: engineering.depop.com

51.SourceDiving (Cookpad's Engineering Blog) - [Електронний ресурс] – Режим доступу. – URL: sourcediving.com

52.Auto Trader Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: engineering.autotrader.co.uk

53.Indeed Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: engineering.indeedblog.com/blog

54.Gusto Engineering Blog - [Електронний ресурс] – Режим доступу. – URL: engineering.gusto.com/

55.Engineering at Birdie - [Електронний ресурс] – Режим доступу. – URL: medium.com/engineering-at-birdie

56.Forethought Engineering - [Електронний ресурс] – Режим доступу. – URL: engineering.forethought.ai

57.Capital One - [Електронний ресурс] – Режим доступу. – URL: www.capitalone.com/tech/blog