

УДК 620.91: 330.567.4

Чорний К. П.¹, Горбенко В. І.²

¹ студ. гр. 8.1213, Запорізький національний університет

² проф., Запорізький національний університет

РОЗРОБКА БІБЛОТЕКИ ДЛЯ ОПТИМІЗАЦІЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ НА ПЛАТФОРМІ АРАСНЕ SPARK МОВОЮ JAVA

Сучасні задачі аналізу даних є доволі складними та громіздкими. Дані які потрібно проаналізувати можуть мати будь-який вигляд (починаючи від звичайних текстових форматів, до більш складних як медичні дані, чи просторові дані). Окрім складності вигляду даних, операції які проводяться над даними також можуть бути доволі складними, витрачати багато ресурсів та часу. Обидві проблеми можна вирішити через розподілення роботи. Якщо процес перетворення даних буде виконувати не один процесор, а одразу декілька, швидкість перетворення виросте в декілька разів. Для розподілення обробки даних існують так звані двигуни обробки даних. Найбільш популярний — це Apache Spark [1].

Apache Spark — це паралельний двигун обробки даних розроблений спеціально для обробки великих масивів даних. Двигун написаний на Scala, проте має багато API для багатьох мов програмування як: Python, R, Java, C#.

Apache Spark підтримує кластерну модель для паралельного обчислення. В кластері може бути багато вузлів які будуть виконувати роботу, проте обов'язково має бути вузол який буде виступати у ролі диригенту який буде розподіляти роботу поміж робочими вузлами, та отримувати результат роботи з вузлів.

Spark складається з декількох модулів: Spark SQL and DataFrames, Spark Streaming, Spark Mllib, Spark GraphX

Spark SQL and DataFrames [2] — це модуль двигуна для роботи зі структурованими даними. Використання цього модуля дозволяє

використовувати SQL мову для написання запитів для маніпуляції структурованими даними. Модуль підтримує уніфіковану систему доступу для даних. Це означає що провайдери даних можуть бути різними, як: Apache Hadoop, JSON, JDBC, проте засоби якими можна маніпулювати даними завжди працюватимуть однаково для всіх типів структурованих даних. В модулі присутній оптимізатор який аналізує SQL запити, та може за потреби змінити їх на більш оптимізовані щоб зменшити навантаження на кластер. Якщо інженер даних не хоче писати SQL код напряду, для цього існує DataFrame. DataFrame — це розподілена колекція як і RDD, проте має більше вбудованих методів для більш гнучкого аналізу даних. В сучасному використанні Apache Spark краще використовувати DataFrame аніж RDD.

Spark Streaming [3] — це модуль який надає функціонал обробки даних з певного джерела по потокам. Дані які надходять в потоки можуть бути з будь-якого джерела даних. Пайплайн обробки даних може бути таким самим складним як і в статичній обробці даних, при цьому не буде ніяких просадок в швидкості обробки даних. Модуль приймає дані в асинхронному порядку, збирає їх певну кількість щоб заповнити буфер, відправляє їх на обробку в іншому потоці, а сам повертається накопичувати дані для наступної обробки.

Spark Mllib [4] — це модуль який відповідає за машинне навчання та іншого роду схожі алгоритми. Ціль цього модуля — це полегшити створення практичних мережей машинного навчання та тренувати їх швидше в паралелі. Модуль надає стандартні алгоритми машинного навчання як: класифікація, кластеризація та регресія. Також присутні стандартні для цього роду методи очищення даних, фільтрації та перетворення даних. Деякі моделі яка знаходяться в цьому модулі підтримують збереження та завантаження з диску. Це зручно в ситуаціях коли розроблена модель була протестована на слабкому комп'ютері, та готова для навчання на більш потужному кластері.

Spark GraphX [5] — це модуль який додає функціонал графів в двигун. Граф є абстракцією над RDD, тому всі операції проводяться в паралелі. Оскільки вершини графа є доволі простими структурами даних, то перенесення даних з одного фреймворку до Apache Spark доволі проста задача.

Apache Spark — це доволі потужний та зручний інструмент обробки даних. В ньому присутня велика кількість вбудованих модулів обробки даних що робить його легким у використанні прямо з коробки.

Не дивлячись на те що модулі Apache Spark є доволі різноманітними, в них все рівно відсутні певні примітивні речі які мали б існувати. Окрім цього вже існуючі модулі написані на Scala. Це не є їх головним мінусом, проте їх використання з Java коду доволі заплутане та в багатьох випадках потребує знань Scala.

Мета цієї роботи - розробити класи які б допомагали інженерам даних швидше проводити маніпуляції над даними. Таким чином інженера більше часу зосередять саме на обробці даних а не створенні класів допоміжників для маніпуляції з даними.

Класи мають бути розділені по “package” для більшого розуміння того який клас за що відповідає. Наприклад може бути присутнім пакет “math” чи “strings” в якому будуть присутні класи які допоможуть інженеру даних в побудові пайплайну.

В пакеті математики може бути присутнім клас який відповідає за математичну матрицю та різного роду дії над нею. Підпакети пакету математики можуть відноситися до класів як допоміжні класи для основних функцій цього пакету.

Клас матриці відповідає за збереження матриці та відповідні дії над нею. Діями може бути додавання, віднімання або множення матриць. Також мають бути присутніми методи для створення матриці як з двовимірного чи одновимірного масиву. Для проведення маніпуляцій над матрицями використовується Apache Spark. Кожна з вищезазначених операцій проводиться паралельно на декількох машинах для того щоб пришвидшити час виконання операції. Для проведення самих операцій над матрицями, самі матриці мають бути розділені на частини для проведення операцій. Для додавання чи віднімання це просте додавання чи віднімання рядків матриць між собою. Для множення ця задача стає складнішою, оскільки там декілька разів проходять по рядкам та стовпцям матриць щоб отримати правильний результат.

Розробка бібліотеки для оптимізації обробки великих даних на платформі Apache Spark мовою Java є важливою задачею для підвищення продуктивності. Бібліотека знижує час на створення пайплайну обробки даних завдяки розробленим класам які присутні в бібліотеці. Вона забезпечує високу масштабованість та продуктивність на розподілених системах. Написані класи бібліотеки спрощують інтеграцію та використання у проєктах. Використання Apache Spark дозволяє адаптувати бібліотеку під конкретні завдання. Бібліотеку можна продовжувати розвивати додаючи нові класи які б допомогли інженерам даних швидше створювати пайплайни для обробки даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://en.wikipedia.org/wiki/Apache_Spark
2. <https://spark.apache.org/sql/>
3. <https://spark.apache.org/streaming/>
4. <https://spark.apache.org/mllib/>
5. <https://spark.apache.org/graphx/>