Питання для закріплення

1. У чому полягає проблема обчислювальної функції?
2. Опишіть особливості технології Spark.
3. Яка відмінність між Spark та MapReduce?
4. Які пакети в R використовуються для роботи з Spark?

Список рекомендованої літератури

1. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics

2. Apache Spark // Електронний ресурс. Режим доступу: https://spark.apache.org/

3. Spark і sparklyr для роботи з великими даними в R // Електронний ресурс. Режим доступу: https://r-analytics.blogspot.com/2020/02/spark-intro.html

4. tidyverse/nycflights13 // Електронний ресурс. Режим доступу: https://github.com/tidyverse/nycflights13

5. Локальний Spark-кластер // Електронний ресурс. Режим доступу: https://r-analytics.blogspot.com/2020/02/spark-r-connect.html

6. Аналіз даних в Spark-кластері за допомогою пакета dplyr // Електронний ресурс. Режим доступу: https://r-analytics.blogspot.com/2020/03/spark-dplyr.html

Лекція 18.

Lambda та Kappa архітектури оброблення великих даних

План лекції

18.1. Lambda - архітектура.
18.2. Переваги і недоліки Lambda -архітектури.
18.3. Kappa - архітектура.
18.4. Переваги і недоліки Kappa-архітектури.

18.1. Lambda - архітектура

Прагнучи зблизити аналіз «історичних» даних та даних, отримуваних у режимі реального часу [1], була створена архітектура Lambda (рис. 18.1).

Lambda – це архітектура обробки даних, яка використовує як обробку потоків, так і пакетну обробку для отримання точного перегляду як "живих" даних, так і пакетних даних.

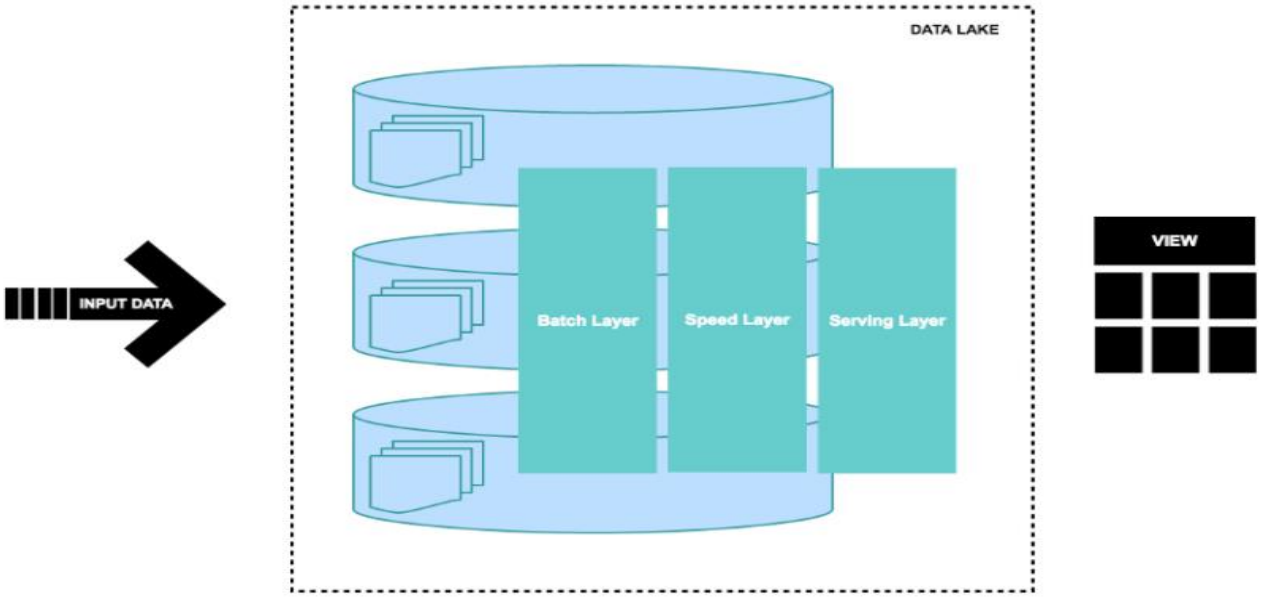

Рис. 18.1. Узагальнена Lambda – архітектура [2]

Lambda -архітектура має чотири шари (рис. 18.2):

Ingestion – цей рівень імпортує дані. Це можуть бути дані з багатьох джерел, включаючи потоки даних.

Batch – це дані в рівні спокою. Дані тут часто будуються за графіком і включають імпорт даних із потокового шару. Точність важливіша за швидкість.

Stream – це складний шар, що підтримує поступове оновлення. Низька затримка тут має більший пріоритет, ніж точність.

Presentation – цей шар запускає операцію та приймає всі запити і використовує швидкісний або пакетний шар.

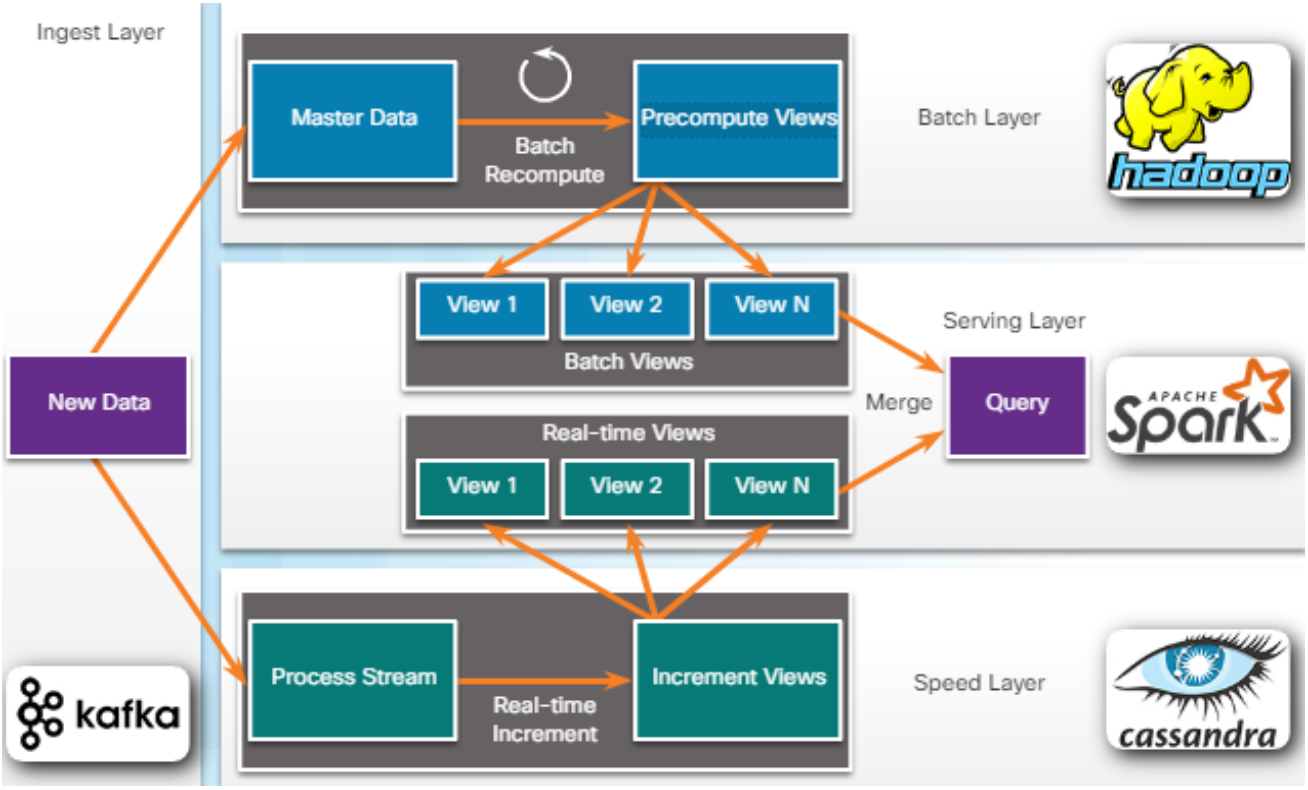

Рис. 18.2. Технології Lambda – архітектури [1]

Усі дані, що надходять, надсилаються одночасно і в пакетний шар, і в швидкісний шар. Пакетний шар управляє набором основних даних і попередньо обчислює пакетні представлення, які постійно обчислюються. Обслуговуючий шар індексує перегляди пакетів, щоб їх можна було запитати.

Швидкість шару стосується лише останніх даних, що компенсує затримку оновлень рівня, який обслуговує дані. Два різні результати запиту можуть бути об'єднані, щоб сформувати новий тип даних.

IoT є ідеальною областю для реалізації архітектури Lambda. Дані генеруються з великою швидкістю, набори даних можуть бути дуже великими, а запити можуть бути як для даних у спокої, так і для даних у русі.

Розглянемо приклад із реального життя використання архітектури Lambda для візуалізації 171 автобусних маршрутів у Лос-Анджелесі, Каліфорнія [1]. Lambda включає SACK (Spark, Akka, Cassandra, Kafka). Це дозволяє здійснювати аналітику даних у режимі реального часу.

Akka – це безкоштовний інструмент з відкритим кодом для створення розподілених та стійких програм, керованих повідомленнями [3].

Akka використовується для отримання метаданих інформації про маршрут та зберігання їх у Cassandra кожні 30 секунд. У цьому прикладі Akka для запиту даних використовує безкоштовний API REST. Далі дані зберігаються в Cassandra. Подальші запити надсилаються до Kafka. Потім використовується Spark для зчитування інформації про транспортний засіб від Kafka. Коли всі ці дані доступні, інший API використовується для візуалізації положення шин за допомогою OpenStreetMap. Поточні положення автобусів передаються прямо з Kafka на карту за допомогою веб-розмови [3, 4].

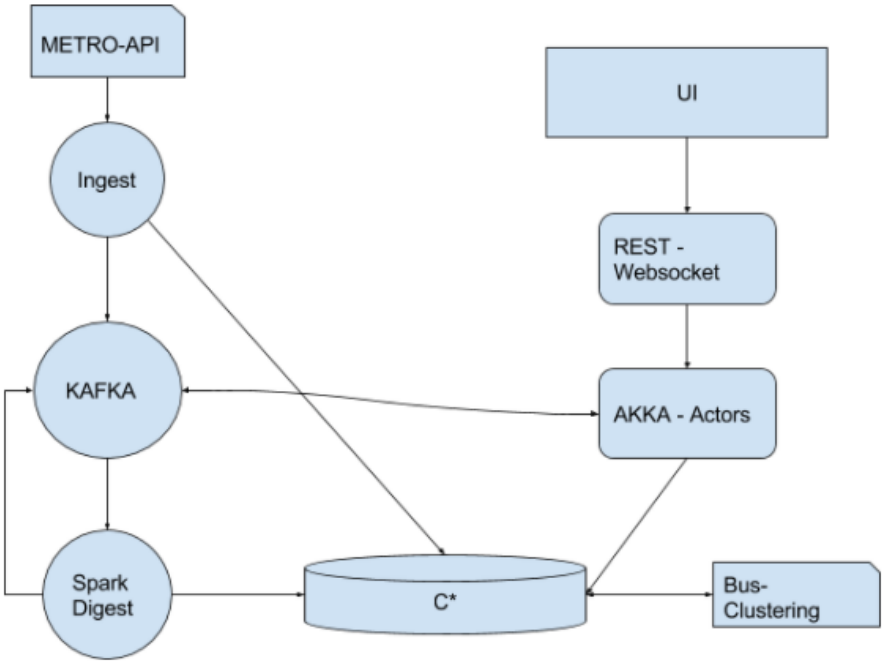

Рис. 18.3. Архітектура IoT Analytics Platform (Ingest – Akka, Digest – Spark, UI – Javascript, Openstreetmap, Backend – Akka) [3]

Ця платформа не обмежується лише даними IoT. Вона підходить для рішень, де є багато вхідних даних. Це дуже корисно, коли є кілька паралельно вхідних потоків даних. Це краще рішення, ніж традиційні RDBMS, які не зможуть надати дані в реальному часі.

Lambda -архітектура може бути розгорнута для тих корпоративних моделей обробки даних, де призначені для користувача запити повинні оброблятися з використанням незмінного сховища даних; потрібні швидкі відповіді, система повинна бути здатна обробляти різні оновлення у формі нових потоків даних;

жоден зі збережених записів не повинен бути видалений, і це повинно дозволити додавати оновлення та нові дані в базу даних. Такі компанії, як Twitter, Netflix і Yahoo, використовують цю архітектуру для відповідності стандартам якості обслуговування.

18.2. Переваги і недоліки Lambda –архітектури

Lambda -архітектура має наступні переваги для Big Data системи:

- збереження історичних даних за рахунок пакетного рівня на базі Hadoop Data Lake або іншого відмовостійкого розподіленого сховища з низькою ймовірністю помилок і збоїв;
- баланс швидкості і надійності;
- масштабованість.

Однак, Lambda -підходу властиві такі недоліки:

- неможливість зміни стратегії аналізу даних «на льоту» – кінцева несуперечливість даних унеможливлює відправку інформації назад на пакетний рівень. Потрібне повторне проведення всіх обчислень, через прив'язку до сховища, дані важко перенести або реорганізувати;
- більшість інструментів, орієнтованих на Lambda-архітектуру, відносяться до NoSQL та не підтримують SQL-запити або інші засоби бізнес-аналітики;
- складність – багато різнорідних компонентів, які передають дані один одному, що затримує обчислення в реальному часі, логіка обробки інформації дублюється з використанням різних структур даних, ускладнюючи загальне управління. Також збільшуються накладні витрати на розробку.

Такі недоліки частково компенсуються за допомогою Apache Spark.

Однак для швидкої обробки подій в режимі реального часу без пакетного представлення більш доцільний інший підхід – Kappa – архітектура (рис.18.4).

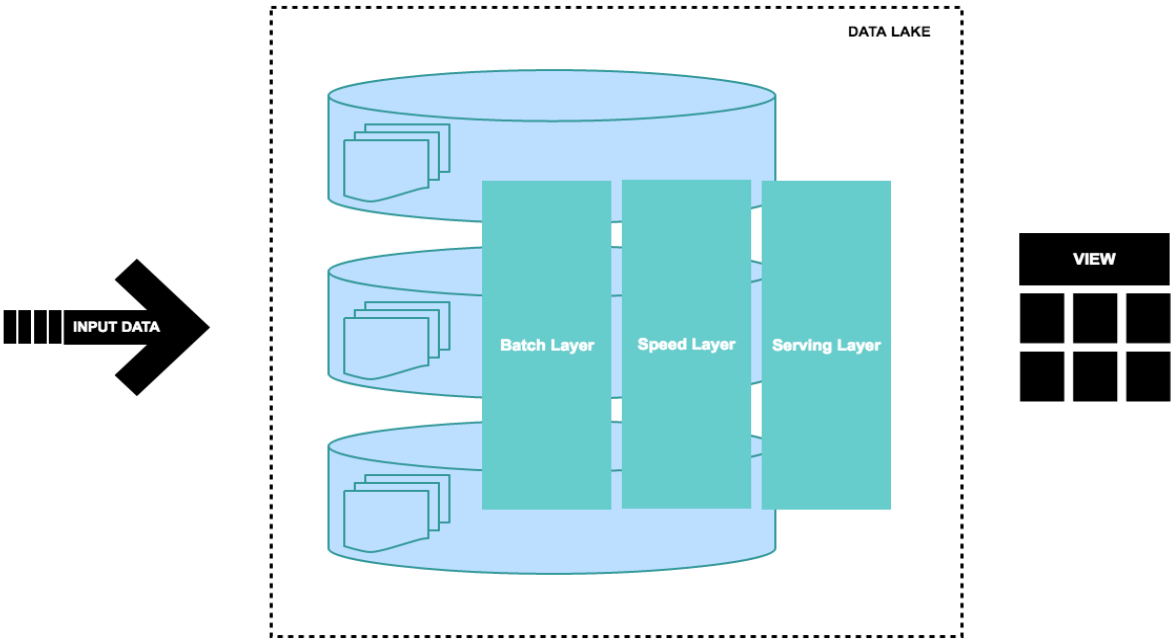

Рис. 18.4. Узагальнена Kappa – архітектура [3]

Деякі варіанти додатків для соціальних мереж, пристроїв, підключених до хмарної системи моніторингу, Інтернету речей (IoT), використовують оптимізовану версію архітектури Lambda, яка використовує служби швидкісного рівня в поєднанні з потоковим рівнем для обробки даних через Data Lake.

18.3. Kappa – архітектура

У 2014 році Джей Крепс почав обговорення, в якому вказав на деякі невідповідності архітектури Lambda, які в подальшому привели світ великих даних до іншої альтернативної архітектури – Kappa, яка використовувала менше ресурсів коду і була здатна добре працювати в певних корпоративних сценаріях. Архітектура Kappa може бути розгорнута для тих корпоративних моделей обробки даних, де:

- кілька подій або запитів даних заносяться в чергу для обробки в розподіленому сховищі файлової системи або в історії;
- порядок подій і запитів не визначений, платформи потокової обробки можуть взаємодіяти з базою даних в будь-який час;
- обробка терабайтів сховища потрібно для кожного вузла системи для підтримки реплікації.

Вищезазначені сценарії даних обробляються за допмогою платформи Apache Kafka, яка забезпечує швидкість обчислень, відмовостійкість і горизонтальну масштабованість. Це дозволяє поліпшити механізм управління потоками даних. Збалансоване управління потоковими процесорами і базами даних дозволяє програмам працювати відповідно до очікувань Kafka, зберігає впорядковані дані протягом більш тривалого часу і обслуговує аналогічні запити, пов'язуючи їх з відповідною позицією збереженого журналу в режимі реального часу.

LinkedIn і деякі інші додатки використовують цей різновид обробки великих даних і отримують вигоду від збереження великого обсягу даних для обслуговування тих запитів, які є простою копією один одного. Архітектура Kappa не може розглядатися як заміна архітектури Lambda, її слід розглядати як альтернативу, яка буде використовуватися в тих випадках, коли для забезпечення стандартної якості обслуговування не потрібно активна продуктивність пакетного рівня.

18.4. Переваги і недоліки Kappa-архітектури

Розглянемо переваги Kappa-архітектури. Архітектура Kappa може бути використана для розробки систем даних, які навчаються в режимі онлайн і тому не потребують пакетноого рівня:

- повторна обробка потрібна тільки при зміні коду;
- може використовуватися для горизонтально масштабованих систем;
- потрібно менше ресурсів, так як машинне навчання здійснюється в режимі реального часу.

Розглянемо недоліки Kappa-архітектури. Відсутність пакетного рівня може призвести до помилок при обробці даних або при оновленні бази даних, що вимагає наявності диспетчера винятків для повторної обробки даних. Вибір між архітектурою Lambda і Kappa здається компромісом. Якщо потрібна архітектура, більш надійна в оновленні Data Lake, а також ефективна в розробці моделей машинного навчання для надійного прогнозування нових подій, слід

використовувати архітектуру Lambda, оскільки вона використовує переваги пакетного рівня і швидкість шару, щоб забезпечити менше помилок.

Якщо потрібно розгорнути архітектуру великих даних з використанням менш дорогого обладнання і вимагати від неї ефективної обробки на основі унікальних подій, що відбуваються під час виконання, доцільно використовувати архітектуру Kappa для оброблення даних в реальному часі.

Висновок до лекції 18

Lambda – це архітектура обробки даних, яка використовує як потокову обробку, так і пакетну обробку, щоб отримати точний перегляд як «живих» даних, так і пакетних даних. Переваги архітектури Lambda:

1. пакетний рівень архітектури Lambda управляє історичними даними за допомогою відмовостійкого розподіленого сховища, яке забезпечує низьку ймовірність помилок, навіть якщо відбувається збій системи;

2. це хороший баланс швидкості і надійності;

3. відмовостійка і масштабована архітектура для обробки даних.

Недоліки архітектури Lambda:

1. накладні витрати кодування через використання комплексної обробки;

2. повторне оброблення кожного пакетного циклу, що не вигідно в певних сценаріях;

3. дані, змодельовані з використанням архітектури Lambda, важко перенести або реорганізувати.

Архітектура Kappa може бути використана для розробки інтелектуальних систем даних, які навчаються в режимі онлайн і тому не потребують пакетного рівня, коли повторна обробка потрібна тільки при зміні коду, може використовуватися для горизонтально масштабованих систем, де потрібно менше ресурсів, так як машинне навчання здійснюється в режимі реального часу.

Питання для закріплення

1. Опишіть Lambda – архітектуру оброблення великих даних.
2. Опишіть переваги і недоліки Lambda -архітектури.
3. Опишіть Kappa - архітектуру оброблення великих даних.
4. Опишіть переваги і недоліки Kappa-архітектури.

Список рекомендованої літератури

1. IoT Fundamentals: Big Data & Analytics // Електронний ресурс. Режим доступу: https://www.netacad.com/courses/iot/big-data-analytics

2. IoT Analytics Platform // Електронний ресурс. Режим доступу: https://blog.codecentric.de/en/2016/07/iot-analytics-platform/

3. Akka // Електронний ресурс. Режим доступу: https://akka.io/

4. IoT-Analyse-Plattform: Floating Bus Data // Електронний ресурс. Режим доступу: https://www.youtube.com/watch?v=VYxc-3ZRRL4&ab_channel=codecentricAG

5. A brief introduction to two data processing architectures — Lambda and Kappa for Big Data // Електронний ресурс. Режим доступу: https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb