УДК 004.272.2:519.63
DOI: 10.31474/2415-7902-2020-1(4)-2(5)-27-36

О.А. Дмитрієва, Д.В. Нікулін

РОЗПОДІЛЕНА ОБРОБКА ВЕЛИКИХ ОБСЯГІВ ТРАНЗАКЦІЙНИХ ДАНИХ

Роботу присвячено питанням розподіленої обробки транзакцій при проведенні аналізу великих обсягів даних з метою пошуку асоціативних правил. На основі відомих алгоритмів глибинного аналізу даних для пошуку частих предметних наборів AIS та Apriori було визначено можливі варіанти паралелізації, які позбавлені необхідності ітераційного сканування бази даних та великого споживання пам'яті. Досліджено можливість перенесення обчислень на різні платформи, які підтримують паралельну обробку даних. В якості обчислювальних платформ було обрано MapReduce – потужну базу для обробки великих, розподілених наборів даних на кластері Hadoop, а також програмний інструмент для обробки надзвичайно великої кількості даних Apache Spark. Проведено порівняльний аналіз швидкодії розглянутих методів, отримано рекомендації щодо ефективного використання паралельних обчислювальних платформ, запропоновано модифікації алгоритмів пошуку асоціативних правил.

В якості основних завдань, реалізованих в роботі, слід визначити дослідження сучасних засобів розподіленої обробки структурованих і не структурованих даних, розгортання тестового кластера в хмарному сервісі, розробку скриптів для автоматизації розгортання кластера, проведення модифікацій розподілених алгоритмів з метою адаптації під необхідні фреймворки розподілених обчислень, отримання показників швидкодії обробки даних в послідовному і розподіленому режимах з застосуванням Hadoop MapReduce. та Apache Spark, проведення порівняльного аналізу результатів тестових вимірів швидкодії, отримання та обґрунтування залежності між кількістю оброблюваних даних, і часом, витраченим на обробку, оптимізацію розподілених алгоритмів пошуку асоціативних правил при обробці великих обсягів транзакційних даних, отримання показників швидкодії розподіленої обробки існуючими програмними засобами.

Ключові слова: розподілена обробка, транзакційні дані, асоціативні правила, обчислюваний кластер, Hadoop, MapReduce, Apache Spark

Вступ. Сучасне суспільство все частіше переповнюється цифровою інформацією, яка транслюється з супутників і передається по радіоканалах, кабелях, оптичних мережах. Великі обсяги даних (Data Science) можуть бути отримані як повідомлення з соціальних мереж, датчиків потоку трафіку, супутникових знімків, трансляції аудіо потоків, банківських транзакцій, музичних MP3-файлів, вмісту веб-сторінок, скан-копій урядових документів, GPS-трекінгу, телеметрії від автомобілів і т.і. За прогнозами Cisco [1] очікується, що до 2021 року річний обсяг даних, які передаються через мобільні і фіксовані мережі, досягне 3,3 зеттабайт і буде збільшуватися більш ніж у 10 разів протягом наступного десятиліття [2], при тому, що загальний обсяг цифрової інформації, створений людством до 2006 р., склав всього 0,16 зеттабайт.

Успіхи сучасних комерційних компаній практично в кожній галузі визначаються ефективним використанням систем аналізу великих об’ємів даних для отримання більш точної інформації про своїх клієнтів, процеси, персонал, продукти та пропозиції, наприклад, за допомогою Google AdSense, який збирає дані від користувачів Інтернету і персоналізує рекламу. Такі системи працюють ефективно завдяки аналізу великої кількості даних, на основі яких можна будувати прогнози. Більш того, системи спроектовані таким чином, щоб з часом вдосконалюватися за рахунок відстеження корисних сигналів по мірі надходження нових даних. Так, Amazon може порекомендувати ідеальну книгу, Google - оцінити релевантність сайту, Facebook знає, що нам подобається, а LinkedIn передбачає, які знайомства можуть бути корисними в професійних галузях. Аналогічні системи можуть бути застосовано для діагностики захворювань, рекомендацій щодо лікування, навчання і т.і. [3].

Для обробки великих обсягів інформації не завжди можна застосувати традиційні системи роботи з даними [4], тому з’являються інші технології обробки, які позбавлені

вимог щодо жорсткої ієрархії і однорідності даних. З'явилися нові технології обробки, наприклад, модель MapReduce компанії Google і її аналог з відкритим вихідним кодом – Hadoop від компанії Yahoo. Такі технології надали можливість управляти набагато більшою кількістю даних, ніж раніше, а сучасне обладнання, хмарні сервіси Amazon Web Services (AWS), Cloudera, Microsoft Azure [5], та програмне забезпечення з відкритим кодом дозволили проводити обробку без значних фінансових ресурсів.

Для застосування підходів щодо збереження великих обсягів даних в роботі було проаналізовано велику кількість варіантів, що мають широке розповсюдження. На основі проведеного аналізу перевагу було надано таким системам, як Hadoop, MapReduce та Apache Spark, для яких, здебільшого, використовується своя розподілена файлова система. Підґрунтям такого вибору стало урахування особливостей предметної області, яку характеризують великі обсяги неструктурованих транзакційних даних фармацевтичного сектору з завданням пошуку асоціативних правил. Пошук асоціативних правил в роботі базувався на класичних алгоритмах Apriori, Apriori AIS та AprioriTID [6], що дозволяють обробляти величезну кількість даних про продажі, які містять, як правило, дати транзакцій та переліки товарів, придбаних у процесі транзакції.

Виходячи з вищевказаного, можна сформулювати мету роботи, яка полягає у розробці та оптимізації розподілених алгоритмів пошуку асоціативних правил при обробці великих обсягів транзакційних даних та отриманні показників швидкодії розподіленої обробки існуючими програмними засобами.

1. Розподілена реалізація алгоритмів пошуку асоціативних правил

Для проведення експериментів та заміру швидкодії роботи паралельних обчислень в роботі було обрано алгоритми, які можуть працювати як у послідовному, так і у паралельному режимах. Головними чинниками при визначенні прийнятних алгоритмів для проведення досліджень були складність реалізації та можливість перенесення на різні платформи з паралельних обчислень, однією з яких виступала платформа MapReduce – потужна база для обробки великих, розподілених наборів даних на кластері Hadoop з файловою системою HDFS [7-8]. Взаємодію Hadoop з файловою системою HDFS (рис. 1) в роботі було організовано як виконання наступних етапів: завантаження даних у HDFS, операції MapReduce, отримання результатів від HDFS.

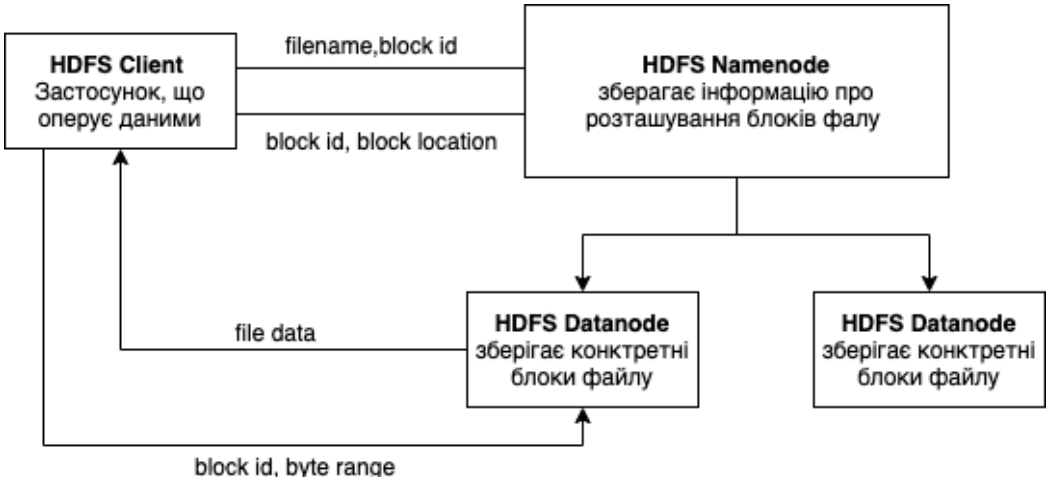


Рисунок 1 – Організація взаємодії з файловою системою HDFS

Також в роботі в якості швидкої, універсальної, розподіленої обчислювальної платформи використовувався Apache Spark [9-10] – програмний інструмент для обробки надзвичайно великої кількості даних. В якості алгоритмів глибинного аналізу даних для

пошуку частих предметних наборів було визначено AIS та Aprori [11-12], які мають декілька можливих варіантів паралелізації, а також позбавлені недоліків послідовних алгоритмів, а саме, високої вартості вводу, виводу через ітераційне сканування бази даних та велике споживання пам'яті. Для підвищення продуктивності зазначених алгоритмів запропоновано декілька паралельних реалізацій, а також популярні послідовні підходи, засновані на хешуванні, зменшенні транзакцій та використанні динамічного визначення наборів елементів [12]. Найпоширенішим засобом паралелізації алгоритмів Apriori та AIS є паралелізація за розподілом підрахунків.

Для реалізації завдань map та reduce за допомогою мови Python було використано утиліту Hadoop Streaming [8], яка дозволяє створювати та запускати завдання map, reduce та combiner із будь-якими виконуваними файлами, що можуть читати зі стандартного потоку вводу stdin. Принцип роботи Hadoop Streeming зображено на рис. 2. На відміну від map-завдання першої ітерації, яке є досить тривіальним, reduce-завдання є більш складнішим. Від кількості завдань reduce, що встановлюються індикатором mapreduce.job.reduces=1, залежить кількість результуючих файлів, саме через це кількість reduce-завдань буде обмеженою. Це також обумовлено тим, що результат роботи reduce після першої ітерації необхідно використати на наступній ітерації. Звісно, можна об’єднати результати роботи кількох map-завдань, але це призведе до підвищення витраченого часу на обробку та вплине на репрезентативність результатів під час порівняння з послідовною та Apache Spark реалізаціями.

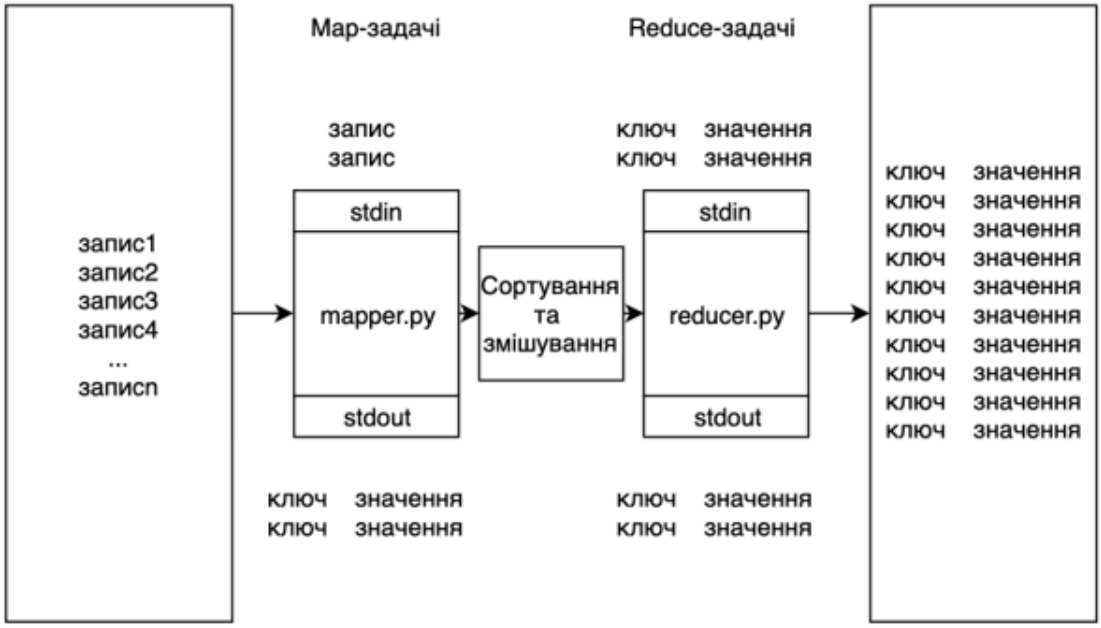
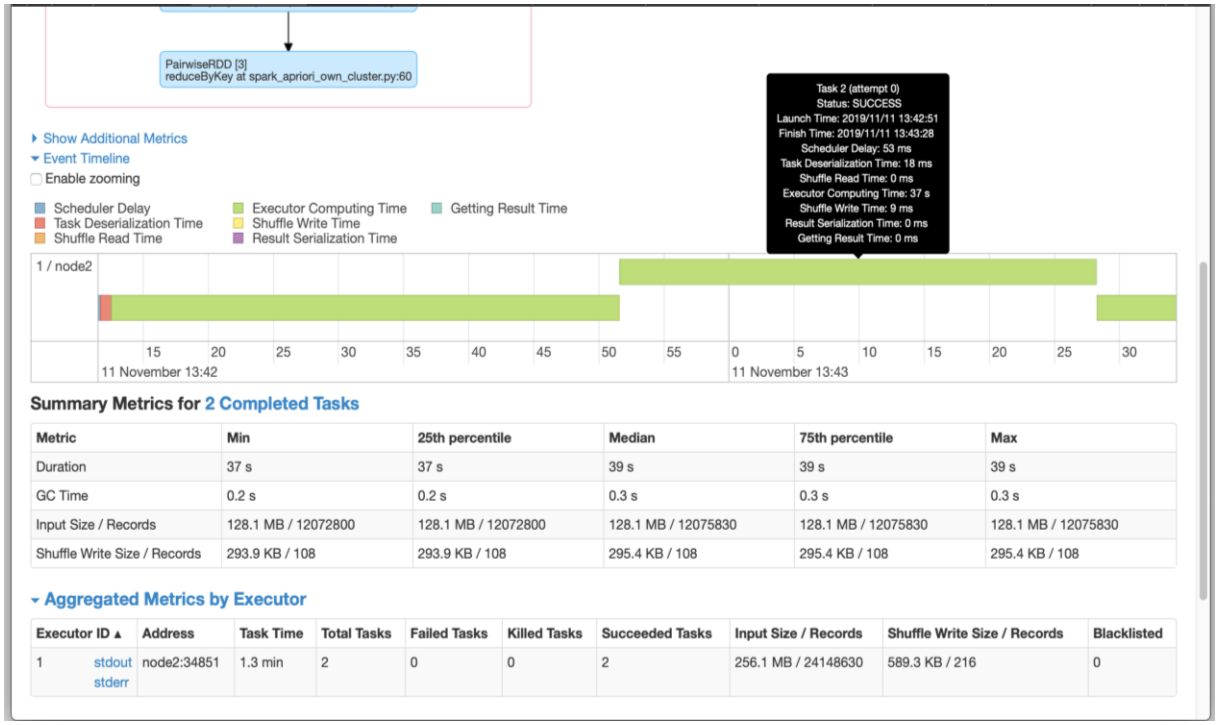


Рисунок 2 – Організація роботи Hadoop Streaming

Важливою властивістю роботи MapReduce є те, що після етапу map, на етапі сортування та змішування, дані сортуються за ключем, це спрощує реалізацію reduce-завдання. Реалізована перша ітерація алгоритму AIS для reduce отримує на вхід відсортовані результати роботи усіх map-завдань, підсумовує результати за ключем і відсікає всі результати, в яких кількість входжень елементів менше заданого показника n, для визначення якого використовується співвідношення

$$n = |T| * min\_supp, \quad (1)$$

де $|T|$ – кількість транзакцій,
min_supp – мінімальна підтримка.



Metric	Min	25th percentile	Median	75th percentile	Max
Duration	37 s	37 s	39 s	39 s	39 s
GC Time	0.2 s	0.2 s	0.3 s	0.3 s	0.3 s
Input Size / Records	128.1 MB / 12072800	128.1 MB / 12072800	128.1 MB / 12075830	128.1 MB / 12075830	128.1 MB / 12075830
Shuffle Write Size / Records	293.9 KB / 108	293.9 KB / 108	295.4 KB / 108	295.4 KB / 108	295.4 KB / 108

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Input Size / Records	Shuffle Write Size / Records	Blacklisted
1 stdout stderr	node2:34851	1.3 min	2	0	0	2	256.1 MB / 24148630	589.3 KB / 216	0

Рисунок 3 – Розподіл виконання стадій роботи Apache Spark

Перевага реалізованого коду reduce-завдання полягає в тому, що воно підходить для будь-якої ітерації алгоритму, тому немає жодної необхідності витрачати час на розробку коду reduce-завдань для наступних ітерацій. На відміну від цього map-завдання для наступних ітерацій потребують завантаження додаткових переліків частих предметних наборів з попередніх ітерацій. З файлу зчитуються усі часті предметні набори попередньої ітерації, які потім використовуються під час сканування усіх транзакцій. Якщо предметні набори поточної транзакції є частими у порівнянні з (1), то відбувається генерація кандидатів та їх запис до актуального переліку. В якості ще одного розподіленого варіанту організації процедури пошуку асоціативних правил було використано платформу Apache Spark. Дані у середовищі Apache Spark обробляються у значно простішому для користувача вигляді, бо, на відміну від реалізації Hadoop MapReduce, немає необхідності перезапускати задачу для кожної ітерації. Між усіма ітераціями алгоритму у реалізації на Apache Spark немає значущою різниці у часі, що є перевагою в порівняні з реалізацію Hadoop MapReduce. Графічний інтерфейс Apache Spark (рис. 3) наочно ілюструє розподіл і опрацювання даних на кластері.

2. Порівняльне тестування розподілених алгоритмів на Hadoop MapReduce

Оцінювання швидкодії розроблених розподілених алгоритмів здійснювалося з використанням платформа Hadoop MapReduce та Apache Spark. Дані для тестування є набором транзакцій купівлі товарів фармацевтичної мережі. Для створення умов для рівномірного тестування швидкодії, було згенеровано основні набори даних, інформацію про які наведено на рис. 4. Треба зауважити, що великі файли, обсяг яких перевищував 128 МБ, розбивалися на блоки та розподілялися по обчислювальним вузлам.

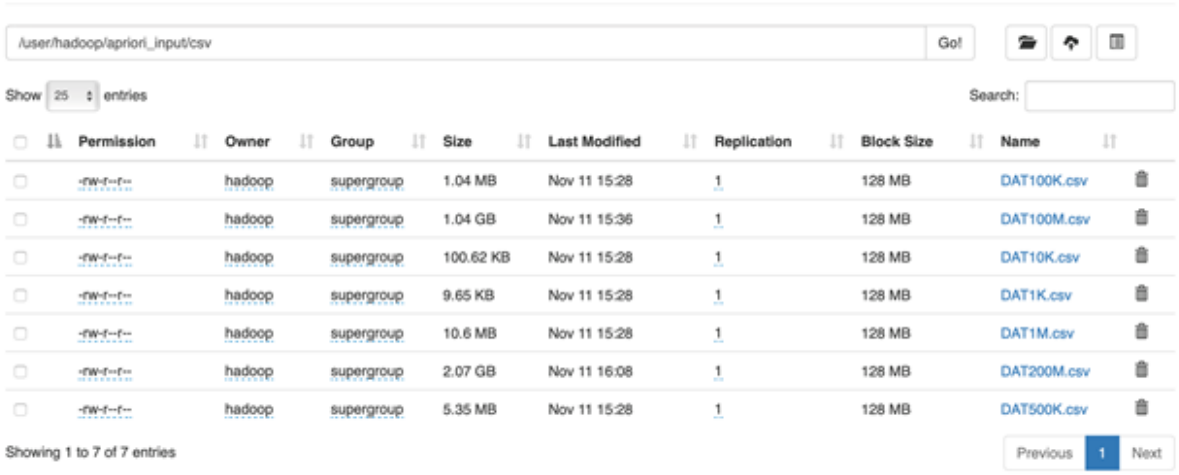


Рисунок 4 – Розбиття завантажених тестових даних

Саме завдяки тому, що дані були розбиті на рівні частини та знаходилися на декількох фізичних машинах одночасно, відбувалася дійсна паралелізація при використанні MapReduce або Apache Spark.

Після проведення низки експериментів було отримано достатню кількість даних для оцінювання швидкодії розроблених алгоритмів та проведення порівняного аналізу. Треба звернути увагу, що усі проведені експерименти включали обробку пакетів до 2 ГБ даних, що є вагомим навантаженням на лінійний алгоритм, але не є реальним навантаженням системи великих даних, а саме, розподіленої обробки. Але для академічного дослідження розподіленої обробки даних результати отриманих досліджень є ґрунтовними аби робити висновки. На отриманих даних (рис. 5-8) явно видно, що з ростом кількості транзакцій у вхідному файлі також зростає витрачений час на обробку. З кожною ітерацією час, витрачений на сканування усього набору даних, зменшується, це напряму залежить від встановленої підтримки. Також було помічено, що на невеликих об’ємах даних (200К, 300К), подвоєння кількості вхідних даних не подвоює час обробки цих даних (рис. 5). Час, витрачений на обробку даних великих об’ємів (100М, 200М), збільшується пропорційно до розміру файлу (рис. 6).

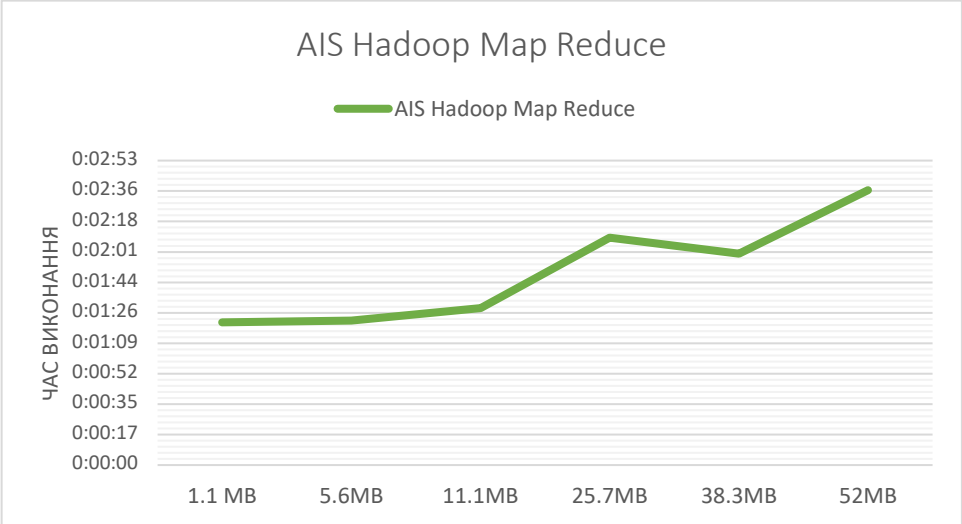


Рисунок 5 – Діаграма витраченого часу для алгоритму AIS за допомогою Hadoop MapReduce на невеликих файлах

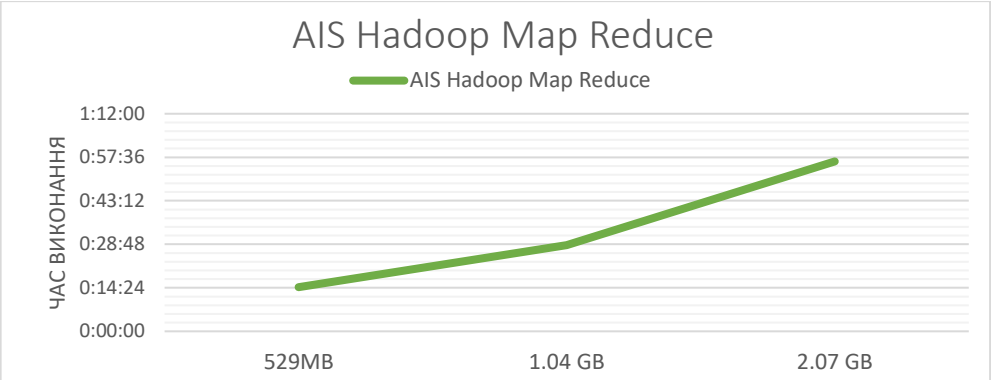


Рисунок 6 – Діаграма витраченого часу для алгоритму AIS при обробці за допомогою Hadoop MapReduce на великих файлах

3. Порівняльне тестування розподілених алгоритмів на Apache Spark

Після проведення тестів для Hadoop MapReduce, тестовий кластер було перезапущено, та сконфігуровано наново для запуску задач Apache Spark за допомогою YARN. Розроблений код було скопійовано на віддалену машину node-master та запущено в обробку. На рис. 7-8 можна побачити діаграми швидкодії обробки даних за допомогою Apache Spark. На них видно, що час виконання зростає лінійно в залежності від кількісті даних, що обробляються.

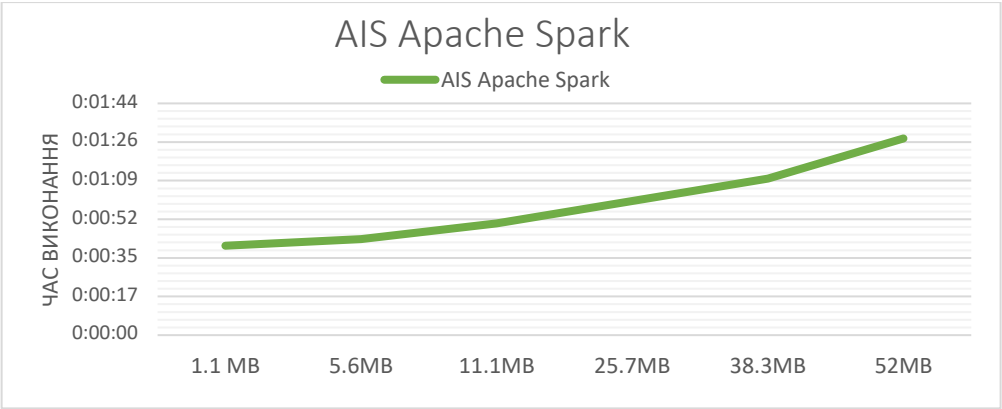


Рисунок 7 – Діаграма витраченого часу для алгоритму AIS при обробці за допомогою Apache Spark на невеликих файлах

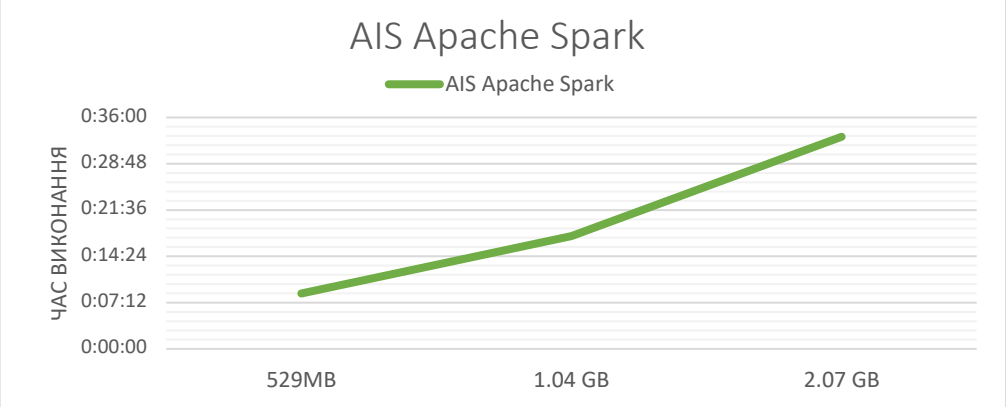


Рисунок 8 – Діаграма витраченого часу для алгоритму AIS при обробці за допомогою Apache Spark на великих файлах

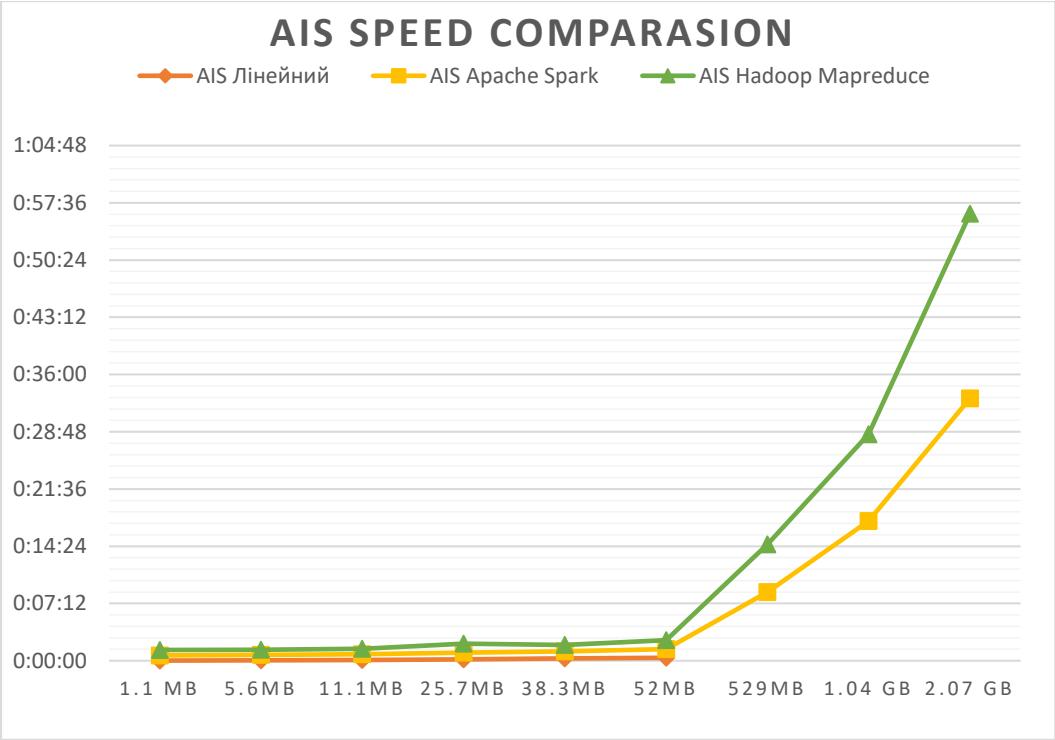


Рисунок 9 – Порівняння швидкодії усіх реалізацій AIS

На рис. 9 зображено зведений порівняльний графік швидкодії. Продуктивність роботи системи великих даних за допомогою Apache Spark майже у два рази більша за аналогічний результат для Hadoop MapReduce для найбільших файлів. Підтверджено, що при ітеративній обробці даних Apache Spark має вагому перевагу у часі обробки. Також на графіку можна побачити, що лінійний (не паралельний) алгоритм AIS перестає нормально функціонувати при значущих обсягах даних. Це підтверджує припущення про те, що лінійні алгоритми не завжди можуть обробити великий об'єм даних. Також використання HDFS дає можливість обійти ліміти однієї машини на зберігання даних, а Apache Spark та Hadoop допомагають обробляти їх у розподілених режимах.

Висновки. Представлена робота була мотивована розумінням того, що розподілена обробка великих даних не завжди виконується найбільш відповідними програмними рішеннями. Залежно від природи даних та типу поставленого завдання, обробка даних може виконуватись великою кількістю різних програмних засобів. Наприклад, зберігання та обробка звичайних структурованих текстових даних може здійснюватися без використання великих спеціалізованих фреймворків розподілених обчислень, буде достатнім застосування рішення РСУБД або NoSQL. Але коли потрібно обробити неструктуровані дані або є потреба в більш складних сценаріях, то необхідно використовувати спеціальні рішення для розподіленої обробки таких даних.

В якості основних завдань, які було реалізовано в роботі, слід визначити дослідження існуючих засобів розподіленої обробки структурованих і неструктурованих даних, розгортання тестового кластеру у хмарному сервісі, розробку скриптів для автоматизації розгортання тестового кластеру, проведення модифікацій розподілених алгоритмів з метою адаптації під необхідні фреймворки розподілених обчислень, отримання показників швидкодії обробки даних у послідовному та розподіленому режимах на кластері, проведення порівняльного аналізу отриманих результатів тестових замірів швидкодії, отримання і обґрунтування залежності між кількістю даних, що обробляються, та часом, витраченим на обробку, оптимізацію розподілених алгоритмів

пошуку асоціативних правил у великих обсягах транзакційних даних, отримання показників швидкодії розподіленої обробки існуючими програмними засобами.

Доведено, що обробка даних за допомогою Apache Spark при невеликих об'ємах оперативної пам'яті робочих вузлів не є дуже продуктивною. На тестовому стенді було отримано повідомлення про нестачу пам'яті через те, що процес обробник (executor) не може бути розміщеним у пам'яті машини. Причиною було те, що під час запуску завдання у режимі cluster, наглядовий процес (driver) запускається на робочому вузлі. Це призводить до того, що перша машина кластеру відповідає за роботу HDFS та YARN, друга - за роботу driver Apache Spark, та лише одна машина залишається для обробки завдань. Та, незважаючи на обробку даних лише на одному вузлу, завдяки оптимізації процесу функціонування ітеративних алгоритмів та утримання блоків файлу з HDFS у оперативній пам'яті, дані обробляються все одно швидше, ніж у MapReduce.

Визначено, що на теперішній час головними рішеннями з розподіленої обробки даних виступають програмні комплекси, тісно зав'язані на екосистемі Hadoop. Функціонально парадигма програмування MapReduce найкраще реалізується за допомогою засобів Hadoop, а саме Hadoop MapReduce. Рішення Apache Spark менш зав'язане на Hadoop, але використовуючи HDFS та велику кількість оперативної пам'яті, можна досягти у рази більшої швидкодії.

Під час роботи було отримано необхідні практичні навички конфігурацій та розгортання Hadoop кластера у середовищі Amazon AWS. Написання скриптів розгортання тестового кластеру дало змогу провести тестування на реальному обчислювальному обладнанні, що ізольовано від зовнішніх факторів впливу на репрезентативність результатів. Розроблені скрипти розгортання можна використовувати в подальших дослідженнях.

Було проведено близько 20 експериментів на даних різних розмірів, що дало змогу оцінити продуктивність кожного з програмних рішень Після виконання усіх поставлених завдань та огляду сучасного стану проблеми, що розроблюється, базуючись на отриманих в ході дослідження результатах експериментів, можна дійти висновку, що в залежності від поставленої математичної задачі на обробку великих обсягів даних, потрібно обирати найкраще рішення саме для конкретної ситуації. Якщо задача потребує ітеративної обробки однієї множини даних, то доцільніше використовувати Apache Spark. Час, витрачений на обробку невеликих файлів об'ємом до 1 ГБ, не є значущим. Різниця помітніша при збільшені розміру оброблюваних даних. Час обробки найбільшого тестового файлу засобами Apache Spark майже у два рази менший, ніж при застосуванні того ж самого алгоритму для Hadoop MapReduce. Ітеративні алгоритми дуже важко розробляти та тестувати для Hadoop MapReduce, бо кожна ітерація потребує окремого запуску завдань map та reduce. Лінійна реалізація алгоритму AIS працює лише на невеликих об'ємах даних до 100 МБ. В роботі не проводилось тестування неітеративного алгоритму, але можна припустити, що рішення MapReduce теоретично може мати більшу високу швидкодію, коли в системі обмежений обсяг оперативної пам'яті через відсутність використання еластичного розподіленого набору даних.

Список літератури

1. Complete updated Cisco VNI forecast for the period 2016-2021. [Electronic resource]. – Access mode: https://www.cisco.com/c/en/us/solutions/service-provider/vni-network-traffic-forecast/vni-forecast-info.html.

2. World faces data storage crunch ahead [Electronic resource]. – Access mode: https://www.straitstimes.com/singapore/world-faces-data-storage-crunch-ahead.

3. Майер-Шенбергер В. Большие данные/ В. Майер-Шенбергер, К. Кукьер. – М.: «Манн, Иванов и Фербер». – 2014. – 234 с.

4. Ashlesha S. Overview on Performance Testing Approach in Big Data /S. Ashlesha, R. M. Tugnayat // International Journal of Advanced Research in Computer Science. – 2014. – Vol. 5, № 8. – P. 165-169.
5. Cloud Computing Characteristics and Services/ A Brief Review//International Journal of Computer Sciences and Engineering. – 2019. – Vol. 7, № 2. – P. 421-426.
6. Bouraoui M. Efficient Mining of Association Rules based on Clustering from Distributed Data/ M. Bouraoui, A. G. Touzi// International Journal of Advanced Computer Science and Applications. – 2019. – Vol. 10, № 4. – P. 401-409.
7. Abualkishik A. Hadoop and Big Data challenges /A. Z. Abualkishik// Journal of Theoretical and Applied Information Technology. – 2019. – Vol. 97, № 12. – P. 3488-3500.
8. Cohen J. Towards a More Secure Apache Hadoop HDFS Infrastructure/ J. Cohen, S. Acharya// International Conference on Network and System Security. – 2013. – P. 731-741.
9. Zecevic P. Spark in Action / P. Zecevic, M. Bonaci. – Shelter Island: Manning Publications. – 2017. – 42 p.
10. Карау Х. Эффективный Spark. Масштабирование и оптимизация/ Х. Карау, Р. Уоррен. – СПб.: Питер, 2018. – 352 с
11. Дмитриева О.А. Алгоритмическая поддержка параллельных методов поиска ассоциативных правил / О.А. Дмитриева, О.Л. Половинка // Наукові праці ДонНТУ. Серія: «Обчислювальна техніка та автоматизація». – 2018. – № 1(31). – С. 83-90.
12. Дмитрієва О.А. Дослідження швидкодії розподіленої обробки великих обсягів даних/ О.А. Дмитрієва, Д.В. Нікулін // Матеріали XIX міжнародної науково-технічної конференції «Проблеми інформатики та моделювання». – Харків. – 2019. – С.34.

References

1. Complete updated Cisco VNI forecast for the period 2016-2021. [Electronic resource]. – Access mode: https://www.cisco.com/c/en/us/solutions/service-provider/vni-network-traffic-forecast/vni-forecast-info.html.
2. World faces data storage crunch ahead [Electronic resource]. – Access mode: https://www.straitstimes.com/singapore/world-faces-data-storage-crunch-ahead.
3. Majer-Shenberger, V. and Kuk'er, K., (2014), “Big data”, [Bol'shie dannye], Moskow, “Mann, Ivanov i Ferber”, 234 p. (in Russian).
4. Ashlesha, S. and Tugnayat, R. M., (2014), “Overview on Performance Testing Approach in Big Data”, International Journal of Advanced Research in Computer Science, P. 165-169.
5. Brief Review, (2019), “Cloud Computing Characteristics and Services”, International Journal of Computer Sciences and Engineering, P. 421-426.
6. Bouraoui, M. and Touzi, A. G., (2019), “Efficient Mining of Association Rules based on Clustering from Distributed Data”, International Journal of Advanced Computer Science and Applications, P. 401-409.
7. Abualkishik, A., (2019), “Hadoop and Big Data challenges”, Journal of Theoretical and Applied Information Technology, P. 3488-3500.
8. Cohen, J. and Acharya, S., (2013), “Towards a More Secure Apache Hadoop HDFS Infrastructure”, International Conference on Network and System Security, P. 731-741.
9. Zecevic, P. and Bonaci, M., (2017), “Spark in Action”, Shelter Island: Manning Publications, 42 p.
10. Karau, H. and Uorren, R., (2018), “Effective Spark. Scaling and optimization“, [Jeffektivny Spark. Masshtabirovanie i optimizacija], SPb.: Piter, 352 p.
11. Dmitrieva, O. and Polovinka, O., (2018), “Algorithmic support for parallel search methods for associative rules”, [Algoritmicheskaja podderzhka parallel'nyh metodov poiska associativnyh pravil], Nauchnye trudy DonNTU. Serija “Vychislitel'naja tehnika i avtomatizacija”, P. 83-90. (in Russian).
12. Dmitrieva, O. and Nikulin, D., (2019), “Research of the speed of distributed processing of large amounts of data”, [Issledovanie bystrodejstvija raspredelennoj obrabotki bol'shih ob"emov dannyh], Materialy XIX mezhdunarodnoj nauchno-tehnicheskoj konferencii «Problemy informatiki i modelirovanija» Kharkov, P.34. (in Ukrainian).

Надійшла до редакції 25.08.2020

O. Dmytriieva, D. Nikulin

DISTRIBUTED PROCESSING OF LARGE VOLUMES OF TRANSACTIONAL DATA

Purpose. *The article is devoted to the issues of distributed transaction processing in the analysis of large volumes of data in order to search for associative rules.*

Methodology. *Based on the well-known data mining algorithms for searching frequent subject sets AIS and Apriori, the possible parallelization options needless iterative database scanning and high memory consumption, have been identified. The possibility of transferring computations to various platforms that support parallel data processing has been investigated. MapReduce has been chosen as a computing platform having a powerful base for processing large, atomized data sets on a Hadoop cluster, and being a software tool for processing a very large amount of Apache Spark data.*

Results. *A comparative analysis of the speed of the considered methods has been carried out. The recommendations on the effective use of parallel computing platforms have been received.*

Practical novelty. *Some modifications of the algorithms for searching associative rules have been proposed.*

Scientific significance. *As the main tasks implemented in the work, it is necessary to determine the research of existing means of distributed processing of structured and unstructured data, the deployment of a test cluster in a cloud service, the development of scripts to automate the deployment of a test cluster, the modification of distributed algorithms in order to adapt to the necessary frameworks of distributed computing, obtaining data processing performance indicators in sequential and distributed modes per cluster, a comparative analysis of the results of test measurements of performance, reception and study the relationship between the amount of data to be processed, and time spent on processing, optimization of distributed algorithms for searching association rules in the processing of large volumes of transactional data, and obtaining a distributed processing performance parameters of existing software.*

Keywords: distributed processing, transactional data, associative rules, countable cluster, Hadoop, MapReduce, Apache Spark*.*

Відомості про авторів

Дмитрієва О.А., докт. техн. наук, проф., зав. кафедри прикладної математики та інформатики ДВНЗ «Донецький національний технічний університет», e-mail: olga.dmytriyeva@donntu.edu.ua

Нікулін Д.В., магістрант ДВНЗ «Донецький національний технічний університет», e-mail: danil.nikulin@gmail.com

Dmytriieva O., Doctor of Technical Sciences, Professor, Head of the Department of Applied Mathematics and Informatics, SHEE Donetsk National Technical University, e-mail: olga.dmytriyeva@donntu.edu.ua

Nikulin D., Master's Degree Student, SHEE Donetsk National Technical University, e-mail: danil.nikulin@gmail.com