УДК 004.942

СИСТЕМА ОБРОБКИ ІНФОРМАЦІЇ СТАНЦІЇ ОРЕНДИ ВЕЛОСИПЕДІВ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

О.В. Петрунів, Ю.Є. Кинаш, О.Я. Різник, Н.О. Кустра, В.М. Мищишин

Національний університет «Львівська політехніка»
вул. Степана Бандери 12, Львів, 79013, Україна

У статті розглядаються і аналізуються сучасні методи обробки інформації станцій оренди велосипедів. Показано, що оптимальним рішенням для створення системи, яка буде отримувати інформацію в реальному часі є використання Big Data рішень. Запропонована система станції оренди велосипедів надсилатиме та опрацьовуватиме інформацію в режимі реального часу.

Програмне рішення опрацювання інформації пропонованої системи в режимі реального часу написано мовою Scala. Застосування пропонованої системи дає можливість створювати повідомлення про потреби станцій у велосипедах або вільних місцях, що дозволить покращити ведення бізнесу. Розроблена система дозволяє будувати звітні таблиці за результатами діяльності конкретних станцій протягом певного періоду часу.

Показано, що результати фільтруються відповідно до часових параметрів та сортуються відповідно до кількості вільних місць. Отримані результати роботи системи відображаються візуально на екрані та зберігаються у файлах на диску.

Ключові слова: *big data, обробка інформації в режимі реального часу, паралельне програмування, Spark, Scala, Kafka.*

Постановка проблеми. Значне зростання обсягів інформації призводить до ускладнення її оброблення звичними способами. Саме тому і набирає популярності такий напрямок як Big Data. Підходи, що тут використовуються дозволяють значно легше обробляти великі об'єми інформації використовуючи для цього кластери комп'ютерів і паралельну обробку.

Бізнес все частіше для своєї діяльності починає потребувати big data рішень. Вирішення проблем опрацювання значних обсягів даних звичайними підходами вже не відповідає сучасним умовам.

В статті використано Big Data підходи для обробки потокової інформації, яку будуть надсилати станції оренди велосипедів в режимі реального часу. Ця інформація буде статус-звітом, про стан станції в конкретний момент часу. Кожна станція, як правило, відправляє цей звіт в різний час, і таких звітів може формуватися дуже багато за день. Традиційні рішення та підходи за таких умов не є ефективними.

Аналіз останніх досліджень та публікацій. Великі дані є такого об'єму, котрий не можна зберігати і опрацьовувати за використання традиційних комп'ютерних рішень [1]. При цьому не існує сталої класифікації, від якого об'єму даних їх можна вважати великими.

Класифікуються великі дані як:

- Структуровані, що є різноманітними таблицями, базами даних і т. п.
- Напів структуровані, що не мають якоїсь сталої структури, наприклад, документи, листи, файли логів і т. п.
- Не структуровані, що не мають жодної структури, наприклад, відео, фото, аудіозаписи і т. п.

Відповідно до виду даних існують різні підходи до їх обробки. Щоб визначити, що дані вже можуть бути віднесені до великих існують відповідні характеристики для великих даних. Основними є 5 характеристик [2] і всі вони починають з літери «V». Англійською ці характеристики є наступними: volume (кількість даних), velocity (як часто нові дані створюються і потребують збереігання), variety (наскільки дані є різноманітними), veracity (наскільки дані є правдивими) і value (наскільки вони є значущими).

Якщо дані задовільняють наведеним вище характеристикам, то вважаються «великими» і підхід до їхньої обробки змінюється.

Першою програмною моделлю для обробки «великих» даних була Map Reduce [3]. Вона була розроблена Google і дозволяє здійснювати паралельну обробку великих масивів даних на кластерах, що складаються з не надто потужних та недорогих комп'ютерів. Виконувались ці програми на Hadoop кластерах, і там для цього був спеціальний ресурсний менеджер, який розприділяв програму до даних і видавав необхідні обчислювальні ресурси для неї. Основною суттю Map Reduce є поділ обробки на дві фази, які відповідають двом частинам назви моделі.

На першій фазі, яка називається map, програма проходиться по масиву даних і будує пари ключ значення, які є результатом виконання цієї фази. На вхід другої фази приходить набір ключ і всі значення, які були зібрані для цього ключа на першій фазі. Далі відбувається так званий reduce, результатом якого є пари ключ – значення, при чому значення обраховується на цій фазі.

В свій час Map Reduce була ефективною, проте писати такі програми було важко і не зручно. Саме тому з'явився новий кращий і зручніший спосіб обробки даних – Apache Spark, що представляє швидкий і ефективний рушій, для обробки великих даних. Основною різницею в обчисленнях було те, що дані Spark зберігає в оперативній пам'яті (рис. 1.).

Рис.1. Відображення потоку даних при обробці Spark

Своєю чергою Map Reduce кожну трансформацію записує на диск, а потім знову читає і так поки не завершиться програма (рис. 2).

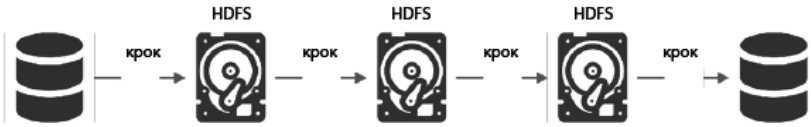

Рис.2. Відображення потоку даних при обробці Hadoop Map Reduce

Така концептуальна різниця привела до того, що програми на Spark виконуються в 10-100 разів швидше у порівнянні з Hadoop Map Reduce [4].

Перевагою Spark є не лише швидкість, а й функціонал, простота і зручність написання програм. Spark швидко набув популярності і майже повністю відтіснив Map Reduce. Саме тому використано Spark для цього проекту.

Мета дослідження – розроблення алгоритму та програмного рішення для систем автоматизації обробки текстової інформації з використанням Spark.

Виклад основного матеріалу дослідження. Для збереження «великих» даних використовують спеціальні формати, які дозволяють розбивати дані на так звані партиції, для того аби їх було зручніше використовувати при розподіленому обчисленні.

Найпростішим форматом є csv. Такий файл представляє собою просто набір даних, що розділені комою, при чому кома тут відділяє одну колонку від іншої. З цим форматом просто працювати, але він не підходить для зберігання великої кількості інформації, бо не має вбудованих методів стискання. Окрім нього є ще багато інших форматів. Серед основних можна виділити два, що є найбільш популярними [3] – це parquet і avro.

Parquet є колонко орієнтованим бінарним форматом файлу. Схема до файлу розташована в його кінці, а сам файл містить значення для наборів рядків. Такий формат підтримує стискання. Його легко читати з використанням Spark [4], оскільки для цього не потрібно жодних додаткових бібліотек.

Avro є форматом файлу, який орієнтований на рядки. Схема є розташованою спочатку файлу. Цей формат підтримує дуже компактне стиснення файлів. Для читання цього формату вже є необхідними сторонні бібліотеки, які поставляються компанією Apache.

Система, що розроблена в цьому проекті буде обмінюватись подіями з використанням системи, яка називається Apache Kafka або просто Kafka. Ця система призначення для обміну інформацією про події в реальному часі.

Kafka надає можливість робити наступні речі [5]:

- записувати і читати потоки подій;
- зберігати ці потоки використовуючи реплікацію для забезпечення надійності;
- обробляти потоки подій.

Kafka складається з таких складових як broker і topic. Broker виступає сервером Kafka, на якому запущена сама програма Kafka, а topic є місцем в котре записуються події, за аналогією до документу або папки.

Також, окрім цього є дві основні сутності Producer і Consumer. Перша записує інформацію в topic, а друга підписується на цей topic і читає інформацію

з нього. Їх може бути декілька, але якщо сутностей типу consumer більше ніж партицій в topic, то вони будуть лише висіти в пам'яті.

Kafka є хорошим рішенням, але часто використовують її обгортку confluent, що є надбудовою, котра має основну перевагу регістр схем.

Ця платформа дозволяє закидувати в topic лише ту інформацію, яка відповідає схемі [6], тобто на етапі розробки не потрібно буде валідувати повідомлення, яке було отримане з топіка, на випадок того чи відповідає воно схемі. Схема тут зберігається в регістрі схем і відсилається туди лише раз, а так як формат avro є дуже економним в плані розміру, то це дозволяє зменшити навантаження на мережу і пересилати дані набагато швидше.

Проблемою при розробці програм, що залежать від деяких зовнішніх налаштувань системи, різних програм і бібліотек є те, що при застосуванні таких програм необхідно виконувати додаткові встановлення. Це потребує витрачання додаткового часу. Проблему вирішує Docker [7], що є контейнеризатором, який дозволяє запакувати програми і їх оточення та створити так званий контейнер, який потім за допомогою програми docker можна легко розгорнути на іншому комп'ютері. На відмінну від віртуалізації, яка використовує завідомо визначені ресурси комп'ютера, контейнери використовують лише рівно стільки скільки їм потрібно. У роботі використано docker для розгортання confluent, оскільки він містить багато компонентів.

Результати досліджень. В статті розроблено систему, яка отримує інформацію зі станцій оренди велосипедів в режимі реального часу і на її основі створюються повідомлення про потреби станцій у велосипедах або вільних місцях. Це дозволяє покращити ведення бізнесу. Окрім цього створюються звітні таблиці стану справ на тій чи іншій станції за певний період.

Пропонована система складається з двох основних частин: streaming і batch. Програмний код написано мовою Scala [8]. В першій частині є дві програми. Перша бере дані зі станцій оренди велосипедів і відправляє в Kafka topic під назвою «station_info». Друга програма зчитує topic в режимі реального часу та виконує обчислення. У результаті отримаємо дані, що є збережені в Delta Lake Storage, котрий може бути розташованим локально на ПК або на Azure Cloud. Ці дані будуть розподілені по id конкретної станції. Основний результат роботи програми полягає у побудові, так званих, алертів (повідомлень про певні проблеми на конкретній станції), котрі відправляються в окремий Kafka topic з назвою «station_alerts», де в режимі реального часу можна буде бачити які проблеми є на конкретній станції та вирішувати їх.

В batch частині системи відбувається зчитування даних з Delta Lake Storage і будуються на їх основі узагальнені дані за день, тиждень, місяць або рік топів станцій з найбільшою і найменшою середньою кількістю вільних місць на станціях. Результати роботи цієї програми буде збережено у вигляді csv таблиць. Перші декілька записів, кількість яких також вводиться, як параметр програми, відправляються в окремий Kafka топік під назвою «top_station_by_docks_result».

Результати фрагменту роботи програми про відіслані в Kafka topic повідомлення зображені на рис. 3.

key	value.station_i	value.num_bike	value.num_doc	value.last_rep	value.num_ebik	value.station_stat
station_2	station_2	26	2	1617942107	2	active
station_4	station_4	21	6	1617930326	3	active
station_2	station_2	1	2	1617959498	27	active
station_4	station_4	3	11	1617911098	16	out_of_service
station_2	station_2	25	4	1617934023	1	active
station_3	station_3	10	6	1617961834	14	active

Рис. 3. Kafka topic з повідомленнями про статус станцій

Наступна потокова програма зчитує дані з топіку, що були записані попередньою. Дані записуються, а потім обробляються для визначення чи кількість велосипедів або вільних місць є менша, ніж мінімально допустима. Окрім цього, враховується можлива затримка повідомлень. Результати фрагменту роботи програми можна побачити на рис. 4.

key	value
bikes	At stations station_1: critically insufficient bikes
docks	At stations station_4: critically insufficient docks
bikes	At stations station_4: critically insufficient bikes
docks	At stations station_1: critically insufficient docks
electric bikes	At stations station_1: critically insufficient electric bikes
docks	At stations station_3: critically insufficient docks

Рис. 4. Kafka topic для відсилання алертів

В подальшому програма бере збережені попередньо дані і обробляє їх. Логіка є наступною: спочатку всі дані усереднюються по певному часовому вікну, щоб вирівняти їх один відносно іншого і обраховується середня кількість вільних місць для велосипедів. Потім дані фільтруються, щоб зібрати тільки ті, які вказані в параметрах, тобто дані за день, тиждень, місяць чи рік. Після цього дані сортуються по кількості вільних місць і зберігаються на диску. Інша частина програми формує повідомлення і відправляє їх в Kafka topic.

Використання системи дає змогу отримати результат для топ максимально та мінімально порожніх станцій за день, тиждень чи місяць. Отримані дані не лише виводяться на екран, а й зберігаються на диску у файлах.

Висновки. В статті запропоновано систему, що допомагає бізнесу з оренди велосипедів вчасно реагувати на критичні ситуації на станціях і вирішувати їх в режимі реального часу. Ця система розроблена, як Big Data рішення. Програмний код реалізовано мовою Scala, що забезпечує можливість розгортання її на кластері, що є перевагою у випадку зростання кількості станцій і розвитку бізнесу.

Для застосування пропонованого алгоритму розроблено три програмні модулі автоматизованої обробки інформації з використанням Big Data рішенн. Розроблена система працює з інформацією, що вже є збереженою, та з тією, що надходить в режимі реального часу.

Розроблена система дає змогу моніторити поточний стан технічних станцій на екрані та зберігати інформацію на дисках.

Список використаних джерел

1. Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data // Видавництво: John Wiley & Sons, Inc., 2015 р., 410 с.
2. The 5 V's of Big Data // [Електронний ресурс] Режим доступу: https://www.flydata.com/blog/5-vs-of-big-data.
3. Hadoop: The Definitive Guide, Tom White // Видавництво: O'Reilly Media, 2015 р., 756 с.
4. Spark in action, Petar Zečević and Marko Bonaći // Видавництво: Manning Publication, 2016 р., 472 с.
5. Apache Kafka Introduction // [Електронний ресурс] Режим доступу: https://kafka.apache.org/intro.
6. What is Confluent Platform? // [Електронний ресурс] Режим доступу: https://docs.confluent.io/home/platform.html
7. Docker // [Електронний ресурс] Режим доступу: https://docs.docker.com/
8. The Scala Programming Language // [Електронний ресурс] Режим доступу: https://www.scala-lang.org/

REFERENCES

1. Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data (2015). // Vydavnyctvo: John Wiley & Sons, Inc., 410 s. (in English)
2. The 5 V's of Big Data // [Elektronnyj resurs] Rezhym dostupu: https://www.flydata.com/blog/5-vs-of-big-data. (in English)
3. Hadoop: The Definitive Guide, Tom White (2015). // Vydavnyctvo: O'Reilly Media, 756 s. (in English)
4. Spark in action, Petar Zečević and Marko Bonaći (2016). // Vydavnyctvo: Manning Publication, 472 s. (in English)
5. Apache Kafka Introduction // [Elektronnyj resurs] Rezhym dostupu: https://kafka.apache.org/intro. (in English)
6. What is Confluent Platform? // [Elektronnyj resurs] Rezhym dostupu: https://docs.confluent.io/home/platform.html (in English)
7. Docker // [Elektronnyj resurs] Rezhym dostupu: https://docs.docker.com/ (in English)
8. The Scala Programming Language // [Elektronnyj resurs] Rezhym dostupu: https://www.scala-lang.org/ (in English)

DOI 10.32403/2411-9210-2021-2-46-36-42

INFORMATION PROCESSING SYSTEM OF BICYCLE RENTAL STATIONS IN REAL TIME

O. Petruniv, Y. Kynash, O. Riznyk, N. Kustra, V. Myshchyshyn

Lviv Polytechnic National University
12, S.Bandera St., Lviv, 79013, Ukraine
yuk.itvs@gmail.com

The article considers and analyzes modern methods of information processing of bicycle rental stations. It is shown that the best solution for creating a system that will receive real-time information is to use Big Data solutions. The proposed bicycle rental station system will send and process information in real time.

The software solution for processing the information of the proposed system in real time is written in Scala. The application of the proposed system makes it possible to create messages about the needs of stations in bicycles or free spaces, which will improve business. The developed system allows building reporting tables on the results of specific stations for a certain period of time.

To apply the proposed algorithm, three software modules have been developed for automated information processing using the Big Data solution. The developed system works with information that is already stored and with incoming information.

It is shown that the results are filtered according to time parameters and sorted according to the number of free places. The results of the system are displayed visually on the screen and stored in files on disk.

Keywords: *big data, real-time information processing, parallel programming, Spark, Scala, Kafka.*

Стаття надійшла до редакції 12.05.2021
Received 14.05.2021