УДК 004.415.2

Олещенко Л.М., к.т.н.

ТЕХНОЛОГІЇ BIG DATA АНАЛІТИКИ В РОЗПОДІЛЕНИХ СИСТЕМАХ ОБЧИСЛЕНЬ

Національний технічний університет України «КПІ імені Ігоря Сікорського»

oleshchenkoliubov@gmail.com

Проаналізовані існуючі технології та програмні засоби Big Data аналітики, виділено їх переваги та недоліки. Розглянуто інфраструктуру та статистичні методи аналізу Big Data. Проаналізовані технології розподіленої обробки, проблеми іденфікації та зберігання Big Data. На основі проведеного аналізу виділено основні проблеми Big Data аналітики, які у подальшому потребують вирішення

Ключові слова: Big Data, програмне забезпечення, розподілені обчислення, IoT, Pandas, Python, статистична обробка даних, бази даних, ЦОД, Kafka, Cassandra, MapReduce, Hadoop, Spark

Вступ та постановка проблеми

В сучасному світі дані постійно створюються споживачами Інтернет послуг, користувачами соціальних мереж, датчиками, реактивними двигунами, фондовими ринками та практично всім, що пов'язано з мережею. Ці дані зростають кількісно та змінюються у реальному часі. Аналіз даних також повинен здійснюватися в режимі реального часу під час збирання даних. Прийняття рішень на основі Big Data аналітики може значно підвищити рентабельність інвестицій для бізнесу [1]. Обсяг даних, які необхідно зберігати та аналізувати, продовжує розширюватися. Швидкість генерації даних не має ознак уповільнення. Різноманітність даних буде й надалі охоплювати нові області, які ніколи раніше не були доступні для аналізу.

Поява великих наборів даних вимагає застосування ефективних методів, технологій та інфраструктури для обробки даних та перетворення їх у діючу інформацію. Дані більше не можуть зберігатися на декількох машинах, або оброблятись одним інструментом.

Дослідники та аналітики шукають способи доступу та аналізу даних, які колись вважалися непридатними для використання. Розширені методи аналізу можуть бути використані на великих наборах даних, таких як текстова аналітика, машинознавство, прогнозування та аналіз статистичних даних. Ці знання необхідні для прийняття ефективних бізнес рішень.

Інфраструктура Big Data

Після збору даних відбувається їх фільтрування, тобто видалення небажаних даних, зміна невірних даних та заповнення відсутніх даних. Для фільтрування даних використовується програмний код. Це досягається шляхом пошуку критеріїв та роботи над даними, доки не залишиться жодних аномалій.

Після очищення даних їх можна спростити, проаналізувати та візуалізувати. За допомогою аналізу даних можна виявляти тенденції. Це часто призводить до нових запитів, які ще не були реалізовані. Наприклад, камера безпеки, яка стежить за кількістю злочинів на автостоянці, також може використовуватися для повідомлення водіям про кількість вільних місць для парковки.

Парадигма Big Data складається з розподілу систем даних по горизонтально пов'язаних між собою незалежним ресурсам для досягнення масштабованості, необхідної для ефективної обробки великих наборів даних. Горизонтальна масштабованість відрізняється від вертикальної масштабованості тим, що не намагається використовувати додаткову потужність або пам'ять існуючих машин. Ці інфраструктури дозволяють багатьом ко-

ристувачам одночасно та безпечно отримувати доступ до даних.

З поширенням автоматизації бізнесу та зростанням веб-додатків та машинно-генерованих даних, аналітика стає дедалі складнішою для керування лише за допомогою рішень Relational Database Management System (RDBMS). Понад 90% даних, які існують сьогодні, були створені лише протягом останніх кількох років. Цей збільшений обсяг протягом короткого періоду часу є властивістю експоненційного зростання. Такий великий обсяг даних важко обробляти та аналізувати за короткий проміжок часу.

Замість великих баз даних (БД), що обробляються великими та потужними комп'ютерами і займають величезний дисковий простір (вертикальне масштабування), розподілена обробка даних приймає великий обсяг даних і розбиває її на менші фрагменти. Ці менші обсяги даних поширюються в багатьох місцях для обробки багатьма комп'ютерами з меншими процесорами. Кожен комп'ютер у розподіленій архітектурі аналізує свою частину Big Data. Більшість розподілених файлових систем призначені для невидимих для клієнтів програм. Усі користувачі бачать один і той самий перегляд файлової системи і можуть одночасно отримувати доступ до даних з іншими користувачами. У бізнесі багато важливих проблем неможливо вирішити за допомогою простого SQL-запиту і вимагати більш складного аналітичного процесу. Для цього використовуються більш потужні мови програмування аналізу даних, такі як R або Python.

Різні програми створюють файли в різних форматах, які не завжди сумісні один з одним. З цієї причини потрібен універсальний формат файлів. Значення файлів, розділених комами (Comma-separated values – CSV), є типом файлу з відкритим текстом, наведеним у RFC 4180. JSON і XML також є текстовими файлами, які використовують стандартний спосіб подання записів даних. Ці формати файлів сумісні з широким колом програм. Перетворення даних у єдиний формат є цінним способом об'єднання даних з різних джерел.

Веб-скрепінг (Web scraping) використовується для видобування таких даних як списки нерухомості, дані про погоду, дослідження та порівняння цін тощо [2]. Багато великих постачальників веб-послуг, таких як Facebook, надають стандартизовані інтерфейси для автоматичного збору даних за допомогою API. Найбільш поширеним підходом є використання RESTful інтерфейсів прикладних програм (API). RESTful API використовує HTTP як протокол зв'язку та JSON-структуру для кодування даних. Інтернет-сайти, такі як Google і Twitter, збирають велику кількість статичних та часових даних. Знання API для цих сайтів дає змогу аналітикам отримувати доступ до великих обсягів даних, які постійно створюються в мережі Інтернет. Інтернет речей (IoT) використовує датчики для створення даних (датчики температури та вологості, датчики від смартфонів до автомобілів, реактивних двигунів, побутової техніки тощо).

Вилучення, перетворення та завантаження (extract, transform, load – ETL) – це процес збирання даних з різних джерел, перетворення та завантаження даних у БД. Дані однієї компанії можна знайти у документах Word, електронних таблицях, простому тексті, PowerPoints, електронних листах та файлах PDF. Ці дані можуть зберігатися на різних серверах, які використовують різні формати.

Виділяють три основні кроки процесу ETL [3]:

1. Дані вибираються з декількох джерел.

2. Після вилучення даних вони мають бути перетворені. Трансформація даних включає збирання, сортування, очищення та об'єднання даних.

3. Перетворені та оброблені дані завантажуються в БД для запиту.

Етап вилучення збирає потрібні дані з різних джерел та робить їх доступними

для обробки завдяки їх перетворенню в єдиний формат.

ELT (extract, load, transform) – це аналогічна концепція, основна відмінність полягає в тому, що вона не обмежується даними, які можуть зберігатися в традиційному сховищі даних або можуть бути визначені заздалегідь. Процес високого рівня показаний на рис.1 [4].

Об'єднання даних з сервера NOSQL та DB Oracle дасть дані у різних форматах. Ці дані повинні бути перетворені в єдиний формат. Крім того, дані повинні бути перевірені, щоб забезпечити необхідний тип інформації (значення). Це робиться за допомогою правил перевірки. Якщо дані не відповідають правилами перевірки, вони будуть відхилені. Іноді ці відхилені дані виправляються, а потім перевіряються. Дані IoT часто передаються в режимі реального часу, уворюючи часові ряди даних. Наприклад, це може бути температура повітря, що вимірюється на метеостанції кожну хвилину, дані про споживання електричної енергії будинку, що повідомляються інтелектуальним термометром кожні 15 хвилин. Під час експерименту чи аналізу можуть спостерігатися різні змінні та пов'язані з ними значення.

Статистичні методи аналізу Big Data

Основними поняттями Big Data аналізу є:

- кластер – для пошуку схожих груп спостережень;
- асоціація – для пошуку спільного входження значень для різних змінних;
- регресія – для кількісної оцінки відношень між варіантами однієї чи кількох змінних.

У процесі machine learning комп'ютерне ПЗ надає та забезпечує власний набір правил для аналізу.

Pandas – це бібліотека Python, яка додає високопродуктивні структури даних та інструменти для аналізу великих наборів даних. Структури даних Pandas включають структури серії та структури даних. Дані-кадри – це основна структура Pandas, вони використовуються найчастіше. Фрейми даних можуть мати додаткові індекси та стовпці, які є мітками для рядків і стовпчиків. Фрейми даних створюються з ряду інших структур даних та зовнішніх файлів, таких як csv. Рядками та стовпцями можна маніпулювати різними способами, оператори доступні для виконання математичних, рядкових та логічних перетворень у

Рис. 1. Процес вилучення, завантаження та перетворення даних [4]

вміст інформаційного фрейму.

Набори даних мають несумісності, такі як неналежне або невідповідне форматування, небажану інформацію та пропущені частини даних. Очищення даних може включати в себе видалення відсутніх та небажаних значень або зміну формату значень, щоб зробити їх послідовними. Іншою поширеною причиною NaNs є повторна індексація у наборі даних. Відсутні значення можуть приймати різні форми залежно від типу даних.

Для ідентифікування листа як спам; класифікації новин про технології, політику чи спорт; перевірки тексту, що виражає позитивні або негативні емоції та для розпізнавання осіб використовуються ймовірнісні класифікатори Байєса, засновані на теоремі Байєса з незалежними припущеннями між функціями. Логістична регресія використовується для моделювання біноміального результату з однією або декількома змінними. Вона вимірює взаємозв'язок між категоріально-залежною змінною і однією або декількома незалежними змінними шляхом оцінки ймовірностей з використанням логістичної функції. Регресія використовується для аналізу кредитного рейтингу; вимірювання успішності маркетингових кампаній; прогнозування доходів певного продукту.

Перетворення даних у цінну статистику вимагає обчислень та значного обсягу пам'яті. Різні архітектури IoT мають різні підходи щодо того, де і коли дані обробляються та зберігаються. В архітектурі Device-Network-Cloud усі точки даних, зібрані датчиками, які входять до підключеного пристрою, надсилаються безпосередньо у хмару для зберігання та обробки. Це відбувається з більшістю носіїв, які використовуються для відстеження фізичної активності. Дані, зібрані пристроєм, надсилаються без хмарної мережі в хмару. Там вони трансформуються за допомогою описової аналітики та представляються користувачеві у веб-профілі. Ця архітектурна модель проста, але не масштабована. Коли кількість датчиків збільшується разом із кількістю отриманих точок даних або коли для обробки даних потрібен менший час відгуку, це ситуації, коли дані потрібно обробляти ближче до місця, де вони створюються. Тут використовується архітектура Device-Gateway-Network-Cloud. Залежно від програми, дані можуть оброблятися майже відразу після їх створення, дуже близько джерела створення на шлюзі або інших проміжних місцях в мережі. Незалежно від використовуваної архітектури IoT, більшість або всі дані, збиратимуться та зберігатимуться у хмарі, де доступна потенційно нескінченна обчислювальна потужність та об’єм пам'яті.

Розподілена обробка Big Data

На початку 2000-х років після появи Web 2.0 та систем електронної комерції, стало зрозуміло, що реляційні БД не можуть відповідати обсягам та швидкості пошукових запитів в мережі Інтернет. Щоб задовольнити цей попит, Google розробила розподілену файлову систему GFS та алгоритм розподіленої паралельної обробки MapReduce.

NoSQL БД можуть використовувати підхід сховища ключових значень замість реляційного на основі таблиці. Інші БД NoSQL зберігають дані як структуровані документи у форматах XML або JSON. NoSQL БД набагато швидші, ніж реляційні БД, вони можуть імпортувати неструктуровані дані. NoSQL БД розроблені для масштабування по горизонталі, тобто можливості зберігання та керування можуть бути збільшені шляхом додавання інших кластерів до машин.

Hadoop використовує розподілену файлову систему, яка розширюється, додаючи більше комп'ютерів та жорстких дисків. Зберігання збільшується за рахунок додавання серверів, які запускаються на другому обладнанні [5].

Коли Google потребує додаткової потужності обробки, використовується Google Modular Data Center, індивідуальне рішення, яке додає ще 1000 процесорів, підключених до мережі. Процесори працюють разом у кластері завдяки технології Google MapReduce. Система Hadoop Big Data також базується на розподіленій обробці MapReduce. Додавання нових серверів або вузлів до кластера Hadoop не тільки додає додатковий дисковий об'єм, але й забезпечує додаткову обробку процесора.

Розподілена файлова система Hadoop (HDFS) – це файлова система, де Hadoop зберігає дані. HDFS не замінює файлову систему Linux на окремих серверах, замість цього знаходиться на верхній частині кластера серверів в одній файловій системі, що охоплює весь кластер. HDFS зберігає дані з використанням, як мінімум, трьох серверів DataNode. HDFS управляє інформацією в межах кластера від централізованої координації сервера NameNode. Сервер NameNode відстежує збереження даних на різних серверах DataNode. Коли дані введені в систему, вони імпортуються до NameNode, а NameNode потім ділить дані в 64Mb сегменти, які потім дублюються по три або більше DataNodes в залежності від конфігурації. Ця надмірність забезпечує відмовостійкість, схожу на дзеркальний масив RAID, в тому, що якщо один DataNode буде невдалим, DataNode сервер може бути замінений, а файлова система, і дані будуть відновлені з дубльованих DataNodes.

Розподілена обробка MapReduce використовується для розпаралелювання алгоритмів завдяки великій кількості серверів та можливості обробки великих наборів даних. MapReduce ділить обробку даних на дві фази: фазу відображення, в якій дані розбиті на сегменти та можуть бути оброблені за допомогою окремих потоків, і працюють на різних машинах; і фази зменшення, яка поєднує в собі вихід з декількох мапперів у кінцевий результат.

Проблеми іденфікації та зберігання Big Data

Big Data приходять з різних джерел. Велика частина цих даних надходить у потоковому режимі. Масштабованість також є проблемою. Чим більше пристроїв стають з'єднаними, тим більше даних повинні надходити в організм. Якість даних також може бути проблемою. Дані можуть бути структуровані, неструктуровані, мати складні формати.

Одним із інструментів, доступних для збору і переміщення великих обсягів даних з декількох джерел в сховище даних є розподілений програмний брокер повідомлень Kafka [6]. Kafka використовується в режимі реального часу для передачі потокових даних між різними системами і додатками. Він також використовується для перетворення і реагування на потоки даних через додатки в режимі реального часу. Kafka працює на одному або декількох серверах в якості кластера. Сервери в кластері називаються брокерами. Кластери зберігають потоки записів в спеціальні угруповання – теми (topics). Кожен запис має ключ, значення і мітку часу. Kafka працює аналогічно розподіленій БД. Повідомлення, написані у Kafka, розповсюджуються на багато серверів і записуються на диск. Сховище даних може тривати вічно, поки це необхідно. Через це розподілене проектування, Kafka має високу доступність, підтримує автоматичне відновлення, і є стійким до збоїв мережі. Кожна тема (topic) складається з набору журналів, які називаються розділами. Виробники додають ці журнали і споживачі можуть читати журнали, коли їм потрібно. Теми є даними, які реплікуються багатьма брокерами для досягнення відмовостійкості. Kafka має високу пропускну здатність. Це відбувається за рахунок того, що багато функцій розміщені на виробниках і

споживачах. Це робить Kafka бажаним інструментом для багатьох платформ IoT. З величезною кількістю даних, що надходять від датчиків та інших пристроїв в режимі реального часу і бажанням аналізувати ці дані без втрати інформації, Kafka може регулювати цей прийом і надавати дані для декількох споживачів одночасно.

Cassandra є системою керування БД, що відноситься до класу NoSQL-систем і розрахована на створення високомасштабованих і надійних сховищ величезних масивів даних, представлених у вигляді хеша. NoSQL БД підтримує горизонтальне масштабування і забезпечує більший контроль доступності. Cassandra повністю розподілена і може бути розгорнута по всьому світу. Cassandra використовує файлову систему CFS. Кожен вузол в кластері – це реалізація рівноправних вузлів. Кластери зберігають дані в реальному часі і аналітичні операції виконуються на цих даних. Розглянемо переваги використання CFS у порівнянні з HDFS [7].

Чим більше вузлів і кластерів, тим краща доступність. Це також покращується за рахунок підвищення коефіцієнту реплікації, який визначає, скільки вузлів отримуватимуть репліковані дані. Для CFS ніякі спеціальні сервери та мережеві пристрої не потрібні.

Вузли, або кластери, навіть цілі центри обробки даних (ЦОД) можуть дати збій і дані по вузлах і кластери будуть доступні в інших місцях. Усі дані реплікуються до вузлів пошуку. Це дозволяє аналітикам у режимі реального часу виконувати обчислення, які будуть завершені одночасно, не впливаючи одне на одне. Кластери прості в установці і запускаються протягом кількох хвилин. CFS може працювати з єдиною БД та з кількома ЦОД. Інструменти доступні, щоб кожен ЦОД мав локальну копію всіх даних в БД. Аналітику можна виконувати в кількох ЦОД одночасно.

Spark – це система розподіленої обробки даних, що використовується для маніпуляцій з Big Data. Spark використовує кешування в пам'яті для досягнення високої продуктивності, обмежуючи читання і запис з диску. Spark може використовувати HDFS, що дає більш високу продуктивність, ніж MapReduce. Spark підтримує різні мови програмування, такі як Python, Scala, R і Java та структуровані БД SQL, включає в такі бібліотеки для побудови додатків, як SparkSQL, SparkMLib, SparkGraphX.

Spark використовує розподілені набори даних (RDDs) для зберігання вхідних, вихідних і проміжних даних в пам'яті, а не на диску. RDDs є стійкими, оскільки вони відстежують історію, і в разі збою вони легко відновлюються. Вони поширюються на багато вузлів в кластері. Це забезпечує надмірність, а також більш високу ефективність за рахунок паралельної обробки. Коли запит зроблено з програми, Spark створює і завантажує дані в RDDs. Дані можуть надходити з будь-якого джерела, включаючи HDFS, CFS, AWS S3 або SQL.

Spark може працювати у верхній частині екземпляру Hadoop, використовуючи HDFS для зберігання і управління кластером. Spark може не використовувати Hadoop взагалі. Інші рішення для зберігання даних можуть бути такі як CFS або AWS S3 [8].

Розглянемо основні переваги Spark.

1. Потокові дані – Spark здійснює обробку великих наборів даних (від мобільних пристроїв, соціальних мереж, датчиків IoT) у режимі реального часу.

2. Неоднорідні дані – багато рішень Big Data отримують дані з різних джерел. Через гнучкість і здатність виходити за межі цих неоднорідностей, Spark є хорошим вибором для вирішення даної проблеми.

3. Spark містить вбудовану бібліотеку машинного навчання.

4. У пам'яті обробка дозволяє Spark повертати результати обчислення набагато швидше, ніж MapReduce. Це

важливо у всіх ситуаціях, коли результати необхідно отримувати у режимі реального часу.

Висновки

Аналітика, проведена на великих наборах даних, може надати нові можливості і непередбачені тенденції, забезпечуючи краще представлення клієнтів і ринок у цілому. Точна аналітика клієнтів, виявлення випадків шахрайства та аналіз ризиків – все це вигоди від Big Data аналітики. Ці складні обчислення вимагають не лише великих запасів даних, але й малу затримку, високу пропускну спроможність обробки потоку. На сьогодні основними інструментами та технологіями, які використовуються для Big Data аналітики є Python, R, Scala, Apache Spark, Hadoop, MapReduce, Cassandra, Kafka, алгоритми пошуку даних, комп'ютерне навчання, статистичні методи, NoSQL.

На основі проведеного огляду програмних засобів та технологій Big Data аналітики було виділено наступні проблеми, які у подальшому потребують вирішення:

1. безпека даних – чим більше даних переміщається в хмару, тим більше інформації розміщується у каналах глобальної мережі, для Big Data важко забезпечити аутентифікацію, доступ та облік;

2. неструктурованість даних – неструктуровані дані важко знаходити та аналізувати;

3. введення і виведення – великі проекти, які виробляють дуже велику кількість даних, повинні обмежувати кількість запитів вводу / виводу.

Входячи з виділених проблем, виникає гостра необхідність у створенні нових інструментів, способів зберігання та обробки Big Data для перетворення необроблених даних у цінну інформацію, для подальшого їх аналізу та прийняття ефективних управлінських рішень.

Розроблювані рішення повинні бути безпечними, реплікованими, мати високу відмовостійкість та можливість масштабування.

Список літератури

1. BIG DATA: Інноваційні можливості підвищення прибутковості агробізнесу // [Електронний ресурс] – Режим доступу: http://www.agro-business.com.ua/ideii-i-trendy/8383-big-data-innovatsiini-mozhlyvosti-pidvyschennia-prybutkovosti-agrobiznesu.html

2. Web Scraping Tools to Extract Online Data // [Електронний ресурс] – Режим доступу: https://www.hongkiat.com/blog/web-scraping-tools/

3. David Haertzen. ETL Tools // The Analytical Puzzle: Profitable Data Warehousing, Business Intelligence and Analytics. – Technics Publications, 2012. – 346 p.

4. ETL or ELT and the Use Case // [Електронний ресурс] – Режим доступу: https://www.linkedin.com/pulse/etl-elt-use-case-mich-talebzadeh-ph-d-/

5. What Is Apache Hadoop? // [Електронний ресурс] – Режим доступу: http://hadoop.apache.org/

6. Apache Kafka. A distributed streaming platform // [Електронний ресурс] – Режим доступу: https://kafka.apache.org/

7. Apache Cassandra // [Електронний ресурс] – Режим доступу: http://cassandra.apache.org/

8. Apache Spark. A fast and general engine for large-scale data processing // [Електронний ресурс] – Режим доступу: https://spark.apache.org/

Статтю подано до редакції 11.12.2017