УДК 004.94 https://doi.org/10.15407/pp2019.03.101

О. Захарова

ПЛАТФОРМИ ВЕЛИКИХ ДАНИХ. ОСНОВНІ ЗАДАЧІ, ВЛАСТИВОСТІ ТА ПЕРЕВАГИ

Дана робота представляє огляд існуючих платформ великих даних. Мета полягає у визначені основних проблем та рішень, які існують в цій сфері, а також властивостей платформ великих даних, що визначають їх можливості, переваги чи недоліки у вирішенні цих проблем. Актуальність теми обумовлена стрімким розвитком мобільних пристроїв і прикладних систем, відповідним ростом обсягів інформації та нездатністю традиційних систем обробляти таки обсяги даних у придатні терміни. Тобто, це платформа інформаційних технологій класу підприємства, яка забезпечує властивості та функціональність прикладної системи в одному рішенні для розробки, розгортання, обробки та управління великими даними. Метою створення та використання таких платформ є покращення масштабованості, доступності, продуктивності та безпеки організацій, які працюють з великими даними. Платформи великих даних надають можливість обробляти об'ємні багатоструктурні дані в режимі реального часу та дозволяють різним користувачам застосовувати її для виконання різних задач, що пов'язані із використання великих даних. В роботі розглядаються фреймворки, що розроблені для вирішення задач великих даних, аналізуються їх характеристики, принципи роботи, переваги у контексті проблем, які вони здатні вирішувати, визначаються існуючі «прогалини» та напрямки подальшого розвитку. Вирішення проблем великих даних, а саме забезпечення ефективного зберігання, обробки та аналізу даних, дозволить зробити інформацію кориснішою, а підприємства, які працюють з великими даними, конкурентноздатнішими.

Ключові слова: платформа великих даних, машинне навчання, Apache Hadoop, документоорієнтоване сховище, сховище типу «ключ-значення», стовпчикове сховище, графове сховище, управління даними, розподілене зберігання, потокові обчислення, NoSQL бази даних, розподілена файлова система, MapReduce, TaskTracker, JobTracker, Apache Spark, обробка великих даних, аналітика великих даних.

Вступ

Сьогодні, в умовах зростаючої конкуренції, успіх будь-якої організації визначається можливістю миттєвого доступу до операційних даних та аналізу цих даних. Компанії намагаються своєчасно отримувати необхідну інформацію, щоб оперативно реагувати на змінення ринку. Але обсяг операційних даних може бути настільки великим, що традиційні системи на базі жорстких дисків не справляються з їх обробкою у необхідні терміни. В результаті керівники не отримують потрібну інформацію, а між збором даних та їх аналізом триває занадто багато часу.

Внаслідок технологічної революції, протягом останніх років постійно зростає кількість ядер процесорів та обсяг інтегрованої кеш-пам'яті в обчислювальних системах. Сьогодні обсяг основної пам'яті є практично не обмеженим, що дозволяє завантажувати дані будь-якого розміру. З іншого боку, стрімке розширення ринку мобільних пристроїв та прикладних систем призвело до того, що інформація для прийняття рішень стала доступною для бізнес-користувачів саме в той момент, коли вона необхідна. Індустрія обробки інформації усвідомила вибухове зростання даних, які складно укласти в реляційні сховища завдяки їх неструктурованій природі та не можливо швидко привести до ладу внаслідок великої швидкості зростання обсягів. Це дані від багатьох вимірювальних приладів, з потоків повідомлень у соціальних мережах, метеорологічна інформація та дані журналів подій. Ще одна причина, з якої існуючі раніше рішення виявилися не придатними, – дуже висока вартість зберігання дійсно великих обсягів даних в реляційних БД великих виробників [1].

Так виник термін "великі дані". Він застосовується до наборів даних, які настільки великі, що з ними не можна працювати, використовуючи традиційні системи керування даними. Це набори даних, розмір яких перевищує можливості загально використовуваних програмних засобів та систем зберігання для вирішення задач витягнення, зберігання, управління та обробки у допустимий час.

У таких умовах, організація, щоб лишатися конкурентоспроможною, повинна мати здатність виявляти змінення ситуації на ринку, нові ризики та можливості у режимі реального часу. Традиційні методи управління даними та сховища даних не дозволяють адекватно обробляти та аналізувати дані великих обсягів. Великі дані потрібно не лише генерувати та радіти їх обсягу й різноманітності, інформація, яку вони містять, повинна бути корисною. А для цього її необхідно правильно зберігати, обробляти, аналізувати та синтезувати нові знання з існуючих – тобто робити виводи. Це пояснює стрімкий розвиток платформи великих даних на протязі останніх років.

Платформа великих даних

Платформа великих даних – це інструмент, розроблений постачальниками програмного забезпечення (ПЗ) для управління даними з метою покращення масштабованості, доступності, продуктивності та безпеки організацій, які працюють з великими даними.

Платформа призначена для обробки в режимі реального часу об'ємних багатоструктурних даних. Різні користувачі можуть її використовувати для виконання різних задач. Так, наприклад, інженери даних – для очищення, агрегування та підготовки даних для аналізу, бізнес-користувачі – для запуску запитів, а вчені вважають її корисною при аналізі шаблонів з наборів великих даних за допомогою алгоритмів машинного навчання.

Це платформа інформаційних технологій (ІТ) класу підприємства, яка забезпечує властивості та функціональність прикладної системи в одному рішенні для розробки, розгортання, обробки та управління великими даними. Програмне забезпечення (ПЗ) аналітики великих даних допомагає розкрити приховані шаблони, невідомі кореляції, ринкові тенденції, вподобання клієнтів та іншу корисну інформацію з широкого різноманіття наборів даних.

Головне питання організації роботи з великими даними на корпоративному рівні: обрати реляційну (SQL) чи нереляційну (NoSQL) базу даних? Головною причиною відмови від SQL баз даних (БД) є не правильна робота з самою базою. Більшість компаній не можуть собі дозволити тримати спеціалістів для постійного налагодження баз даних, а для того, щоб розпочати використовувати NoSQL БД не потрібно додаткових розробок. При розробці NoSQL БД особлива увага приділяється забезпеченню високої масштабованості та гнучкості рішень. NoSQL [2] БД – це, перш за все, швидкий доступ до даних, що зберігаються в оперативній пам'яті, гнучкість використання та можливість швидкого розподілення даних між вузлами. Однак можливі такі сценарії, коли дані згодом виходять з-під контроля або вже просто не вміщуються в оперативній пам'яті.

Основні властивості та переваги платформ великих даних

До основних властивостей платформ великих даних можна віднести:

- *забезпечення ефективного зберігання та обробки даних, а також їх інтеграції, управління, витягнення, трансформації, та завантаження* (*ETL*);
- *використання системи Hadoop*: забезпечують функції для масового зберігання даних будь-якого типу, величезну потужність обробки та можливість обробляти практично необмежену кількість паралельних задач;
- *потокові обчислення*: забезпечують функції для *затягування* даних у потік, обробки даних та передачі їх назад єдиним потоком;
- *функції розвиненої аналітики та машинного навчання*;
- *функції управління життєвим циклом контенту та документів;*
- *функції інтеграції великих даних з будь-якого джерела*;
- *управління даними*: містять комплексну систему безпеки, рішення для управління даними та забезпечують дотримання вимог щодо захисту даних.

До головних переваг платформи великих даних та ПЗ аналітики великих даних можна віднести:

- *точні дані.* Платформа великих даних пропонує точні дані, що сприяє прийняттю правильних рішень. Її аналітичні засоби зменшують ризик отримання недостовірних даних, які виникають внаслідок використання сирих, не проаналізованих даних;
- *підвищення ефективності праці.* Платформа спрощує отримання джерела необхідної інформації. Пропонує також інформацію, що може стати у нагоді в майбутньому, таким чином, зберігаючи час та підвищуючи ефективність роботи користувачів;
- *швидкі відповіді на складні питання.* Ефективне управління бізнесом вимагає швидких адекватних відповідей на критичні питання, які впливають на успішність бізнес-операції. Платформа великих даних дозволяє робити це більш надійно. Деякі критичні питання, відповіді на які вимагають тижнів або місяців, за наявності правильного інструменту можуть вирішуватись лише за кілька годин або хвилин;
- *безпека даних.* Забезпечує безпечну інфраструктуру, яка гарантує безпеку даних.

Задачі та ПЗ великих даних

Програмні засоби великих даних можна класифікувати за задачами, які вони вирішують. У процес створення рішення повинні бути інтегровані різні засоби для зберігання, управління та аналізу великих даних. Інструменти великих даних відповідно до їх задач включають наступні групи:

- ПЗ зберігання великих даних;
- ПЗ управління великими даними;
- ПЗ обробки великих даних;
- методи та засоби візуалізації великих даних;
- методи та засоби аналітики великих даних.

Таким чином, відповідний фреймворк [3] повинен відображати інструменти для зберігання, управління та обробки великих даних, інструменти та методи аналітики, візуалізації та оцінювання у різні етапи процесу побудови рішення. Аналітика великих даних може застосовуватись до виявлення знань та обгрунтованого прийняття рішень.

Зберігання та управління великими даними

Традиційні методи структурованого зберігання та витягування даних, такі як реляційні БД, вітрини або SQL сховища даних, мають певні обмеження, які роблять їх не придатними для роботи з великими даними, а саме вони:

- не дозволяють включати нові джерела даних без їх попереднього очищення та інтеграції,
- не дозволяють швидко виробляти та адаптувати дані,
- не забезпечують можливості синхронізації логічного та фізичного вмісту БД з швидкою еволюцією даних,
- не забезпечують поточні потреби аналізу даних.

Необхідність порівняно недорогого зберігання та обробки гігантських обсягів неструктурованої інформації призвела до створення спеціалізованого ПЗ, яке дозволило розподіляти дані за кластерами з сотень та тисяч вузлів, а також обробляти їх у паралельному режимі. Засоби нового покоління для зберігання та управління не структурованими (не реляційними) даними, а саме NoSQL БД, дозволили використовувати репозиторій даних без додаткових розробок, підготовки або налагодження, забезпечили високу масштабованість, розподілення даних між вузлами та швидкий доступ до даних, що зберігаються в оперативній пам’яті. NoSQL [2] БД дозволяють записувати задачі управління даними на прикладному рівні. Кожна база, в даному випадку, є колекцією незалежних документів, де кожний документ підтримує власні дані та схеми та може мати метадані – оглядову інформацію про дані документа. Прикладна програма може мати доступ до багатьох БД, розташованих у різних місцях.

Нові вимоги до зберігання, управління та обробки даних обумовили виникнення Hadoop – фреймворка з відкритим кодом під крилом Apache Sotware Foundati-

on, що дозволяє створювати розподілені системи на базі відносно недорогого обладнання масового попиту. З часом Hadoop був розширений набором бібліотек та утиліт, та сформував навколо себе екосистему проектів з розподіленої обробки даних. Розглянемо його більш детально.

Apache Hadoop фреймворк

Apache Hadoop [4] забезпечує розподілене зберігання та обробку дуже великих наборів даних на комп'ютерних кластерах з промислового комп'ютерного обладнання. Тобто, замість того, щоб використовувати один великий комп'ютер, Hadoop дозволяє кластеризувати апаратне забезпечення для паралельного виконання аналізу масивних наборів даних. Сервіси Hadoop забезпечують виконання наступних функцій:

- зберігання даних;
- обробка даних;
- доступ до даних;
- управління даними;
- безпека та
- операції з даними.

Екосистема Hadoop [5, 10] складається з багатьох модулів (процедур, бібліотек та властивостей), які розглядаються як частини фреймворка. Кожний модуль виконує певну задачу, необхідну для виконання аналітики великих даних. Ядро Hadoop складається з двох основних модулів: розподіленої файлової системи Hadoop Distributed File System (HDFS) та базового інструменту для обробки даних MapReduce та підтримується майже всіма відомими постачальниками систем великих даних. Також до найбільш використовуваних відносять планувальник завдань та управління кластерами YARN і множину загальних утиліт Hadoop Common.

Розподілена файлова система. HDFS [4] дозволяє зберігати дані у простому доступному форматі. Це досягається завдяки використанню великої кількості пов'язаних пристроїв зберігання даних та механізму MapReduce для їх обробки.

"Файлова система" є методом, що застосовується комп'ютером для зберігання даних таким чином, щоб їх можна було знаходити та використовувати. Зазвичай, він визначається операційною системою (ОС) комп'ютера, але система Hadoop використовує власну файлову систему, яка надбудовується "над" файловою системою хост-комп'ютера. Це означає, що доступ до даних можна отримати з будь-якого комп'ютера, на якому встановлена будь-яка підтримувана ОС.

Hadoop розділяє файли на великі блоки та розподіляє їх між вузлами у кластері. Потім він передає пакетований код у вузли для обробки даних у паралельному режимі. В даному підході вузли маніпулюють даними, до яких вони мають доступ. Це дозволяє обробляти набір даних швидше та ефективніше, ніж в більш традиційній архітектурі суперкомп'ютера, яка спирається на паралельну файлову систему, де обчислення та дані розподіляються у високошвидкісній мережі.

HDFS є розподіленою, масштабованою та портативною файловою системою, що написана на Java для Hadoop фреймворк. Вона забезпечує виконання команд та Java інтерфейсів (API), подібних до інших файлових систем, для зв'язку використовує протокол TCP/IP. Клієнти для спілкування один з одним використовують виклики віддаленої процедури (RPC). Надійність зберігання даних досягається шляхом реплікації між декількома хостами. Щоб зменшити трафік у мережі, Hadoop необхідно знати, які сервери є найближчими до даних або інформації, яка може забезпечити встановлення мостів з HDFS.

Hadoop може працювати безпосередньо з будь-якою розподіленою файловою системою, яка може бути встановлена основною операційною системою.

Прикладами файлових систем, що підтримуються Hadoop (окрім HDFS), є:

- FTP (зберігає всі свої дані на віддалених FTP-серверах);
- сховище об'єктів Amazon S3 (Simple Storage Service), орієнтоване на кластери, які розміщені на інфраструктурі Amazon Elastic Compute Cloud типу сервер-на-запит;
- Windows Azure Storage Blobs (WASB), розширення HDFS, яке дозволяє

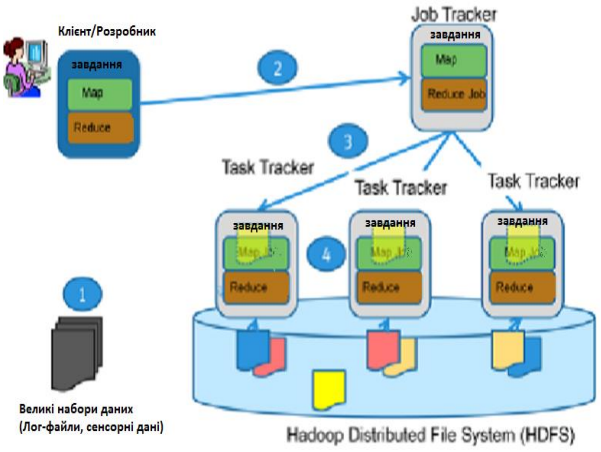
розподілення Hadoop для доступу до даних в Azure блог-сховищах без постійного переміщення даних у кластер.

Існують також файлові системи, які не розповсюджуються разом з Hadoop, але постачаються як альтернативні, що використовуються за замовченням, з де-якими його комерційними рішеннями. Наприклад: IBM General Parallel File System, Parascale, Appistry (драйвер файлової системи Hadoop для використання з CloudIQ Storage), драйвер файлової системи IBRIX Fusion, альтернативна файлова система MapR FS, що заміщує HDFS системою повністю випадкового доступу для читання/запису файлів.

Модуль MapReduce. MapReduce названий за двома головними операціями, які він виконує, а саме: читання даних з БД і переведення їх у формат, що підходить для аналізу (map), та виконання математичних операцій (reduce).

Функціонування MapReduce [11] забезпечується двома компонентами: JobTracker та TaskTracker. Клієнтські прикладні програми направляють завдання MapReduce до JobTracker, JobTracker працює з доступними у кластері вузлами TaskTracker, щоб наблизитись до потрібних для виконання цих завдань даних. JobTracker відомо, які вузли містять дані, та які інші комп'ютери є поруч (рис. 1).

Якщо робота не може бути виконана на тому вузлі, де розміщені дані, перевага надається вузлам, що розміщені на тій самій стойці. Таким чином, скорочується трафік на головній магістральній мережі. Якщо TaskTracker виходить з ладу або зазнає затримку, здійснюється перепланування частини завдань. TaskTracker в кожному вузлі породжує окремий процес віртуальної машини Java (JVM). Це дозволяє запобігти виходу з ладу TaskTracker, якщо запущене на виконання завдання зруйнує свою JVM. Кожні кілька хвилин з TaskTracker до JobTracker надсилається імпульс, щоб перевірити його стан. Стани JobTracker і TaskTracker та інформація про їх роботу відображаються у контейнері сервлетів Jetty та її можна також переглядати у веб-браузері [11].



❶ Дуже великий набір даних. HDFS зберігає репліки даних у вузлах даних.

❷ Клієнт виконує Map та Reduce завдання на конкретному наборі даних та відсилає їх JobTracker.

❸ JobTracker розподіляє завдання серед TaskTracker. TaskTracker запускає механізм відображення (map), результат роботи якого зберігається у HDFS.

❹ Запускається завдання Reduce на даних, що вже оброблені завданням Map

Рис. 1. Робота Map Reduce модуля

Слід зазначити, що даний підхід має певні обмеження:

- Алгоритм розподілення роботи TaskTracker є дуже простим. Кожний TaskTracker має множину наявних слотів. Кожна активна задача займає один слот. JobTracker розподіляє роботу на TaskTracker з наявним слотом, найближчий до даних. При цьому не приймається до уваги поточна завантаженість системи призначеної машини – її реальна доступність.
- Якщо один TaskTracker дуже повільний, це може затримати роботу MapReduce в цілому. Однак, коли дозволене паралельне виконання, окрема задача може виконуватися на декількох підпорядкованих вузлах.

В первинному варіанті, для впорядкування завдань з робочої черги, Hadoop підтримує First-In-First-Out (FIFO) планування та опціональне планування пріоритетів, які використовуються за замовченням. Згодом до планувальника завдань бу-

ла додана можливість використовувати альтернативні планувальники, такі як Fair (Facebook AI Research) або Capacity. Планувальник Fair [16] є розробкою Facebook. Розробники мали за мету забезпечити швидку відповідь для невеликих завдань та якість сервісу для виробничих завдань. Завдання групуються у пули і ресурси розподіляються між цими пулами. За замовченням для кожного користувача є окремий пул, так що кожний користувач отримує рівну частку кластера. На відміну від планувальника Hadoop, що використовується за замовченням та формує чергу завдань, Fair дозволяє коротким завданням завершуватись у розумний час, не очікуючи довго своєї черги. Це також є простим способом спільного використання кластера між кількома користувачами, яке також може працювати з пріоритетами завдань. Ці пріоритети використовуються як ваги для визначення частки загального часу обчислення, яке отримує кожне завдання.

Планувальник Capacity, розроблений Yahoo, підтримує декілька властивостей, подібних властивостям Fair, а саме: кожній черзі виділяється частка загального ресурсу, вільні ресурси виділяються чергам за їх потужністю. У черзі завдання з більшим пріоритетом мають пріорітетніший доступ до її ресурсів.

HDFS не обмежується MapReduce завданнями. Вона може працювати з іншими прикладними програмами, багато з яких є розробками Apache, наприклад, база даних HBase, система машинного навчання Apache Mahout, система Apache Hive Data Warehouse. Теоретично, Hadoop може використовуватися для будь якого типу завдань, які є швидше пакетно-орієнтованими, ніж тими, що виконуються у реальному часі, а також для завдань з дуже інтенсивними даними і завдань, для яких корисна паралельна обробка даних. Hadoop також може використовуватися для доповнення системи реального часу, такої як, наприклад, лямбда архітектура, Apache Storm, Flink або Spark.

Hadoop Common та планувальник Yarn. Hadoop Common забезпечує інструменти, які дозволяють комп'ютерним системам користувача читати дані, що зберігаються у файловій системі Hadoop.

YARN керує ресурсами систем, які зберігають дані та виконують їх аналіз.

Використання Hadoop. Hadoop також містить безліч інших інструментів з відкритим кодом, призначених для створення додаткових функцій на компонентах ядра Hadoop.

Так, Apache Tez є фреймворком наступного покоління, який може використовуватися замість Hadoop MapReduce, в якості двигуна. Amazon EMR включає конектор EMRFS, який дозволяє Hadoop використовувати для зберігання даних сховище Amazon S3. Amazon EMR також може використовуватися для легкого встановлення та налаштування у кластері таких інструментів, як Hive, Pig, Hue, Ganglia, Oozie та HBase. Окрім Hadoop на Amazon EMR можна запускати інші фреймворки, такі як Apache Spark для обробки даних у пам'яті або Presto для виконання інтерактивних запитів.

Гнучка природа системи Hadoop дозволяє компаніям, коли вони потребують змін, додавати або змінювати власну систему даних, використовуючи дешеві та легко-доступні частини від будь-яких постачальників інформаційних систем. Сьогодні Hadoop є найбільш використовуваною системою для зберігання та обробки даних на виробничому апаратному забезпеченні. Hadoop використовують майже всі великі постачальники он-лайн продуктів, та кожний має можливість його вільно модифікувати відповідно до своїх цілей. Ці зміни, які вносять до ПЗ експерти, наприклад, Amazon чи Google, відсилаються до спільноти розробників, де вони часто використовуються в подальшому для вдосконалення "офіційного" продукту. Така форма колаборативної розробки є ключовою властивістю ПЗ з відкритим кодом.

Слід зазначити, що використання базових модулів Hadoop Apache є складним навіть для фахівця галузі інформаційних технологій, тому були розроблені комерційні версії продукту такі, як наприклад, Cloudera, що спрощують задачу інсталяції та запуску Hadoop, а також пропонують послуги навчання та підтримки. Завдя-

ки гнучкій природі Hadoop, компанії при розширенні бізнесу мають можливість корегувати та розширювати операції аналізу даних. Підтримка спільноти відкритого коду робить аналіз великих даних доступним для кожного.

Apache Spark

Apache Spark – фреймворк (містить більш ніж 80 операторів для роботи з даними) з відкрититм кодом, який був створений для розподіленої обробки великих даних. На відміну від класичного обробника з ядра Hadoop, що реалізує дворівневу концепцію MapReduce з дисковим сховищем, він використовує спеціалізовані примитиви для рекурентної обробки в оперативній пам'яті, завдяки чому дозволяє отримувати значне прискорення роботи для деяких класів задач. Зокрема, можливість багатократного доступу до даних користувача, що завантажені в оперативну пам'ять, робить бібліотеку дуже привабливою для алгоритмів машинного навчання. Фактично, Spark є переосмисленим MapReduce, але працює у 10-100 разів швидше, залежно від того, працює він в пам'яті або на диску. Spark підтримує мови програмування Scala, Python, Java, R.

Головним поняттям у Spark є Resilient Distributed Dataset (RDD) [14] – це розподілена структура даних, яка розміщується в оперативній пам'яті (рис. 2). Кожний RDD є фрагментом даних, що розподілені по вузлах кластера. RDD є незмінними структурами, тому після виконання перетворень створюються нові RDD. RDD обробляються паралельно за допомогою трансформацій/дій, які виконуються одночасно во всіх розділах (partition). RDD є відмовостійкими: якщо розділ втрачається в результаті відмови вузла, він може бути відновлений з вихідних джерел.

Рис. 2. Розподілення RDD

Фактично RDD являє собою набір даних, над яким можна виконувати перетворення двох типів: трансформації та дії. Відповідно, вся робота з цими структурами полягає у послідовності цих перетворень.

Трансформація. Як правило, перетворює якимось чином елементи даного набору даних. Результатом застосування її до RDD є новий RDD. Далі наведений неповний перелік найрозповсюдженіших трансформацій, кожна з яких повертає новий RDD (рис. 3):

- map(f) – застосовує функцію f до кожного елемента набора даних;
- filter(f) – повертає всі елементи набору даних, на яких функція f повернула істинне значення;
- distinct([numTasks]) – повертає набір даних, який містить унікальні елементи вихідного набору даних;

Також підтримуються наступні операції над множинами:

- union(Dataset) – об'єднання з набором даних Dataset,
- intersection(Dataset) – перетин з набором даних Dataset,
- cartesian(Dataset) – результатом операції є новий набір даних, який містить пари (A,B), де A належить вихідному набору даних, а B – набору даних Dataset.

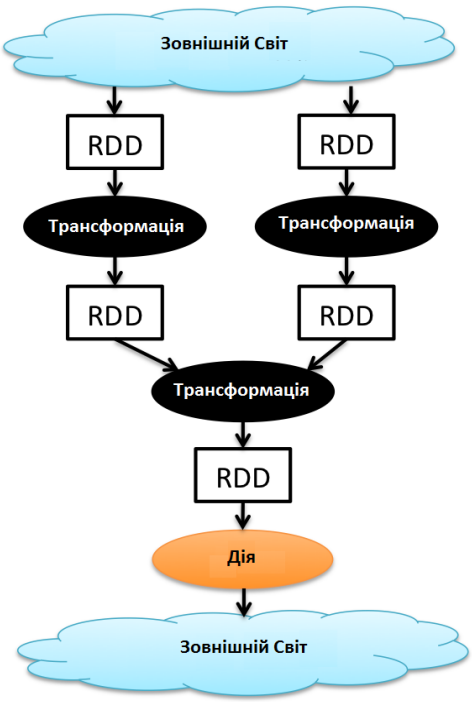


Рис. 3. Робота з RDD

Дії. Застосовуються, коли необхідно матеріалізувати результат – як правило, зберегти дані на диску, або вивести частину даних у консоль. Найбільш розповсюдженими діями, які можна застосувати до RDD, є:

- saveAsTextFile(path) – зберігає дані у текстовому файлі (hdfs) на локальну машину або у будь-яку іншу файлову систему, яка підтримується, path визначає шлях для збереження файлу;
- collect() – повертає елементи набора даних у вигляді масива. Як правило, використовується після застосування до набору даних фільтрів та перетворень для візуалізації або додаткового аналізу результату;
- take(n) – повертає у вигляді масива перші n елементів набору даних;
- count() – повертає кількість елементів у наборі даних;
- reduce(f). Функція f (приймає на вхід 2 аргументи, повертає одне значення) повинна обов'язково бути комутативною та асоціативною.

Spark не змушує думати в парадигмі MapReduce, а дозволяє створювати зрозумілий код, який спрямований саме на виконання поставленої бізнес-задачі. Фреймворк бере на себе розподілення та фрагментацію кода та даних, які автоматично передаються на кластер.

Spark має також колекцію бібліотек (набір готових алгоритмів, підходів та практик), що дозволяють комбінувати існуючі рішення в межах одного програмного кода для досягнення поставленої мети. На цей час Spark містить наступні бібліотеки:

- Spark SQL;
- Spark Streaming (аналіз у реальному часі);
- MLib (машинне навчання);
- GraphX (робота з графами).

Spark SQL. Це модуль Apache Spark, який є частиною ядра Spark та інтегрує реляційну обробку даних та процедурний API Spark. Він може працювати разом з Hive (HiveQL/ SQL) або його заміщувати. Окрім цього, модуль здатний взаємодіяти з інструментами бізнес-аналітики.

Spark SQL підтримує реляційну обробку даних як в межах програм Spark (через RDD), так і з зовнішніх джерел даних. Він може взаємодіяти з новими джерелами даних, включаючи слабоструктуровані дані та зовнішні бази даних, що підтримують федеративні запити.

Spark SQL реалізує та оптимізує реляційну обробку, підтримуючи наступні підходи:

- перетворення даних у більш ефективні формати (з точки зору сховища, мережі та операцій введення/ виведення), зокрема, в різні формати, що орієнтовані на стовбці (columnar format);
- розбиття даних на секції;
- зменшення кількості операцій читання на основі статистики;
- оптимізація операцій над даними;
- виконання оптимізації наскільки можливо пізніше, коли доступна вся інформація по конвейєрах даних.

Spark SQL використовує оптимізатор запитів Catalyst для інтелектуального планування запитів.

Spark SQL може підтримувати пакетний та потоковий SQL. Ядро Spark забезпечує обробку пакетних навантажень через RDD. RDD можуть посилатися на статичні набори даних, а за допомогою розвиненого API Spark можна маніпулювати RDD в оперативній пам'яті із застосуванням «ледачих» обчислень.

Spark Streaming. Реалізує абстракцію DStream (discretized stream, дискретизований потік), що являє собою безперервний потік даних. DStream може бути створений з потоку вихідних даних; на основі таких джерел, як Kafka або Flume, або за допомогою виконання операцій з іншими DStream. По суті, DStream є послідовністю RDD (рис. 4).

Рис. 4. Структура DStream

RDD, що створений за допомогою DStream, можна перетворювати у DataFrame та виконувати SQL запити до нього. Доступ до потоку може надаватися для будь-якої зовнішньої прикладної програми, що підтримує SQL, за допомогою JDBC-драйвера. Пакети потокових даних зберігаються у пам'яті вузла. До цих даних можна будувати інтерактивні запити, використовуючи SQL або API Spark. Для виконання SQL-запитів до Dstream використовується StreamSQL, що поєднує Spark Streaming з Catalyst. StreamSQL є розширенням SQL, яке додатково забезпечує підтримку наступних потокових операцій:

- виборка (SELECT) з потоку для обчислення функцій або фільтрації даних (за допомогою умови WHERE);
- з'єднання (JOIN) потоку з одним або декількома наборами даних для створення нового потоку;
- застосування віконних функцій та виконання агрегацій. Потік можна налаштувати таким чином, щоб він створював набори даних обмеженого розміру. За допомогою віконних функцій можна виконувати складний відбір повідомлень на основі значень полей. Після створення обмеженого пакета можна виконувати аналітику на ньому.

В основу підходу для реалізації аналітики реального часу покладено лямбда-архітектуру, що застосовується для створення аналітичних систем реального часу в контексті великих потокових даних (рис. 5).

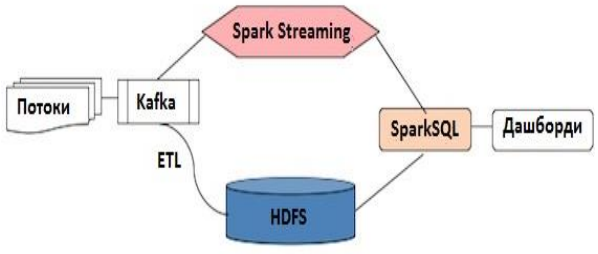


Рис. 5. Логічна схема реалізації аналітики великих даних в реальному часі за допомогою Spark SQL

MLib. Бібліотека для машинного навчання. Її метою є зробити машинне навчання масштабованим та простим. Вона містить розповсюджені алгоритми і утиліти машинного навчання, та дозволяє розпаралелювати на кластері алгоритми машинного навчання (класифікація, регресія, кластеризація і т. і.) лише за пару строк коду. Окрім цього, SparkMLib якісно працює з локальними даними, використовуючи пакет лінійної алгебри Breeze. MLib має добре продуманий API, працює з даними у будь-якому форматі на базі Hadoop та не потребує попереднього встановлення.

GraphX. Розподілений фреймворк обробки графів на основі Apache Spark. Графи є наявною та простою для розуміння моделлю даних. Розподілені обчислення кардинально спростили зберігання та обробку графів.

Головним механізмом ітерації графа в GraphX є розроблений Google алгоритм Pregel. Головною ідеєю цього алгоритму є передача повідомлень між вузлами у графі, які називають супер кроками завдяки послідовності ітерацій. Ітерація часто формується як "think like a vertex", тобто стан поточного вузла залежить лише від стану його сусідів. Використання Pregel є особливо доцільним, коли задачу складно вирішити за допомогою звичайного MapReduce.

Головним примитивом для обходу графа у GraphX є триплет: поточний вузол, вузол, до якого здійснюється перехід, та ребро між ними. Pregel вимагає визначення відстані між вузлами за замовченням, як правило, це PositiveInfinity – UDF (user defined function) функція для кожного вузла, що дозволяє обробити вхідне повідомлення та порахувати наступний вузол, а також UDF функції для злиття двох вхідних повідомлень. Ці функції повинні бути комутативними та асоціативними [15].

GraphX містить статичну та динамічну версії реалізації алгоритму PageRank, який для кожного вузла графа призначає вагу серед решти вузлів. Наприклад, якщо користувач Твітера має велику кількість підписок від інших користувачів, то він буде мати високий рейтинг, тобто, його можна буде легко знайти у пошуковій системі. Статична версія має фіксовану кількість ітерацій, тоді як динаміч-

на версія буде працювати доки рейтинг не почне зходитися до заданого значення.

Через те що GraphX побудований на основі незмінних RDD, графи теж незмінні, тому GraphX непридатний для роботи з графами, які оновлюються, тим більше транзакціями, як у графових БД.

GraphX надає два окремі API для реалізації масово паралельних алгоритмів (таких як PageRank): Pregel-подібний та більш загальний — MapReduce API.

Основні типи NoSQL сховищ

На сьогодні виділяють чотири основних типи NoSQL сховищ [12]:

- *сховище «ключ-значення».* В ньому є велика хеш-таблиця, що містить ключі та значення. (Приклади: Riak, Amazon DynamoDB);
- *документоорієнтоване сховище.* Зберігає документи, які складаються з тегованих елементів. (Приклад: CouchDB);
- *стовпчикове сховище.* У кожному блоці зберігаються дані лише з однієї колонки. (Приклади: HBase, Cassandra);
- *сховище на основі графів.* Мережеве сховище, яке використовує вузли та ребра для відображення та зберігання даних. (Приклад: Neo4J).

Сховище типу «ключ-значення». Відсутність схеми у сховищах типу «ключ-значення» є важливою перевагою для зберігання великих даних. Ключ може бути синтетичним або автосгенерованим, а значення може бути представлено строкою, JSON, блобом (BLOB, Binary Large Object) тощо. Такі сховища, як правило, використовують хеш-таблицю, яка містить унікальний ключ та посилання на певний об'єкт даних. Існує поняття блока – логічної групи ключів, що фізично не поєднують дані у групи. У різних блоках можуть бути ідентичні ключи.

Продуктивність обробки даних значно збільшується за рахунок механизмів хешування, що працюють на основі маппінгів. Щоб прочитати значення, необхідно знати ключ та блок, оскільки насправді ключ є хешем (блок + ключ).

Модель «ключ-значення» проста в реалізації. Такі сховища є доступними та толерантними до розділення, але явно програють у питаннях погодженості даних. В якості недоліків сховищ типу «ключ-значення» можна зазначити:

- модель не надає стандартних можливостей баз даних таких, як атомарність транзакцій або погодженість даних при одночасному виконанні декількох транзакцій. Такі можливості повинні надаватися самою прикладною програмою.
- при збільшенні об'ємів даних, підтримка унікальних ключей може стати проблемою. Для її вирішення необхідно якось ускладнити процес генерації строк, щоб вони залишалися унікальними серед дуже великої множини ключей.

Документоорієнтоване сховище. Дані, які представлені парами ключ-значення, стискаються як сховище документів, що є схожим зі сховищем «ключ-значення». Але на відміну від сховища «ключ-значення», документи, які зберігаються, мають визначену структуру та кодування даних. Деякі зі стандартних розповсюджених кодировок, що використовуються – це XML, JSON та BSON.

Однією з ключових відмінностей між сховищами «ключ-значення» та документоорієнтованим є те, що останнє включає метадані, які пов'язані зі вмістом, що зберігається. Це надає можливість робити запити на основі цього вмісту. Найпопулярнішими прикладами документоорієнтованих сховищ є CouchDB та MongoDB. CouchDB використовує JSON для зберігання даних, JavaScript в якості мови запитів з використанням MapReduce та HTTP для API. Дані та відношення не зберігаються в таблицях так, як це відбувається у традиційних реляційних БД, а за сутністю є набором незалежних документів. Той факт, що такі сховища працюють без схеми, спрощує задачу додавання полів до JSON-документа без необхідності попереднього заявлення про зміни.

Стовпчикове сховище. У стовпчикових NoSQL сховищах дані зберігаються у комірках, що сгруповані у стовпчики, а не строки даних. Стовпчики логічно групуються у стовпчикові сімейства. Стовп-

чикові сімейства можуть складатися з практично необмеженої кількості стовпчиків, які можуть створюватися під час роботи програми або під час визначення схеми. Читання та запис відбувається із використанням стовпчиків, а не строк.

Порівняно зі зберіганням даних у строках, як у більшості реляційних БД, переваги зберігання у стовпчиках полягає у швидкому пошуці/доступі та агрегації даних. Реляційні БД зберігають кожну строку як безперервний запис на диску. Різні строки зберігаються у різних місцях на диску, в той час як стовпчикові сховища зберігають всі комірки, що відносяться до стовпчика, як безперервний запис, що прискорює пошук/доступ.

Стовпчикові сховища використовують наступну модель даних:

- стовпчикове сімейство – структура, яка може легко групувати колонки та суперколонки;
- ключ – постійне ім'я запису. У ключів може бути різна кількість стовпчиків, тому сховище може розширюватися нерівномірно;
- простір ключів – визначає зовнішній рівень організації, як правило, ім'я прикладної програми/БД.
- стовпчик – має впорядкований список елементів – кортежів з іменами та значеннями.

Найвідомішими представниками стовпчикових сховищ є Google BigTable та HBase з Cassandra.

BigTable є високопродуктивним, стислим та пропрієтарним сховищем даних від Google. Воно має наступні атрибути:

- розрідженість – деякі комірки можуть бути порожніми;
- розподіленість – дані розділені між багатьма вузлами;
- постійність – дані зберігаються на диску;
- багатомірність – більш ніж одне вимірювання;
- співставлення – ключ та значення;
- відсортованість.

На стовпчики можна посилатися за допомогою стовпчикового сімейства.

Графове сховище. У графовому сховищі немає строгого формату SQL або представлення таблиць та стовпчиків, замість цього використовується гнучке графічне представлення, яке ідеально підходить для вирішення проблем масштабованості. Графові структури використовуються разом із ребрами, вузлами та властивостями, що забезпечує безіндексну суміжність. При використанні графового сховища дані можуть бути легко перетворені з однієї моделі в іншу (рис. 6).

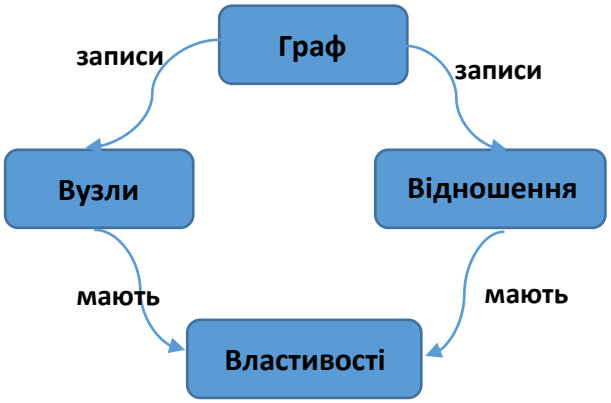


Рис. 6. Принципи використання графової моделі

- Такі сховища використовують ребра та вузли для представлення даних.
- Вузли пов'язані між собою певними відношеннями, які представлені ребрами між ними.
- Вузли та відношення мають деякі властивості.

Розмічений, спрямований, атрибутований мультиграф (рис. 7) – це граф, який містить вузли, які помічені певними властивостями та які мають зв'язки один з одним, що представлені спрямованими ребрами. Наприклад, зв'язок «Аліса знає Боба» виражена ребром з відповідними властивостями. Будь-який рейтинг «вам рекомендовано», представлений на різних сайтах, часто вираховується виходячи з того, як інші користувачі оцінили продукт. Графові БД відмінно підходять для вирішення такого типу задач.

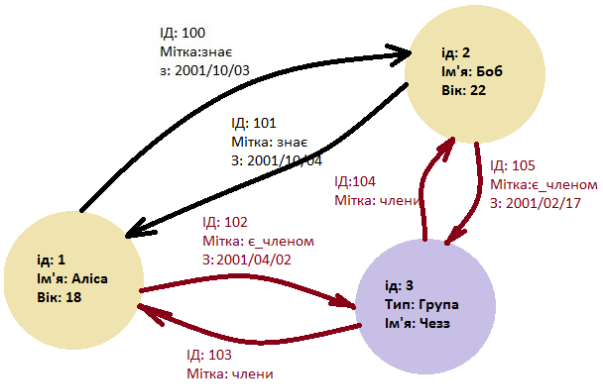


Рис. 7. Приклад атрибутованого мультиграфу

Прикладами найпопулярніших графових сховищ є InfoGrid та Infinite Graph. InfoGrid дозволяє з'єднувати множини ребер та вузлів, що спрощує представлення набора інформації зі складними взаємними посиланнями. InfoGrid пропонує два типи сховищ:

- MeshBase — підходить для автономного розгортання;
- NetMeshBase — підійде для великих розподілених графів та має додаткові можливості для взаємодії з іншими подібними сховищами.

Постачальники та ПЗ великих даних

В області розробки підходів до роботи з пам'яттю найбільшим чемпіоном є SAP зі своєю Hana платформою, але, слід зазначити, що зараз Microsoft та Oracle також вводять спеціальні опції для роботи з пам'яттю для своїх провідних баз даних. Постачальники ПЗ, що фокусуються на використанні аналітичних БД, включаючи Actian, HP Vertica та Teradata ввели спеціальні опції для співвідношення високих-ОЗУ-дисків, а також пропонують інструменти для розміщення конкретних даних у пам'яті для виконання ультра-швидкого аналізу. Прогрес, що має місце у зростанні пропускної здатності та обчислювальної потужності, вдосконалив й можливості потокової обробки та проведення аналізу в реальному часі.

До великих постачальників сервісів обробки даних [13] можна віднести IBM, Microsoft, Oracle, SAP, які пропонують все від ПЗ інтеграції даних та систем керування базами даних (DBMS) до ПЗ для бізнес-аналізу та аналітичної обробки, а також Hadoop опцій для роботи з пам'яттю та потокової обробки. Teradata більш вузько зосереджений на керуванні даними, та подібно Pivotal, він має тісні зв'язки з лідером аналітичного ринку SAS.

Багато постачальників пропонують реалізовані окремі опції хмарних технологій, але такі розробники, як 1010data та Amazon Web Services (AWS) використовують хмарну модель у повному обсязі в своєму ПЗ. З них двох Amazon має найширшу вибірку продуктів і є очевидним вибором для тих, хто працює з великими навантаженнями та зберігає велику кількість даних на AWS платформі. 1010data має високо масштабований сервіс БД та підтримує можливості управління інформацією, бізнес-аналіз та аналітику, що обслуговуються в стилі приватної хмари.

Hadoop довів свою користь та переваги у вартості там, де є екстремальними об'єм та різноманітність даних. На сьогодні це найбільш відомий та поширений програмно-апаратний комплекс для роботи з великими даними. Він виявився настільки гарним, що став фундаментом декількох комерційних реалізацій на його основі, а саме: Cloudera, MapR та Hortonworks, кожна з яких пропонує власний дистрибутив. На сьогодні всі постачальники традиційних BI-систем, як MicroStrategy або SAS, забезпечують інтерфейс з Hadoop. Виробники MPP-систем (масово-паралельних архітектур) у свою чергу забезпечують суттєво більш міцну інтеграцію з Hadoop, коли дані, що зберігаються і в Hadoop, і в реляційній СКБД, можуть оброблятися в одному SQL-запиті. Oracle, IBM, Teradata. Cloudera, Hortonworks та MapR, що також включили Hadoop до своїх продуктових лінійок, роблять все можливе, щоб перемістити Hadoop з високо-масштабованого зберігання даних та Map Reduce обробки у світ аналітики.

Менші постачальники такі як Actian, InfiniDB/Calpont, HP Vertica, Info-

bright та Kognitio, навпаки фокусуються загалом скоріше на аналітиці, ніж на обробці транзакцій.

Такі постачальники аналітики, як Alpine Data Labs, Revolution Analytics та SAS, працюють з платформами, які забезпечуються сторонніми постачальниками СКБД та дистриб'ютерами Hadoop, хоча, зокрема, SAS розмиває це розмежування зі зростаючою підтримкою для середовищ SAS-керованих рядів даних «у-пам'яті» та Hadoop. NoSQL та NewSQL СКБД фокусуються на високо-масштабованій обробці транзакцій, а не на аналітиці.

Взагалі програмні засоби для роботи з великими даними не заміщують решту інструментів обробки, бізнес-аналітики, візуалізації та прогнозування, а лише допомагають підтримати терабайти нових даних та спрямовують їх у потрібне русло.

Так, відповідно до аналітичних платформ для великих даних, деякі експерти вважають найбільш універсальною платформу Pentaho, а для вирішення задач машинного самонавчання, таких як, наприклад, кластеризація, класифікація, регресія та інші, краще підходять Mahout та Spark. Серед найбільш технологічних MPP – платформ спеціалісти виділяють Vertica та Teradata Aster. Останнім часом з'явилася множина платформ, які підтримують швидку аналітику для великих даних, наприклад, MemSQL або Splice Machine.

Окремої уваги заслуговує Intel платформа з відкритим кодом для Hadoop. Привабливість рішення Intel для Hadoop обумовлює також й фактор "апаратного забезпечення", а саме – оптимізація, що виконана Intel з урахуванням архітектури процесорів Xeon та специфіки роботи твердотільних накопичувачів з контролерами Intel, дозволяє досягти значного приросту продуктивності. Процесори Xeon прискорюють операції шифрування або дешифрування за алгоритмом AES (Advanced Encryption Standard), що реалізується за допомогою додаткового набору команд AES-NI (New Instruction). Окрім цього, платформа Intel для Hadoop також пропонує розширені можливості у галузі обробки потокових даних.

Різноманіття платформ для роботи з великими даних доповнюється величезною кількістю прикладних програмних продуктів, комерційних чи безкоштовних, для аналітичної обробки таких даних. Нижче наведений невеликий перелік найпоширеніших прикладів такого ПЗ:

- *Cluvio* – сучасна платформа аналітики даних, що дозволяє виконувати SQL запити, обробляти дані, візуалізувати результати та створювати гарні, інтерактивні дашборди за лічені хвилини. Підтримує потужне вбудовування, що дозволяє додавати аналітичні властивості до будь-якого веб-сайту або веб-застосунку.
- *IBM SPSS Statistics* дозволяє виявляти нові зв'язки між даними та будувати прогнози. Він дозволяє отримувати легкий доступ до даних, управляти та аналізувати набори даних, не маючи попереднього статистичного досвіду. Це дозволяє практично виключити довготривалу підготовку даних та швидко створювати, маніпулювати та розповсюджувати інформацію для прийняття рішень.
- *Qlik Sense Desktop* – безкоштовний продукт, що надає можливість інтерактивного створення звітів та дашбордів з діаграмами та графіками. Програма візуалізації спрощує аналіз даних та допомагає створювати інформовані бізнес-рішення швидше, ніж будь-коли раніше. Перетворення електронних таблиць у більш чіткі візуалізації робить процес аналізу простішим та швидшим для перегляду всіх користувачів.
- *Elasticsearch* – розповсюджений пошуковий та аналітичний движок на базі Apache Logstash, Kibana та Beats складають "Elastic Stack", розроблений фірмою Elastic. Також Elasticsearch забезпечується хостінг Elastic Cloud.
- *Cyfe* – бізнес-панель для управління даними компанії за допомогою звітів, попередньо побудованих віджитів тощо.
- *Forestpin Analytics* – платформа аналізу даних для знаходження нерівностей, кореляцій та дублювань за допомогою простого дашборда.

Висновки

Кількість підприємств, що використовують великі дані, безперервно зростає. Практика останніх років продемонструвала, що застосування результатів аналізу великих масивів даних може принести реальний ефект. Але, окрім переваг існує велика кількість проблем, вирішення яких вимагає застосування досить значних ресурсів.

Для систем, що отримують аналітичні дані в масштабі, близькому до реального часу, ключовими є вимоги не лише до продуктивності, але й до часу відгуку (наприклад, IBM каже про час відгуку, менший за мілісекунди). Це дуже обмежує вибір аналітичних платформ. Неможливо використовувати колосальні обчислювальні можливості Hadoop, якщо накладні витрати на ініціювання та завершення тривіальної MapReduce-програми складають десятки секунд. Забезпечити прийнятний час відгуку можуть або досить недешеві MPP-платформи (такі як Netezza, Teradata, Greenplum), або розподілені системи з розвиненою індексацією або високим рівнем резидентності даних в оперативній пам'яті.

Багато аналітичних систем все ще використовують реляційну модель даних, внаслідок чого вибір платформ обмежується такими рішеннями, як GridGain або Gigaspaces XAP. Для роботи з потоковими даними в режимі он-лайн були створені технології Storm, Spark Streaming та Akka. Але аналіз даних за допомогою SQL на Hadoop не дозволяє досягти того максимуму, який пропонує платформа.

Компанії обирають Hadoop, щоб збирати складні та різноманітні дані: історія відвідувань веб-сайтів, логи, дані про використання мобільних пристроїв й інформації з соцмереж та багато іншого. Цими даними складно оперувати у СКБД. Можна витягувати структуровані дані з Hadoop для SQL-аналізу, але більш перспективними є такі підходи як машинне самонавчання та інші, що дозволяють співвіднести нові дані зі вже накопиченою, проаналізованою та структурованою інформацією. BI та SQL системи досить добре себе проявили, але постійно виникають нові потреби та нові питання, що виходять за межі поточних можливостей. Уже не достатньо просто управляти даними. Окрім цього, компанії не можуть покладатися лише на аналітику, вони потребують також рішень зі сфери BI, системи збору та передачі оперативної інформації та інше. Межа між цими поняттями почала стиратися, а SAS, Alpine Data Labs та інші стали підтримувати кластеризовані серверні середовища, вимогливі до пам'яті та Hadoop.

Література

1. Великі дані: великі проблеми. Є.Є. Чехарин. Міжнародний електронний науковий журнал.
2. https://proglib.io/p/nosql-db-part-1/
3. Big Data: Survey, Technologies, Opportunities, and Challenges. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, ZakiraInayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz and Abdullah Gani. *The Scientific World Journal*. January 2014.
4. Sagiroglu S. and Sinanc D. Big data: are view. Proceedings of the International Conference on Collaboration Technologies and Systems (CTS'13), P. 42–47, IEEE, SanDiego, Calif, USA, May 2013.
5. Wang D. An efficient cloud storage model for heterogeneous cloud infrastructures. Procedia Engineering. 2011. Vol. 23. P. 510–515.
6. Bakshi K. Considerations for big data: architecture and approach. Proceedings of the IEEE Aerospace Conference. P. 1–7, BigSky, Mont, USA, March 2012.
7. Aho A.V. Computation and computational thinking. The Computer Journal. 2012. Vol. 55, N 7. P. 833–835.
8. Bhatnagar S.S.V. and Srinivasa S. Big Data Analytics, 2012.
9. Pastorelli M., Barbuzzi A., Carra D., Dell'Amico M. and Michiardi P. HFSP: size-based scheduling for Hadoop. Proceedings of the IEEE International Congress on Big Data (BigData '13), 2013. P. 51–59.
10. Katal A., Wazid M., and Goudar R.H. Big data: issues, challenges, tools and good practices. Proceedings of the 6th International Conference on Contemporary Computing (IC3'13). 2013. P. 404–409.
11. Big Data Analytics: A Literature Review Paper. Nada Elgendy and Ahmed Elragal. Con-

ference Paper in Lecture Notes in Computer Science · August 2014.
12. https://tproger.ru/translations/types-of-nosql-db/
13. https://www.forbes.com/sites/bernardmarr/2016/02/09/how-to-find-the-best-big-data-product-or-service-vendors/
14. http://datareview.info/article/analitika-v-rezhime-realnogo-vremeni-s-pomoshhyu-spark-sql/
15. https://habr.com/post/415939/
16. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

References

1. Big data: big problems. E. E. Cheharin. International Electronic Scientific Journal, ISSN 2307-2334.
2. https://proglib.io/p/nosql-db-part-1/
3. Big Data: Survey, Technologies, Opportunities, and Challenges. Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, ZakiraInayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz and Abdullah Gani. The Scientific World Journal · January 2014.
4. Sagiroglu S. and Sinanc D. "Big data: are view," in Proceedings of the International Conference on Collaboration Technologies and Systems (CTS'13), P. 42–47, IEEE, SanDiego, Calif, USA, May 2013.
5. Wang D. "An efficient cloud storage model for heterogeneous cloud infrastructures," Procedia Engineering. 2011. Vol. 23. P. 510–515.
6. K. Bakshi, "Considerations for big data: architecture and approach, "in Proceedings of the IEEE Aerospace Conference. P. 1–7. BigSky, Mont, USA, March 2012.
7. Aho A.V. "Computation and computational thinking," The Computer Journal. 2012. Vol. 55, N 7. P. 833–835.
8. Bhatnagar S.S.V. and Srinivasa S. Big Data Analytics, 2012.
9. Pastorelli M., Barbuzzi A., Carra D., Dell'Amico M. and Michiardi P. "HFSP: size-based scheduling for Hadoop," in Proceedings of the IEEE International Congress on Big Data (BigData '13), 2013. P. 51–59.
10. Katal A., Wazid M., and Goudar R.H. "Big data: issues, challenges, tools and good practices," in Proceedings of the 6th International Conference on Contemporary Computing (IC3'13). 2013. P. 404–409.
11. Big Data Analytics: A Literature Review Paper. Nada Elgendy and Ahmed Elragal. Conference Paper in Lecture Notes in Computer Science · August 2014.
12. https://tproger.ru/translations/types-of-nosql-db/
13. https://www.forbes.com/sites/bernardmarr/2016/02/09/how-to-find-the-best-big-data-product-or-service-vendors/
14. http://datareview.info/article/analitika-v-rezhime-realnogo-vremeni-s-pomoshhyu-spark-sql/
15. https://habr.com/post/415939/
16. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

Одержано 28.02.2019

Про автора:

Захарова Ольга Вікторівна,
кандидат технічних наук,
старший науковий співробітник.
Кількість наукових публікацій в
українських виданнях – 28.
http://orcid.org/0000-0002-9579-2973.

Місце роботи автора:

Інститут програмних систем
НАН України,
проспект Академіка Глушкова, 40.
Тел.: 526 5139.
E-mail: ozakharova68@gmail.com