МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ /В. І. Шинкаренко/

«_____» _____ 20____р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Дослідження технологій BigData для структуризації інформації з соцмереж і визначення трендів**

Theme **Research of modern BigData technologies and tools for social networks data analysis purpose.**

Керівник дипломної роботи доц. _____ В. О. Андрющенко

Нормоконтролер доц. _____ О. С. Куроп'ятник

Студент групи П31921 _____ Г. А. Жемела

Student Zhemela Heorhii

Дніпро – 2020

Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Факультет Комп'ютерних технологійі систем кафедра Комп'ютерні інформаційні технології

Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ проф. Шинкаренко В.І.
(підпис)
«15» грудня 2020 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС Магістр
(освітньо-кваліфікаційнийрівень)

студента групи 961-М Жемела Георгій Андрійович
(номер групи) (ПІБ)

1 Тема дипломної роботи: Дослідження технологій BigData для структуризації інформації з соцмереж і визначення трендів
затверджена наказом по університету від « » _____ р. № ___.

2 Термін подання студентом закінченої роботи 21 грудня 2020 р.

3 Вихідні дані до дипломної роботи _____

4 Зміст пояснювальної записки (перелік питань до розробки) Призначення, постановка задачі та огляд аналогів, методика проведення дослідження, проектування і розробка інструментального програмного забезпечення, дослідження сервісів технології BigData , охорона праці та безпека в надзвичайних ситуаціях, висновки.

5 Перелік демонстраційного матеріалу Постановка задачі, опис аналогів, методи вирішення, механізм реалізації, аналіз результатів, висновки

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	доц. Гненний М.В.	03.11.2020	14.12.2020
Охорона праці та безпека в надзвичайних ситуаціях	ст. викладач Музикін М.І.	14.12.20	17.12.20

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва розділів дипломного проекту	Термін виконання розділів проекту (роботи)	Примітка
1	Вступ	1.09.2020 – 10.09.2020	
2	Огляд літератури	10.09.2020 – 15.09.2020	
3	Постановка задачі, технічне завдання	15.09.2020 – 30.09.2020	
4	Створення тестової програми	01.10.2020 – 15.10.2020	
5	Перші тестування	15.10.2020 – 22.10.2020	30%
6	Аналіз результатів	30.10.2020 – 11.11.2020	
7	Розрахунок економічних показників	11.11.2020 – 14.11.2020	
8	Охорона праці	14.11.2020 – 18.11.2020	60%
9	Оформлення пояснювальної записки	18.11.2020 – 01.12.2020	
10	Демонстраційні матеріали	5.11.2020 – 16.12.2020	100%

Дата видачі завдання «10» жовтня 2019 р.

Керівник дипломного проекту _____ (підпис) Андрющенко В.О. (ПІБ)

Завдання прийняв до виконання _____ (підпис) Жемела Г.А. (ПІБ)

РЕФЕРАТ

Об‘єктом дослідження є сучасні засоби та підходи обробки BigData для аналізу текстових даних з соцмереж.

Предметом дослідження є порівняння ефективності інструментів потокової та пакетної обробки даних BigData для задач обробки невеликих повідомлень.

Метою дослідження в контексті даної роботи є визначення найбільш ефективних підходів обробки BigData шляхом порівняння найбільш використовуваних програмних засобів.

Методи дослідження: виміри продуктивності засобів для вирішення поставленої задачі шляхом використання типових алгоритмів NLP.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та 4 додатків.

Вступ – описує суть, мету та актуальність роботи (3 сторінки).

Перший розділ – розкриття проблеми, огляд можливих сфер використання (16 сторінок).

Другий розділ – аналіз існуючих підходів та програмних засобів для вирішення поставленої задачі(16 сторінок).

Третій розділ – побудова алгоритму для вирішення задачі (9 сторінок).

Четвертому розділ – дослідження ефективності використання технлогії bigdata для структуризації інформації із соціальних мереж та пошук трендів (13 сторінок).

П’ятий розділ – позгляд питань охорони та безпеки праці в надзвичайних ситуаціях (8 сторінок).

Додатки – технічне завдання, робочий проект, 1 теза, 1 стаття.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP – natural language processing (укр. обробка природних мов)

ML – machine learning (укр. машинне навчання)

AI – artificial intelligence (укр. штучний інтелект)

ВСТУП

Актуальність дослідження. У XXI столітті найціннішим ресурсом стала інформація, використання якої дозволяє досягти нових висот у всіх областях людської діяльності. Інформації стало настільки багато, що зберігати і обробляти її традиційними способами стало дуже складно, до того ж дані, оброблені традиційно, з'являються, як правило, із запізненням, отже будуть вже не актуальні. Крім того, що різні організації створюють величезний обсяги даних, велика частина яких представлена в різних форматах (текстові документи, відеофайли, машинні коди і т.д.), локалізованих в різноманітних сховищах, ці дані дуже часто оновлюються.

Термін «великі дані» (Big Data) існує вже майже двадцять років, ставши за цей час всесвітньо обговорюваних. За історію свого недовгого існування він встиг здобути широку популярність – з одного боку ці технології викликають деякий скепсис, з іншого боку велика кількість компаній вже впровадили Big Data в свою діяльність, що дозволило оптимізувати роботу з даними.

Ріст обсягів інформації супроводжується появою апаратних і програмних засобів, здатних оперативно обробляти більші обсяги інформації, а також значним зниженням вартості збору, обробки, зберігання й передачі единимы інформації.

У результаті з'єднання цих двох процесів – росту потреби бізнесу в обробці й зберіганні більших обсягів даних і появи технічних засобів, здатних оперативно обробляти такі дані з мінімальними витратами з'явилося одне з найцікавіших і перспективних напрямків розвитку послуг назва, що одержала, Big Data.

Незважаючи на те, що досвіду практичного застосування Big Data у сфері сервісу, у маркетингу поки накопичене не багато, інтерес до проектів у цій області постійно росте. Регулярно з'являються повідомлення про успішне застосування технологій Big Data інноваційними компаніями для розв'язку різних завдань підвищення конкурентоспроможності, створення нових сервісів, удосконалювання управління взаємодії із клієнтами.

З розповсюдженням розумних пристроїв та послуг, розширених соціальними мережами, воно завойовує значення і приваблює людей у великій кількості. Взаємодія

між людьми через соціальні мережі є потенційним джерелом неструктурованих, детальніших та масштабніших цифрових даних. Одним з найважливіших ресурсів для отримання інформації є саме вони. Соцмережі допомагають виконувати задачі збору статистики поведінки клієнтів, їх вподобань, тощо. Також дані соціальних мереж широко використорвуються науковцями у свїх дослідженнях.

Розвиток комп'ютерних технологій і впровадження їх у повсякденне життя відкрили не тільки безліч можливостей перед людством, але й поставили ряд проблем, для яких потрібно знайти розв'язок. Постійно зростаючі обсяги даних, які щодня генеруються, вимагають обробки. При цьому обробка повинна бути не тільки точною і якісною, що враховує модель даних і їх структуру, але й швидкою.

Враховуючи все вищесказане і була обрана тема магістерської роботи:

"Використання технології BigData для структуризації інформації із соціальних мереж та пошук трендів".

Об’єкт дослідження: сучасні засоби та підходи обробки BigData для аналізу текстових даних з соцмереж.

Предмет: порівняння ефективності інструментів потокової та пакетної обробки даних BigData для задач обробки невеликих повідомлень.

Мета роботи: визначити найбільш ефективний підход обробки BigData шляхом порівняння найбільш використовуваних програмних засобів.

Відповідно до мети були визначені наступні **завдання**:

1) Провести аналіз проблеми використання технологій BigData, зокрема для обробки даних з соцмереж;
2) Провести аналіз основних підходів та пррограмних засобів для обробки великої кількості текстових повідомлень засобами BigData;
3) Визначити алгоритми обробки повідомлень для вирішення задач пошуку трендів у повідомленнях;
4) Провести дослідження ефективності потокових та пакетних засобів BigData для структуризації інформації із соціальних мереж та пошуку трендів;
5) Охарактеризувати питання безпеки праці під час розробки ПЗ та виконанні досліджень

Для розв’язання поставлених завдань були використані такі **методи дослідження**:

- теоретико-критичний аналіз літератури з теми дослідження;
- зіставлення, узагальнення і синтезування здобутої інформації;
- аналіз результатів експериментів, тощо;

Робота може бути використана студентами ВНЗ для підготовки до семінарських занять, також може бути використана викладачами для проведення лекції, практик тощо.

1. АНАЛІЗ ПРОБЛЕМИ ВИКОРИСТАННЯ ТЕХНОЛОГІЙ BIGDATA

1.1. Призначення та сфера застосування

Вважається, що термін "Big Data" вперше використовував головний науковий співробітник компанії "Silicon Graphics Inc." Джон Мэши на конференції USENIX Association в 1998 р., але інтерес до даного поняття різко підвищився тільки після публікацій у журналі "Nature" у вересні 2008 р., де обговорювалися проблеми, викликані ростом обсягів даних, одержуваних у процесі проведення сучасних наукових експериментів. Однак змістовні джерела Big Data можна простежити ще в 1940-і роки, коли відзначалося, що фонд університетських бібліотек ріс із неймовірною швидкістю, подвоюючись кожні шістнадцять років.

За минулі із цього моменту роки даний термін став використовуватися з такою неймовірною інтенсивністю, що, ще, не будучи досить понят, став викликати негативну емоційну реакцію в деяких фахівців. Як відзначає компанія "Gartner Inc.", вивчаючи популярність різних технологічних термінів, спочатку відбувається досить різкий сплеск шуму навколо нової технології, потім настільки ж різке падіння інтересу й розчарування. І тільки потім можливо починається реальне масове впровадження. Отож зараз термін "Big Data" перебуває на піку популярності. А це значить, що через 5-10 років ці розробки, можливо, будуть масово поширюватися.

Разом з тим, дотепер немає єдиного поняття терміна "Big Data", який до того ж часто називають новим підходом в одержанні знань. Відзначається, що слово "big" у назві підходу не зовсім коректне, тому що дані не бувають "більшими" (big), а бувають лише "довгими" (long) і "широкими" (wide)". "Довжина" даних визначається кількістю доступних для аналізу спостережень конкретного фактора. "Ширина" же даних говорить про кількість функціональних залежностей, що існують між різними факторами.

Ці дві характеристики, що визначають, наскільки "більшими" даними розташовує дослідник, працюють прямо протилежно. Чим "довше" дані, тем краще - тем точніше алгоритм буде пророкувати значимість конкретного фактора. "Ширина"

же даних приводить до серйозних проблем з обчисленням. Тому обмежувати "ширину" пропонується експертам із числа людей.

Big Data можна визначити як підхід в одержанні нового знання через аналіз більших масивів даних. Уже зараз даний підхід використовується в різних галузях для прогнозування майбутнього.

Технології Big Data – серія підходів, інструментів і методів обробки структурованих і неструктурованих даних величезних обсягів і значного різноманіття. Дані технології застосовуються для одержання сприйманих людиною результатів, ефективних в умовах безперервного приросту, розподілу інформації із численних вузлів обчислювальної мережі. Вони сформувалися наприкінці 2000- х років як альтернативи традиційним системам керування базами даних і розв'язкам класу Business Intelligence. У цей час більшість найбільших постачальників інформаційних технологій для організацій у своїх ділових стратегіях використовують поняття "великі дані", а основні аналітики ринку інформаційних технологій присвячують концепції виділені дослідження.

У сучасних умовах організації створюють велика кількість неструктурованих даних, таких як текстові документи, зображення, відеозаписи, машинні коди, таблиці і т.д. Уся ця інформація зберігається в безлічі репозиториев, часом навіть за межами організації. Компанії можуть мати доступ до величезного масиву власних даних і не мати необхідних інструментів, які могли б установити взаємозв'язки між цими даними й зробити на їхній основі значимі висновки. Традиційні методи аналізу інформації не можуть угнатися за величезними обсягами постійно зростаючих і обновлюваних даних, що в підсумку й відкриває дорогу технологіям Big Data.

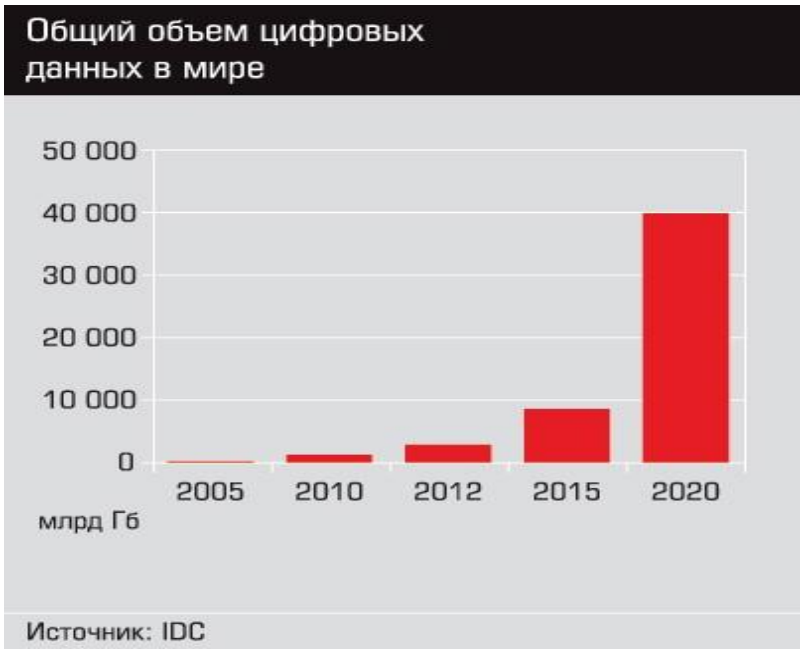


Рисунок 1.1 – Загальний обсяг цифрових даних у світі

Можна виділити наступні особливості технологій Big Data:

- робота з інформацією величезного обсягу й різноманітного складу;
- інформація досить часто обновляється й перебуває в різних джерелах;
- якісно одмінний метод відкриваючої аналітики для виявлення практичних знань, які безпосередньо монетизируются в прибуток;
- наочне відображення звітів і можливості сценарного аналізу;
- ціль застосування технологій Big Data – збільшення ефективності роботи, створення нових продуктів і підвищення конкурентоспроможності.

Обсяг інформації на підприємстві неухильно росте за рахунок даних, отриманих з датчиків, вимірювальних і "розумних" обладнань. Самими перспективними обладнаннями вважаються датчики, які можуть передавати дані в режимі реального часу. Усі обладнання на підприємстві за допомогою таких датчиків можуть бути об'єднані в мережу, а технології Big Data дозволять обробляти інформацію, що надходить із них, і проводити необхідні заходи в автоматичному режимі. Наприклад, підприємства можуть за допомогою датчиків одержувати щохвилинні дані про стан свого встаткування й на основі цих даних пророкувати оптимальний час для заміни й обслуговування. Занадто рання заміна приведе до додаткових витрат, а пізня – до втрати прибутки внаслідок простою встаткування. По

оцінці компанії Cisco, до 2017 р. буде існувати більш 1,7 млрд межмашинных з'єднань [34, с. 18].

Сфери діяльності, у яких прогнозується найбільший ефект від застосування Big Data, представлені на мал. 1.2.

Рисунок 1.2 – Сфери діяльності з найбільш відчутним прогнозованим ефектом від застосування Big Data

Технології Big Data можуть бути полезны для розв'язку наступних завдань:

- прогнозування ринкової ситуації;
- маркетинг і оптимізація продажів;
- ефективне сегментування клієнтів;
- удосконалювання товарів і послуг;
- прийняття більш обґрунтованих управлінських розв'язків на основі аналізу Big Data;
- оптимізація портфеля інвестицій;
- підвищення продуктивності праці;
- ефективна логістика;

− моніторинг стану основних фондів.

Методика й інструменти роботи зі структурованими даними вже давно створені. Це реляційна модель даних і системи керування базами даних. Але в сучасних умовах підприємствам потрібно обробляти більші обсяги неструктурованих даних різних типів, а для цієї роботи колишні методи не зовсім підходять.

У сфері ритейла інструменти Big Data використовуються вже більш десяти років. Це стало можливим після того, як організації стали задіяти у своїй діяльності "Систему керування взаєминами із клієнтами" (CRM, Crm- Система, скорочення від англ.: Customer Relationship Management), яка дозволяє побільшати продуктивність за рахунок автоматизації стратегій взаємодії із замовниками (клієнтами), оптимізації маркетингу й поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів, історію взаємин з ними, установлення й поліпшення бізнес- процесів, а також наступного аналізу даних. джерелами формування Big Data про споживачів стали виступати різні відомості про соціально-демографічні характеристики індивідів, що роблять придбання товарів або послуг, історія їх покупок, детальна інформація з кожного чека, відомості із соціальних мереж, пошукові запити в Інтернеті.

Сьогодні можна із упевненістю говорити про те, що використання Big Data дозволяє організаціям вивчити споживачів своїх товарів і послуг настільки добре, що в багатьох випадках вони можуть передбачити поведінка клієнтів з досить високою точністю.

Очевидно, що в сучасному суспільстві між споживанням товарів і приналежністю до соціальних груп існує тісний зв'язок, обумовлена характером використання матеріальних ресурсів індивідами з різним соціальним станом. Використання Big Data у дослідженнях споживчої поведінки є необхідністю сучасного часу. Даний підхід дозволяє суттєво розширити набір аналізованих ознак, а також дати більш об'єктивні дані. Основними напрямками досліджень із використання Big Data у соціології можуть стати: аналіз закономірностей поведінки

індивідів, що ставляться до певних груп споживачів, а також виявлення схованих потреб індивідів.

Глобальні сховища даних продовжують зростати. Представлений звіт аналітичної компанії IDC під назвою «Digital Universe Study» в середині 2011 року (який був організований компанією EMC) передбачає, що загальний світовий обсяг генерованих і тиражованих даних може досягати в 2011 році близько 1,8 зеттабайт (1,8 трлн гігабайт). Це приблизно в 9 разів більше, ніж те, що було створено в 2006 році [41, .с 95].

Проте, концепт «Big Data » означає набагато більше, ніж просто величезні обсяги інформації. Проблема полягає не в тому, що організації генерують величезні обсяги даних, але більшість з них представлені в форматі, який не дуже добре вписується в традиційний структурований формат бази даних. Це веб-журнали, відео, текстові документи, машинний код, або, наприклад, картографічні дані. У результаті, корпорація може мати доступ до величезного обсягу своїх даних і не мати необхідних інструментів для встановлення зв’язків між цими даними, автоматизованого формулювання конструктивних висновків на основі аналізу цих даних. Крім того, дані зараз оновлюються частіше, і ми маємо ситуацію коли традиційні методи аналізу інформації не можуть опрацювати величезні обсяги даних, що постійно оновлюються, і це, в кінцевому рахунку, прокладає шлях для технологій Big Data.
У соціальних груп існує тісний зв'язок, обумовлена характером використання матеріальних ресурсів індивідами з різним соціальним станом. Використання Big Data у дослідженнях споживчої поведінки є необхідністю сучасного часу. Даний підхід дозволяє суттєво розширити набір аналізованих ознак, а також дати більш об'єктивні дані. Основними напрямками досліджень із використання Big Data у соціології можуть стати: аналіз закономірностей поведінки індивідів, що ставляться до певних груп споживачів, а також виявлення схованих потреб індивідів.

Глобальні сховища даних продовжують зростати. Представлений звіт аналітичної компанії IDC під назвою «Digital Universe Study» в середині 2011 року (який був організований компанією EMC) передбачає, що загальний світовий обсяг генерованих і тиражованих даних може досягати в 2011 році близько 1,8 зеттабайт

(1,8 трлн гігабайт). Це приблизно в 9 разів більше, ніж те, що було створено в 2006 році [41, .с 95].

Проте, концепт «Big Data » означає набагато більше, ніж просто величезні обсяги інформації. Проблема полягає не в тому, що організації генерують величезні обсяги даних, але більшість з них представлені в форматі, який не дуже добре вписується в традиційний структурований формат бази даних. Це веб-журнали, відео, текстові документи, машинний код, або, наприклад, картографічні дані. У результаті, корпорація може мати доступ до величезного обсягу своїх даних і не мати необхідних інструментів для встановлення зв'язків між цими даними, автоматизованого формулювання конструктивних висновків на основі аналізу цих даних. Крім того, дані зараз оновлюються частіше, і ми маємо ситуацію коли традиційні методи аналізу інформації не можуть опрацювати величезні обсяги даних, що постійно оновлюються, і це, в кінцевому рахунку, прокладає шлях для технологій Big Data.

Big Data (Big Data) в інформаційних технологіях за визначенням К. Лінч, Д. Леней – набір методів та засобів опрацювання структурованих і неструктурованих різнотипних динамічних даних великих обсягів з метою їх аналізу та використання для підтримки прийняття рішень. Є альтернативою традиційним системам управління базами даних і рішенням класу Business Intelligence. До цього класу відносять засоби паралельного опрацювання даних (NoSQL, алгоритми MapReduce, Hadoop) [15, с.48].

На думку компанії DCA (Data-Centric Alliance) під Big Data розуміють не якийсь конкретний об'єм даних і навіть не дані, а методи їх обробки, які дозволяють розподілено обробляти інформацію. Ці методи можна застосовувати як до великих масивів даних (таких як дані всіх сторінок в мережі Інтерет), так і до малих масивів (інформація про денні поступлення товару в магазин).

Визначальними характеристиками для Big Data є обсяг (volume, в сенсі величини фізичного обсягу), швидкість (velocity в сенсах як швидкості приросту, так і необхідності високошвидкісної обробки та отримання результатів), різноманіття (variety, в сенсі можливості одночасної обробки різних типів структурованих і слабоструктурованих даних).

Завдання, що виникають під час опрацювання, обробки, інтерпретації, збору та організації Big Data, з'явилися в численних секторах, включаючи бізнес, промисловість, некомерційні організації. Обсяги даних, такі як операції замовника у роздрібній торгівлі, моніторинг погоди, бізнес-аналіз, можуть швидко випереджувати потужність традиційних методів та інструментів аналізу даних. З'явилися нові методи та інструменти, включаючи бази даних NoSQL, MapReduce, обробка природної мови, машинне навчання, візуалізація, придбання, і серіалізація. Усе це стає необхідним, щоб повною мірою усвідомити, що відбувається, коли зростають Big Data , як вони застосовуються і де починають відігравати вирішальну роль. Також необхідо знати вимоги до існуючих методів розроблення систем і аналізу даних.

Big Data є терміном, який використовується для ідентифікації наборів даних, з якими ми не можемо впоратися з використанням існуючих методологій та програмних засобів через їх великий розмір і складність. Багато дослідників намагаються розробити методики і програмні засоби для передачі даних або видобування інформаційних гранул з Big Data [27, с. 54].

В якості джерел Big Data можна виділити пристрої та людей. Приклади перших джерел: національні та міжнародні проекти, такі як Великий адронний коллайдер (LHC) в ЦЕРН, Лабораторія фізики елементарних частинок в Європі, Великий синоптичний оглядовий телескоп на півночі Чилі; промисловість (SCADA, фінанси і т.д.).

Приклади другого типу джерел на рисунку 1.2 : соціальні мережі, охорона здоров'я, роздрібна торгівля, особисті дані про місцезнаходження, управління громадським сектором і т.д. Для збору і обробки Big Data доцільно використовувати технології хмарних обчислень. Хмарні обчислення – це нова парадигма для розміщення кластерів даних і надання різних послуг локальною мережею або через Інтернет.

Хостинг кластерів даних дає змогу клієнтам зберігати та обчислювати величезну кількість даних у хмарі. Оскільки розмір окремих баз даних зростає швидко і подолав петабайтний бар'єр (наприклад, бази даних соціальних мереж),то онлайн опрацювання в режимі OLAP таких обсягів практично неможливе

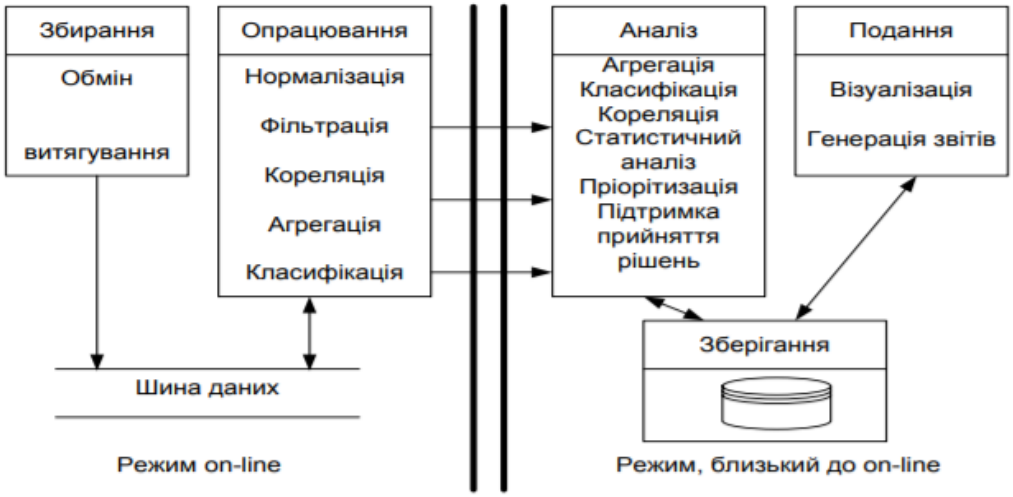


Рисунок 1.3 – Порівняльна характеристика OLAP та Big Data

На рисунку 1.3 подано відомості щодо ряду інструментів опрацювання Big Data з відкритим вихідним кодом, які надаються через інфраструктури хмарних обчислень. Більшість інструментів забезпечується Apache і випущені під ліцензією Apache. Усі ці продукти згруповано за типами задач, що виникають в процесах опрацювання Big Data.

1.2. Важливість обробки даних з соцмереж

Сьогодні соціальні медіа стали найефективнішим засобом спілкування та хобі будь-якої людини буття.

З розповсюдженням розумних пристроїв та послуг, розширених соціальними мережами, воно завойовує значення і приваблює людей у великій кількості. Взаємодія між людьми через соціальні мережі є потенційним джерелом неструктурованих, детальніших та масштабніших цифрових даних. Термін "великі дані" є Поширений і включає величезні обсяги даних, аналітику в соціальних мережах, управління даними наступного покоління можливості, дані в реальному часі та багато іншого. Це також стенограма для просування технологічних тенденцій відкрити двері для нового підходу для розуміння останніх тенденцій та прийняття розумних рішень (Schroeck,

Shockley, Smart, Morales & Tufano, 2014). Більшість сплесків великих даних є дикими і містять слова, зображення, відео або їх поєднання. Він вибухає і обов’язково зростатиме швидшими темпами наступні роки. Хоча лише декілька відсотків цих даних використовуються ефективно, стверджують керівники підприємств що дані,

отримані із соціальних мереж, мають величезну підтримку при прийнятті розумних рішень. Інструменти є розробляється для вилучення даних, отриманих із соціальних мереж, для оцінки поведінки споживачів та перетворення її на активна інформація. У цій главі подано розуміння великих даних, отриманих із соціальних мереж, аналітики та його споживання з точки зору бізнесу.

Згідно з ресурсом hootcuite.com середньостатистичний користувач відвідує Твіттер один раз на 2.5 години. На рис 1.4 зображана статистика корисування соцмережою.

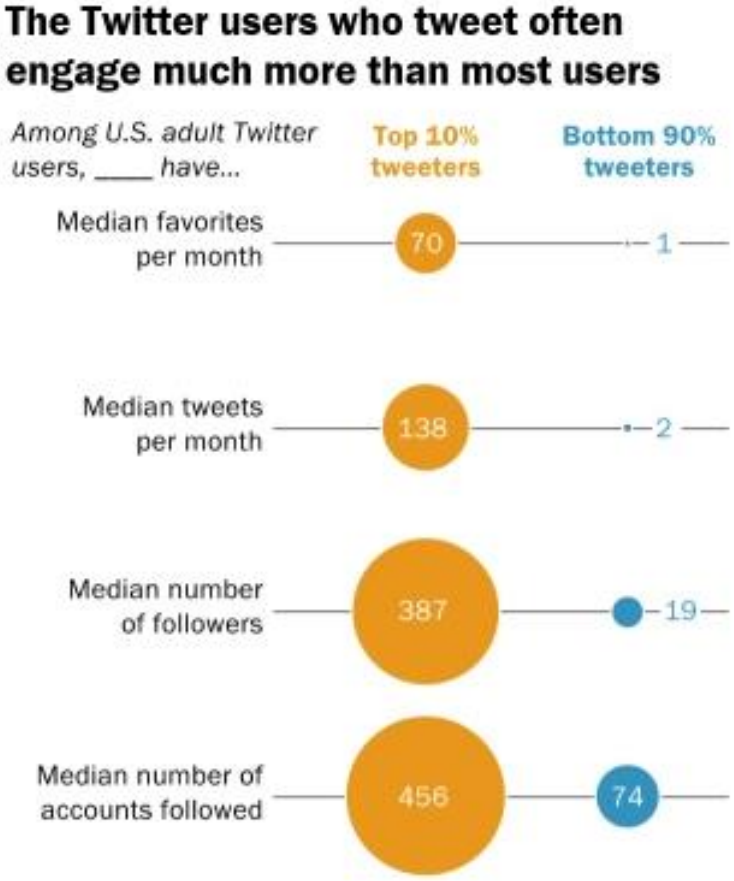


Рисунок 1.4 – Статистика користування Твіттер

Інформаційні технології досягли вершини, і зараз домінують дві рушійні сили - BigData та соціальні мережі. Великі дані охоплюють широкий спектр навантажень на видобуток даних, вилучені з різних джерел, результати яких викликають великий інтерес у керівників підприємств та аналітики з усіх сегментів галузі. Дані соціальних медіа вибухають експоненціально і є джерелом уявлень про поведінку людини. Вилучення інтелектуальної інформації від такого величезного обсягу, різноманітності та швидкості передачі даних, в контексті бізнес-вимог потребою

кожного бізнесу. Тому нові інструменти та методи, що спеціалізуються на аналітиці великих даних, мають вирішальне значення, адже правильно проаналызвані дані з соціальних мереж здатна просунути бізнес на новий рівень.

Переваги, які сучастний бізнес отримує аналізуючи соціальні мережі:

- персоналізація;
- прийняття рішень;
- ефективність інвестицій;
- статистика продукту.

Персоналізація

Великі дані дозволяють персоналізувати, дозволяючи брендам підходити до своїх клієнтів більш персоналізованим способом, виходячи з їх вибору та вподобань. Це дає глибоке розуміння та цілісне розуміння аудиторії, що допомагає компаніям створювати спеціальні комунікації для них, щоб покращити утримання та підвищити їхню довіру.

Завдяки великим обсягам даних брендам стане легше відображати лише ті реклами, які цікавлять споживачів, перетворюючи рекламу на ненав’язливий досвід. Реклама буде націлена на основі публікацій користувачів у соціальних мережах, того, що вони дивляться та діляться і т. Д. Завдяки персоналізованій рекламі маркетологи зможуть зміцнити свої стосунки з користувачами соціальних мереж і перетворити їх на клієнтів після визначення найбільш ефективної платформи час та формат їхніх оголошень.

Прийняття рішень

Великі дані дозволяють маркетологам визначати тенденції соціальних медіа та отримувати уявлення, які можуть бути використані для прийняття рішень про залучення, наприклад, з якими користувачами спілкуватися, з якою групою користувачів слід отримувати маркетингові електронні листи тощо. Це також полегшує відстеження демографічні показники, щоб вирішити, на яку платформу соціальних мереж націлити.

Підприємства можуть легко зрозуміти настрої ринку за допомогою великих даних, що дозволяє їм будувати стратегії виграшу. Замість того, щоб покладатися

виключно на минулі результати, щоб з’ясувати, які покращення потрібні, великі дані допомагають приймати обґрунтовані рішення, щоб краще відповідати майбутнім потребам та очікуванням споживачів.

Ефективність інвестицій

Великі дані корисні для відстеження ефективності кампаній у соціальних мережах та виявлення поступових змін рентабельності інвестицій. Це також дозволяє маркетологам тестувати свої кампанії перед її запуском, аналізувати результати, вносити зміни в кампанію, якщо потрібно, і повторно тестувати їх. Прогнозуючі аналітичні інструменти дозволяють компаніям приймати рішення щодо того, коли зупинити кампанію, щоб уникнути збитків.

Виводячи ефективну інформацію з великих даних, бізнес отримує уявлення про час піків клієнтів, їх уподобання, поведінку тощо, що призводить до підвищення ефективності кампанії в соціальних мережах. Маркетологи можуть отримати важливу інформацію про процес, який виконували їх клієнти, від першого етапу циклу покупки до взаємодії після покупки, що змушує їх точно налаштовувати кампанію на кожному етапі циклу.

Статистика продукту

Маркетологи соціальних мереж можуть ефективно використовувати великі дані, щоб судити про майбутні моделі та тенденції покупок. Великі дані збільшують впевненість у тому, чого хочуть споживачі, коли вони цього хочуть і як вони цього хочуть. Це дає підприємствам уявлення про те, якими повинні бути їх нові продукти.

Підприємства можуть використовувати великі дані для аналізу вибору людей, їх скарг, того, чого не вистачає, несправностей у продуктах тощо. Це дозволить їм вносити зміни в поточний продукт та виходити з новими інноваційними продуктами.

1.3. Характеристика даних з соцмереж

Хоча більшість соціальних мереж дозволяють оримати певні дані використовуючи їх публічні API, сирі дані зазвичай недоступні, адже дуже мало джерел соціальних даних надають їх у відкритий доступ. Ресурси новин, такі як, Reuters та Bloomberg дозволяють доступ до їх даних тільки за певну плату. Соціальна мережа твіттер одна з небагатьох що активно веде дослідження в напрямку аналізу

даних, саме тому виконавши запит можна отримати доступ до твітів корисувачі. Саме тому ресурсом даних був обраний саме Twitter.

Далі розглянемо повідомлення користовачів даной мережі з точки зору аналізу тексту, виявимо складнощі та особливості.

Першою особливістю є неструктурованість даних. “Якість проти кількості” основний принцип при анаілзі даних, адже багато якостей аналітичних моделей визначаються саме при розгляді якості та кількості даних. Далі важливим фактором для аналітики є природа даних. Залежно від природи даних можна визничити степінь структурованості. Неструктуровані дані є нецілисними. Тому однією з найважливіших задач є очищення та структуризація даних. Очищення передбачає видалення помилок орфографії та пунктуації, зайвих символів, приведення тексту до нормальної форми. Очищення даних є необхідним етапом для високоякісної обробки тексту. Переглянувши види та джерела сирих даних, ми можемо перейти "очищення" даних для видалення неправильної, непослідовної або відсутньої інформації.

Другою особливістю є те що текст не є чистим, тим не менш для правильного аналізу треба враховувати такі речі як смайли та хештеги.

Але не дивлячись не недоліки подібних даних є багато переваг, через які вони є дуже привабливими для використання аналітиками. У зв'язку зі своїм невеликим розміром, twitter-повідомлення, як правило, мають підвищену змістовністю, що робить їх привабливими з точки зору контент-аналізу і виявлення трендів (наприклад, для передбачення спалахів грипу, передбачення біржових подій, аналізу настроїв в соціальних групах, швидкості поширення інформації, і т.д.). Розробники сервісу надають відкритий інтерфейс програмування додатків (API), що дозволяє досить легко витягувати тексти повідомлень необхідної тематики з бази даних Twitter'a для їх подальшого аналізу.

Висновки по розділу 1

Таким чином, у данному розділі було визначенно поняття, сутність, види та атрибути BigData (BD): обсяг, швидкість, структурованість. За допомогою даних атрибутів можна висунути вимоги до функціональності ПЗ.

Технологіі BigData надають можливість проаналізувати наукові дані, зробити висновки на основі проведених досліджень, що сприяє науково-технічному прогресу.

В даному розділі булли проаналізовані проблеми та задачі для яких використовується BigData аналітика, зокрема аналіз соціальних мереж. Було виявлено що великі дані є рушійною силою для вирішення багатьох задач сучасного бізнесу і не тільки.

Також у даному розділі були проаналізовані переваги та недоліки аналізу текстових повідомлень та обраний ресурс даних. Незважаючи на неструктурованість та неповноту даних, повідомлення користувачів Твіттеру є надзвичайно змістовними.

Таким чином можна зробити висновок, що BD технології є дуже актуальними в наш час і дослідження цього напрямку є перспективним.

2. АНАЛІЗ ОСНОВНИХ ПІДХОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ОБРОБКИ ВЕЛИКОЇ КІЛЬКОСТІ ТЕКСТОВИХ ПОВІДОМЛЕНЬ ЗАСОБАМИ BIGDATA

2.1. Обробка і методи аналізу Big Data

З точки зору обробки в основу технологій Big Data покладені два основних принципи: розподіленого зберігання даних; розподіленої обробки, з урахуванням локальності даних.

Розподілене зберігання вирішує проблему великого обсягу даних, дозволяючи організовувати сховище з довільного числа окремих простих носіїв. Зберігання може бути організовано з різним ступенем надмірності, забезпечуючи стійкість до збоїв окремих носіїв.

Розподілена обробка з урахуванням локальності даних означає, що програма обробки доставляється на обчислювач, що знаходиться якомога ближче до оброблюваних даних. Це принципово відрізняється від традиційного підходу, коли обчислювальні потужності і підсистема зберігання розділені і дані повинні бути доставлені на обчислювач. Таким чином, технології Big Data спираються на обчислювальні кластери з безлічі обчислювачів, забезпечених локальної підсистемою зберігання. Доступ до даних і їх обробка здійснюються спеціальним програмним забезпеченням.

Основною метою рішень в даній області є розподіл зберігання і обробки даних.

На сьогоднішній день існує величезна кількість архітектурних рішень і інструментів, що застосовуються для великих даних. Найпоширенішою основою для побудови систем великих даних є екосистема Hadoop – проект фонду Apache Software Foundation, який представляє собою набір вільно розповсюджуваних утиліт, бібліотек і фреймворків для розробки розподілених програм і їх виконання на кластерах.

На думку автора все різноманіття архітектурних рішень для додатків великих даних можна класифікувати за характеристиками: типи великих даних і застосовність хмарних обчислень. За типом оброблюваних даних можна виділити два класи

додатків: системи пакетної обробки великих даних (Batch Processing / Data Processing); системи потокової обробки великих даних (Stream Processing).

На малюнках 2.1 та 2.2 зображені діаграми потоку даних цих двух типів систем.

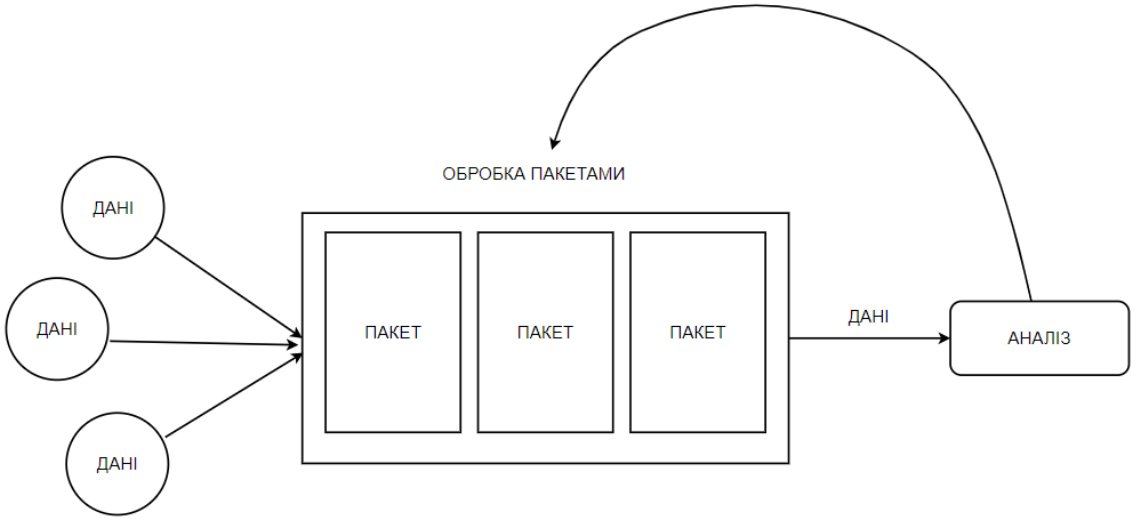


Рисунок 2.1 – Діаграма пакетної обробки даних

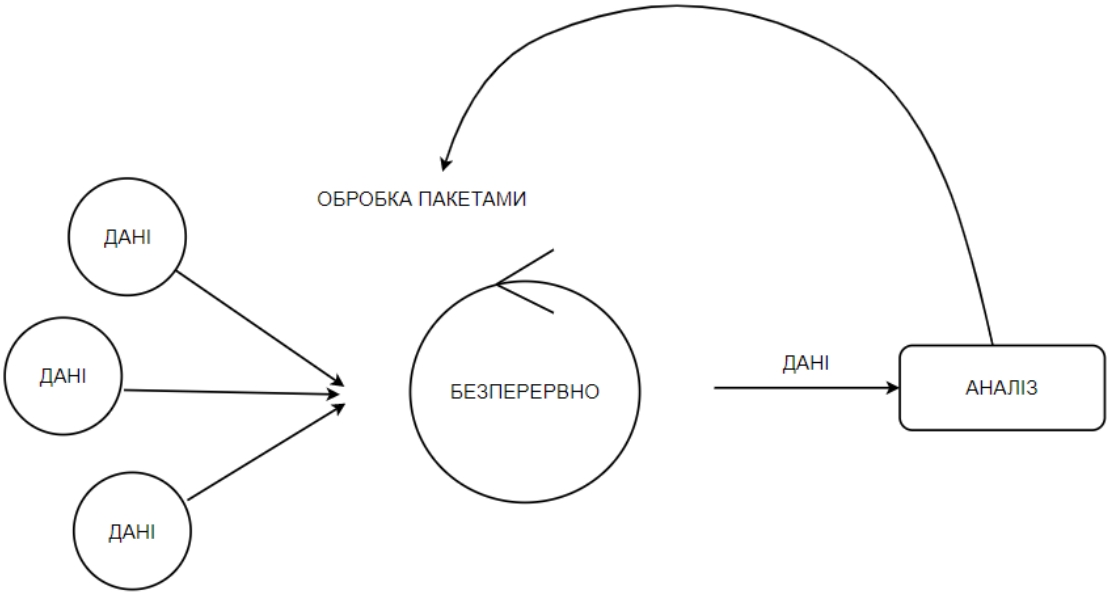


Рисунок 2.2 – Діаграма потокової обробки даних

Системи пакетної обробки обробляють вже накопичені за певний період часу дані. Наприклад, дані за день можуть містити мільйони записів і бути збережені у вигляді файлу. Цей файл може піддаватися обробці в кінці поточного дня або на

наступний день. При цьому, можливо, що для обробки денних даних буде потрібно досить багато часу. Hadoop MapReduce, за оцінкою багатьох фахівців, найкраща платформа для обробки даних в пакетному режимі. На рис. 2.3 показана схема обробки даних за допомогою MapReduce.

Системи пакетної обробки застосовні у випадках, коли немає необхідності обробки в реальному часі, а основною метою аналізу є отримання детальної інформації.

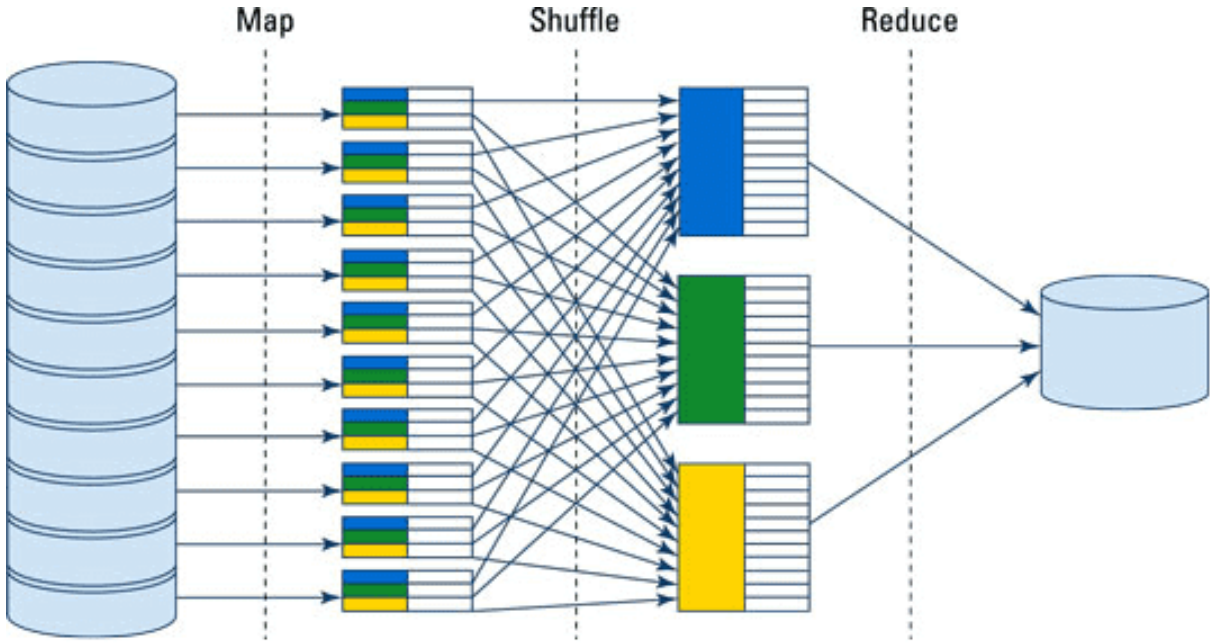


Рисунок 2.3 – Схема пакетної обробки даних в Hadoop MapReduce

Системи потокової обробки застосовуються в ситуації, коли необхідно отримувати аналітичні результати в реальному часі. Передача потокового обробка дозволяє обробляти дані в реальному часі в міру їх надходження і швидко виявляти певні ситуації протягом невеликого періоду часу з моменту отримання даних. Основною причиною високої швидкості потокової обробки є те, що система аналізує дані до їх запису в файлову систему.

Передача потокового обробка корисна для таких завдань, як моніторинг активності сайтів, виявлення шахрайства при проведенні транзакцій в бізнесі або онлайн навчанні. Наприклад, при обробці даних транзакцій в потоковому режимі, можливе виявлення аномалій, що сигналізують про шахрайство в режимі реального часу для їх блокування. На рис. 2.4 приведена схема обробки даних в реальному часі з використанням Apache Spark.

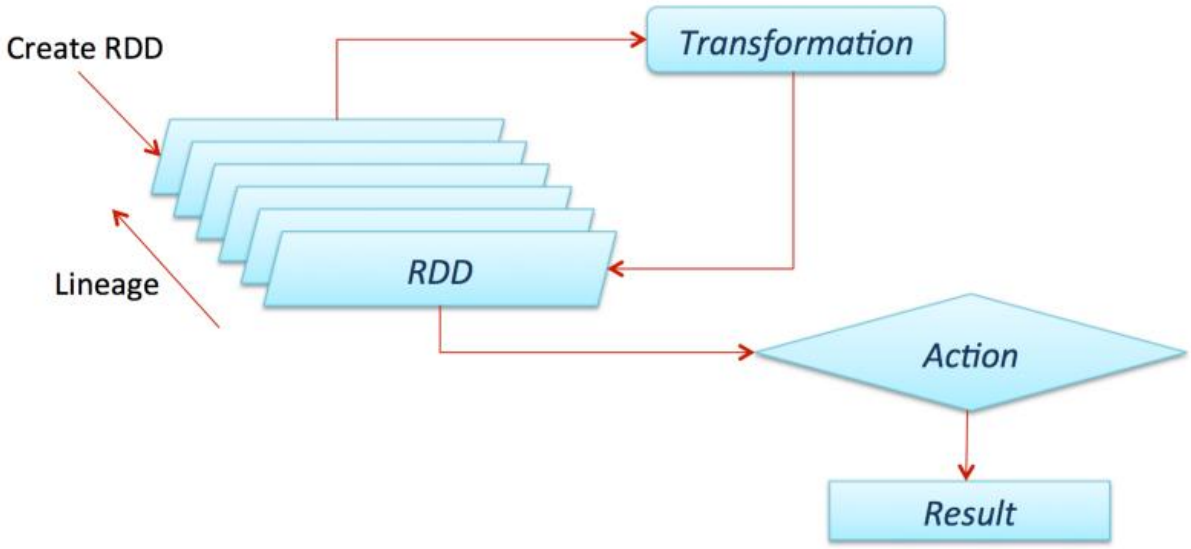


Рисунок 2.4 – Обробка великих даних в реальному часі з використанням Apache Spark

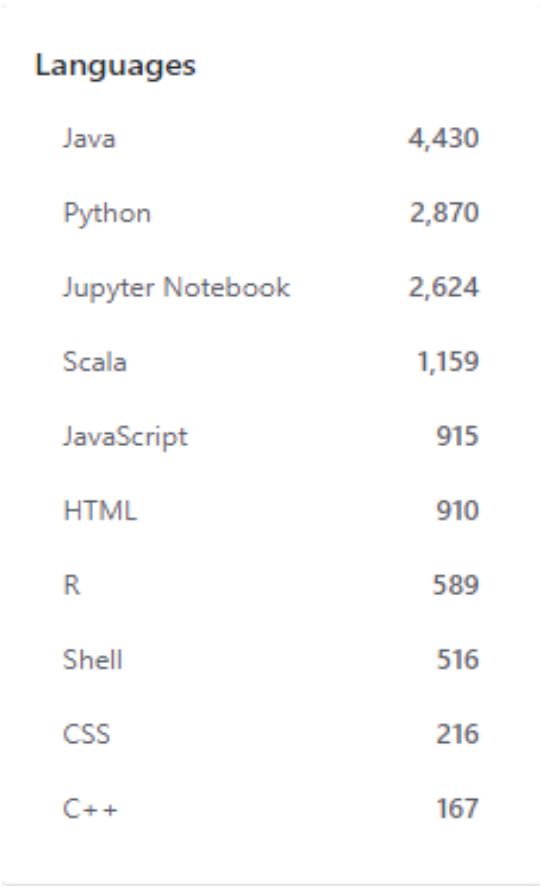
Певний інтерес для побудови систем потокової обробки даних можуть представляти системи, побудовані на основі лямбда архітектури [35]. Однак, в даній роботі це питання не розглядається.

2.2. Аналіз програмних засобів Big Data

Щоб визначити найбільш використовувані програмні засоби була зібрана статистика стосовно кількості проектів на найбільшій платформі що надає репозиторії github.com

За пошуковим запитом “BigData” найбільша кількість проектів на мові Java.

На рис 2.5. можна побачити статистику з кількості проектів.



Languages	
Java	4,430
Python	2,870
Jupyter Notebook	2,624
Scala	1,159
JavaScript	915
HTML	910
R	589
Shell	516
CSS	216
C++	167

Рисунок 2.5 – Кількість BigData проектів на кожній мові на github.com

Як можна побачити, для програмування систем зберігання і аналізу великих даних може застосовуватися досить великий набір мов програмування: Python, R, Java і т.д. В роботі для реалізації елементів систем зберігання і аналізу великих даних найпопулярнішою є саме Java з наступних причин:

- платформи Java SE і Java EE відмінно підходять для розробки добре масштабованих, високонавантажених додатків зберігання і обробки великих даних; мова Java має низку стандартних можливостей для побудови систем обробки великих даних, найцікавішими з яких є:
- базові методи обробки рядків, що включають в себе методи класу String, змінювані клас StringBuilder, потокобезпечна клас StringBuffer, парсер рядків – клас StringTokenizer, а також потужний механізм обробки тексту, заснований на регулярних виразах – Regular Expressions;
- методи функціонального програмування за допомогою лямбда-виразів, введених в Java 8, дозволяють використовувати потужні і виразні засоби створення додатків;

- потокові можливості колекцій (collections streaming), що полегшує роботу з колекціями даних;
- сторонні Java бібліотеки такі як, дають широкий спектр можливостей для більш простою і надійною реалізації алгоритмів зберігання і аналізу великих даних:
- Google Guava (https://github.com/google/guava) і Apache Common Collections (https://commons.apache.org/collections), що розширюють технологію роботи з колекціями;
- Apache Commons IO (https://commons.apache.org/io) для реалізації процесу завантаження / вивантаження даних;
- AOL Cyclops-React (https://github.com/aol/cyclops-react) для більш функціонального паралельного потокового обміну; підтримка інтерактивного обчислення (аналогічно терміналу Python) Read-Evaluate-Print Loop (REPL) в Java 9. Поставка Java 9 включає в себе інтерактивний термінал jshell, що зробило мову Java повністю придатним для використання на етапі дослідного аналізу даних - Exploratory Data Analysis (EDA).

Серед Java платформ була зібрана аналогічна статистика. На рис. 2.6 можна побачити кількість проектів на найпопулярніших BigData Java-платформах.

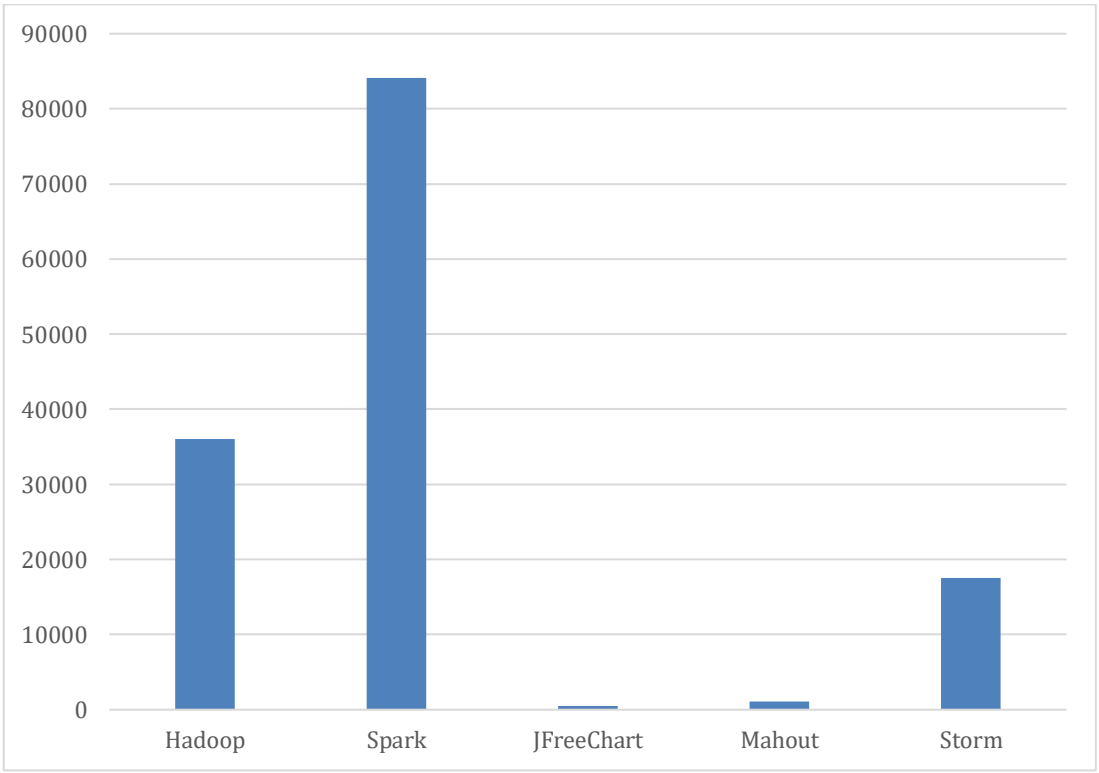


Рисунок 2.6 – Кількість BigData проектів на BigData платформах на github.com

Hadoop – найпопулярніша платформа для пакетної обробки, Spark для потокової. Саме тому ці програмні засоби й будуть використані в дослідженні.

Hadoop

Hadoop - це вільно поширюваний набір утиліт, бібліотек і фреймворк для розробки і виконання розподілених програм, що працюють на кластерах з сотень і тисяч вузлів. Ця основоположна технологія зберігання і обробки великих даних (Big Data) є проектом верхнього рівня фонду Apache Software Foundation.

Система Hadoop це вільно поширюваний набір програмних засобів для розробки і виконання розподілених програм, що працюють на кластерах з багатьох сотень, і іноді навіть тисяч вузлів. У Hadoop передбачено дублювання на випадок виходу з ладу вузлів, система підтримує декілька робочих копій даних. Обробка продубльованих даних може бути перерозподілена на інші, недефектного вузли. Робота Hadoop заснована на принципі паралельної обробки даних - це дозволяє збільшити швидкість роботи. Обсяги оброблюваної інформації вимірюються петабайт. Платформа написана на мові Java і добре працює в середовищі Linux.

Ядро системи Hadoop складається з Hadoop Common, який забезпечує абстракції файлової системи та операційної системи, MapReduce і розподілену файлову систему Hadoop (HDFS). У загальному пакеті Hadoop містяться файли та сценарії для Java-архіва (JAR), необхідні для запуску Hadoop. Для ефективного планування роботи кожна Hadoop-сумісна файлова система повинна забезпечувати інформацію про місцезнаходження – ім'я стійки (або, точніше, мережевого перемикача), де є робочий вузол. Програми Hadoop можуть використовувати цю інформацію для виконання коду на вузлі, де є дані, а у разі відсутності -— на тій же стійці / комутаторі, щоб зменшити магістральний трафік. HDFS використовує цей метод при реплікації даних для надлишковості даних на декількох стійках. Цей підхід зменшує наслідки відключення живлення або перемикання живлення в стойці; якщо відбудеться яке-небудь з цих апаратних збоїв, дані залишатимуться доступними. Багатокомпонентний кластер Hadoop Невеликий Hadoop кластер включає в себе єдиний майстер і кілька робочих вузлів. Головний вузол складається з Job Tracker, Task Tracker, NameNode і DataNode. Рабський або робочий вузол виступає як DataNode, так і TaskTracker, хоча

це можливо лише для робочих вузлів лише для даних і обчислень. Вони зазвичай використовуються тільки в нестандартних програмах. Hadoop вимагає Java Runtime Environment (JRE) 1.6 або новішої версії. Стандартні скрипти запуску та завершення роботи вимагають встановлення безпечної оболонки (SSH) між вузлами кластера.

У більшому кластері вузли HDFS управляються через спеціальний сервер NameNode для розміщення індексу файлової системи та додаткового імені NameNode, який може генерувати моментальні знімки структури пам'яті Namenode, тим самим запобігаючи пошкодженню файлової системи та втрату даних. Подібним чином, автономний сервер JobTracker може керувати плануванням роботи у різних вузлах. Коли Hadoop MapReduce використовується з альтернативною файловою системою, NameNode, Second NameNode і архітектура DataNode HDFS замінюються еквівалентами файлових систем.

Hadoop MapReduce — реалізація моделі програмування MapReduce для обробки великих об'ємів даних.

З часом, термін Hadoop почав вживатись не тільки щодо вищезгаданих базових модулів та підмодулів, а й до «екосистеми», тобто набору додаткових пакетів програмного забезпечення, які можуть встановлюватись поверх, або поряд з Hadoop, наприклад таких як Apache Pig, Apache Hive, Apache HBase, Apache Phoenix, Apache Spark, Apache ZooKeeper, Cloudera Impala, Apache Flume, Apache Sqoop, Apache Oozie, та Apache Storm.

MapReduce та HDFS в Apache Hadoop's були натхненними статтями Google про їх алгоритм MapReduce та Google File System.

Фреймворк Hadoop написаний переважно на Java, з частиною системного коду на C та утилітами командного рядка як shell скрипти. Хоча в програмах MapReduce звичайним є код на Java, для реалізації «map» та «reduce» частин користувацької програми можна використовувати будь-яку мову програмування завдяки «Hadoop Streaming». Інші проекти в екосистемі Hadoop надають багатші інтерфейси користувача.

HDFS – це розподілена, масштабована та портативна файлова система, написана на Java для рамок Hadoop. Деякі вважають це замість того, щоб бути магазином даних

через відсутність відповідності POSIX, але це забезпечує виконання команд оболонки та інтерфейсів Java (API), подібних до інших файлових систем. Hadoop ділиться на дві технології: HDFS і MapReduce. HDFS використовується для зберігання даних, і MapReduce використовується для обробки даних. HDFS має п'ять сервісів, які перелічені нижче.

1) Name Node. HDFS складається з лише одного Name Node, який ми називаємо його основним вузлом, який може відстежувати файли, керувати файловою системою та мати мета-дані та всі дані в ньому. Для того, щоб конкретний вузол Ім'я містив дані про кількість блоків, розташування, на якому вузлі даних зберігаються дані, а також де зберігаються репліки та інші деталі. Оскільки у нас є лише одне ім’я, ми називаємо це як Single Point Failure. Він має прямий зв'язок з клієнтом.
2) Secondary Name Node. Лише для того, щоб піклуватися про контрольні точки метаданих файлової системи, яка знаходиться в Data Node. Також відомий як вузол контрольної точки. Це допоміжний вузол для Name Node.
3) Job tracker. Job Tracker отримує запити , щоб скоротити виконання з клієнта. Робота трекера комунікує з Name Node, щоб дізнатись про місце розташування даних, як Job Tracker, буде запитано вузол імені для обробки даних. Name Node у відповідь надає метадані для роботи трекера.
4) Data Node. Node Data зберігає дані як блоки. Також відомий як підпорядкований вузол, і він зберігає фактичні дані в HDFS, який відповідає за читання та запис клієнта. Це підпорядковані процеси-демони. Кожен вузол даних надсилає повідомлення-Heartbeat до Name Node кожні 3 секунди і передає, що він живий. Таким чином, коли Data Node не отримує “серцебиття” з вузла даних протягом 2 хвилин, він припускає, що цей вузол даних неактивний і запускає процес реплікації блоку на деякому іншому вузлі даних.
5) TaskTracker. Це підпорядкований вузол для відстеження роботи, якийотримує завдання від Job Tracker. А також він отримує код від Job Tracker. Task Tracker застосовує отриманий код до файлу даних. Процес застосування цього коду в файлі відомий як Mapper

Угорі ієрархії знаходяться сервіси Master/ демони / вузли, а нижче – Slave Services. Master Services можуть спілкуватися один з одним і таким же чином Slave Services можуть спілкуватися один з одним. Name Node є основним вузлом, а Data Node (вузол даних) – це відповідний вузловий підлеглий і може спілкуватися один з одним.

На рис. 2.7 можна побачити типову конфігурацію кластеру Hadoop що містить всі вищеописані компоненти.

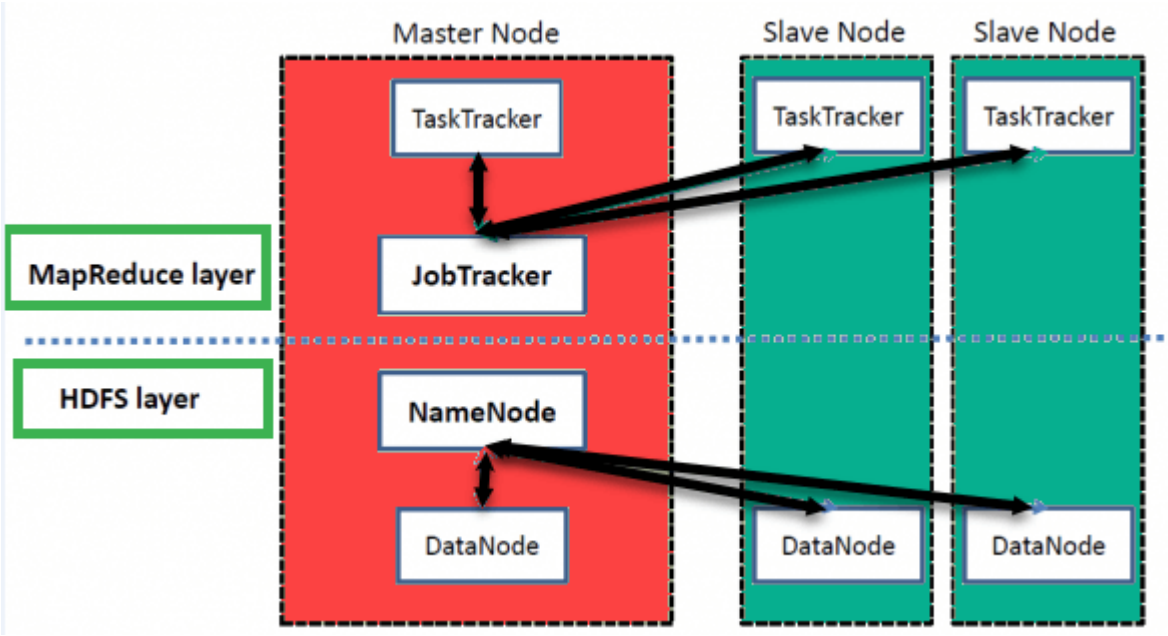


Рисунок 2.7 – Типовий кластер Hadoop

MapReduce – це модель розподіленої обробки даних, запропонована компанією Google для обробки великих обсягів даних на комп'ютерних кластерах.

Парадигму MpaReduce запропонувала компанія Google в свої статті MapReduce: Simplified Data Processing on Large Clusters

Програма MapReduce складається з процедури (або способу) *map*, яка виконує фільтрування та сортування (наприклад, сортування студентів за іменем у чергу, одну чергу для кожного імені) та метод *reduce*, який виконує операцію стиснення (наприклад, підрахунок кількості студентів у кожній черзі, що дає назви частот). Система MapReduce (також називається "інфраструктура" або "framework") організовує обробку шляхом сортування розподілених серверів, паралельно виконуючи різні завдання, керуючи всіма передачами та передачею даних між різними частинами системи та забезпечуючи резервування і відмовостійкість.

Модель є спеціалізацією стратегії split-apply-combine для аналізу даних. Був натхненним *map* та *reduce* функціями, що часто використовуються у функціональному програмуванні, хоча їх призначення в рамках MapReduce не є таким самим, як у їхніх оригінальних формах. Основний внесок структури MapReduce - це не фактичні map та reduce функції (які, наприклад, нагадують стандарти операцій з скороченням протоколу обробки повідомлень для передавання повідомлень 1995 року, але масштабованість та відмовостійкість досягається для різних додатків, оптимізуючи рушій виконання. Таким чином, однопоточна реалізація MapReduce, як правило, не буде швидшою, ніж традиційна (не MapReduce) реалізація; будь-які покращення, як правило, розглядаються лише за допомогою багатопотокових реалізацій на багатопроцесорних пристроях. Використання цієї моделі є корисною лише тоді, коли в процесі роботи відбувається оптимізоване розподілене переміщення (що зменшує вартість мережевого зв'язку) та характеристики відмовостійкості системи MapReduce.

MapReduce передбачає, що дані організовані у вигляді деяких записів. Обробка даних відбувається в 3 стадії:

1) Стадія Map. На цій стадії дані предобрабативаются за допомогою функції map (), яку визначає користувач. Робота цієї стадії полягає в передобробці і фільтрації даних. Робота дуже схожа на операцію map в функціональних мовах програмування - призначена для користувача функція застосовується до кожної вхідного відрізку. Функція map () застосована до однієї вхідний записи і видає безліч пар ключ-значення. Безліч - тобто може видати тільки один запис, може не видати нічого, а може видати кілька пар ключ-значення. Що буде знаходиться в ключі і в значенні - вирішувати користувачу, але ключ - дуже важлива річ, так як дані з одним ключем в майбутньому потраплять в один екземпляр функції reduce.

2) Стадія Shuffle. Проходить непомітно для користувача. У цій стадії результат виконання функції map «розбирається по кошиках» - кожна корзина відповідає одному ключу виведення стадії map. Надалі ці кошики послужать входом для reduce.

Зазвичай найважча стадія при виконанні Map-Reduce завдання - це стадія shuffle. Відбувається це тому, що проміжні результати (вихід mapper'a) записуються на диск, упорядковано і передаються по мережі. Однак існують завдання, в яких така поведінка видається не дуже доцільною.

3) Стадія Reduce. Кожен «кошик» із значеннями, сформований на стадії shuffle, потрапляє на вхід функції reduce ().

Функція reduce задається користувачем і обчислює фінальний результат для окремого «кошика». Множина всіх значень, повернутих функцією reduce (), є фінальним результатом MapReduce-завдання.

Кілька додаткових фактів про MapReduce:

- усі запуски функції map працюють незалежно і можуть працювати паралельно, в тому числі на різних машинах кластера;
- всі запуски функції reduce працюють незалежно і можуть працювати паралельно, в тому числі на різних машинах кластера;
- Shuffle всередині себе представляє паралельну сортування, тому також може працювати на різних машинах кластера.

Пункти дозволяють здійснити принцип горизонтальної масштабованості.

Функція map, як правило, застосовується на тій же машині, на якій зберігаються дані - це дозволяє знизити передачу даних по мережі (принцип локальності даних).

MapReduce – це завжди повне сканування даних без індексації. Це означає, що даний підхід не підходить, коли відповідь потрібно дуже швидко.

Spark

Apache Spark – це платформа розподілених кластерних обчислювальних пристроїв з відкритим кодом. Спочатку розроблений в Каліфорнійському університеті, AMPLab у Берклі. Пізніше кодова база Spark була передана в Фонд програмного забезпечення Apache, який підтримує його. Spark забезпечує інтерфейс для програмування цілих кластерів з неявним паралелізмом даних та відмовостійкістю.

Сьогодні Spark застосовується в багатьох найбільших компаніях, таких, як Amazon, eBay і Yahoo! Багато організацій експлуатують Spark в кластерах, що включають тисячі вузлів. Згідно FAQ по Spark, в найбільшому з таких кластерів налічується більше 8000 вузлів.

Фреймворк складається з п'яти компонентів: ядра і чотирьох бібліотек, кожна з яких вирішує певне завдання, як можна побачити на рис. 2.8.

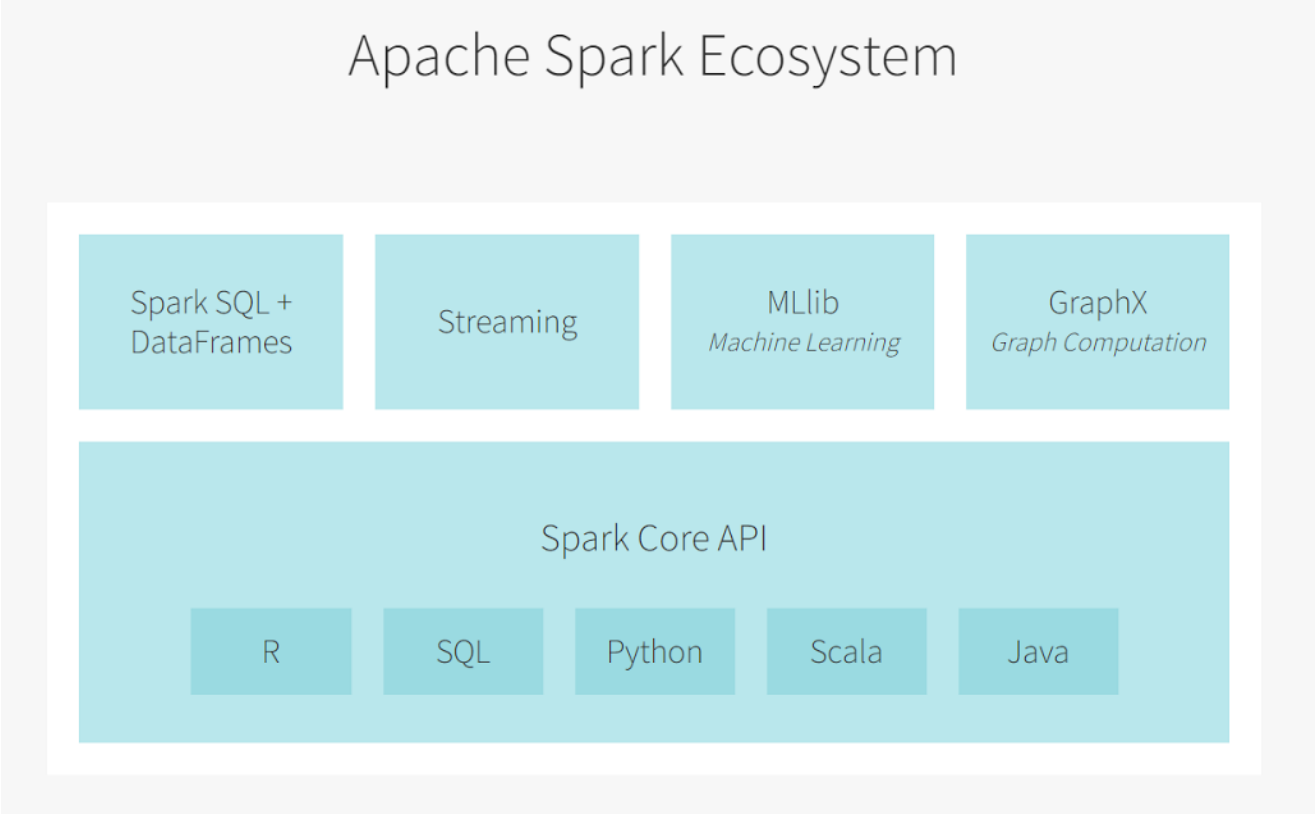


Рисунок 2.8 – Екосистема Spark

Ядро Spark – це базовий рушій для великомасштабної паралельної і розподіленої обробки даних. Ядро відповідає за:

– управління пам'яттю і відновлення після відмов;
– планування, розподіл і відстеження завдань у кластері;
– взаємодію з системами зберігання даних.

В Spark вводиться концепція RDD (стійкий розподілений набір даних) – незмінна відмовостійка розподілена колекція об'єктів, які можна обробляти паралельно. У RDD можуть міститися об'єкти будь-яких типів; RDD створюється шляхом завантаження

зовнішнього набору даних або розподілу колекції з основної програми (driver program). У RDD підтримуються операції двох типів:

1) Трансформації – це операції (наприклад, відображення, фільтрація, об'єднання і т.д.), що здійснюються над RDD; результатом трансформації стає новий RDD, що містить її результат.

2) Дії – це операції (наприклад, редукція, підрахунок і т.д.), які повертають значення, що отримується в результаті деяких обчислень в RDD.

Трансформації в Spark здійснюються в «ледачому» режимі - тобто, результат не обчислюється відразу після трансформації. Замість цього вони просто «запам'ятовують» операцію, яку слід провести, і набір даних (напр., Файл), над яким потрібно зробити операцію. Обчислення трансформацій відбувається тільки тоді, коли викликається дія, і його результат повертається основній програмі. Завдяки такому дизайну підвищується ефективність Spark. Наприклад, якщо великий файл був перетворений різними способами і переданий першій дії, то Spark обробить і поверне результат лише для першого рядка, а не стане опрацьовувати таким чином весь файл. За замовчуванням кожен трансформований RDD може переобчислюють щоразу, коли ви виконуєте над ним нову дію. Однак RDD також можна довготривало зберігати в пам'яті, використовуючи для цього метод зберігання або кешування; в такому випадку Spark буде тримати потрібні елементи на кластері, і ви зможете запитувати їх набагато швидше.Spark SQL – одна з чотирьох бібліотек фреймворка для структурованої обробки даних. Вона використовує структуру даних, так звану DataFrames і може виступати в ролі розподіленого механізму запитів SQL. Це дозволяє виконувати запити Hadoop Hive до 100 разів швидше.

Spark Streaming – простий у використанні інструмент для обробки потокових даних. Незважаючи на назву, Spark Streaming не обробляє жодних даних в реальному часі, а робить це в режимі micro-batch. Творці Spark стверджують, що продуктивність від цього страждає несильно, оскільки мінімальний час обробки кожного micro-batch 0,5 секунди. Бібліотека дозволяє використовувати код додатків batch-аналізу для потокової аналітики, що полегшує реалізацію λ-архітектури.

Spark Streaming отримує вхідні потоки даних і розбиває дані на пакети. Далі вони обробляються движком Spark, після чого генерується кінцевий потік даних (також в пакетної формі) як показано нижче.

API Spark Streaming точно відповідає API Spark Core, тому програмісти без складностей можуть одночасно працювати і з пакетними, і з потоковими даними, як видно на рис. 2.9

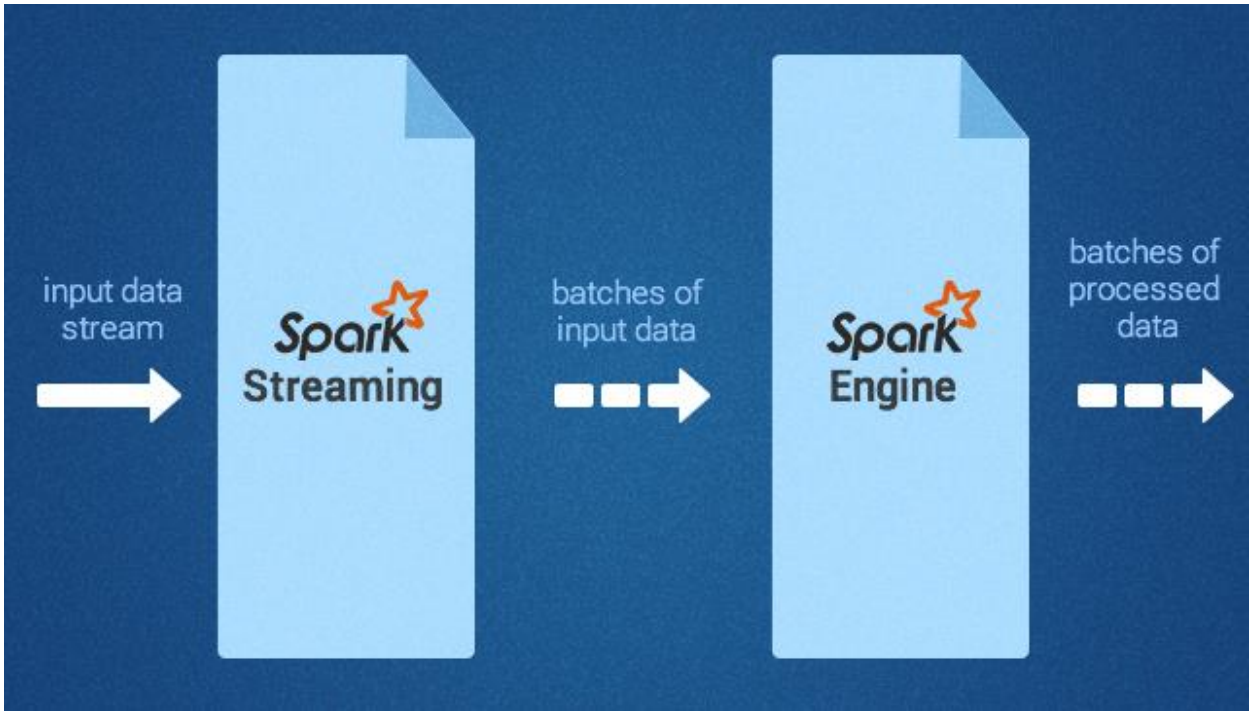


Рисунок 2.9 – Процесинг в Spark

Spark Streaming легко інтегрується з широким спектром популярних джерел даних: HDFS, Flume, Kafka, ZeroMQ, Kinesis і Twitter.

MLlib – це розподілена система машинного навчання з високою швидкістю. Вона в 9 швидше свого конкурента – бібліотеки Apache Mahout при тестуванні бенчмарками на алгоритмі чергуються найменших квадратів (ALS). MLlib включає в себе популярні алгоритми: класифікація, регресія, дерева прийняття рішень, рекомендація, кластеризація, тематичне моделювання.

GraphX – це бібліотека для масштабованої обробки графових даних. GraphX не підходить для графів, які змінюються транзакційних методом: наприклад, бази даних.

Spark працює:

1. в середовищі кластерів Hadoop на YARN;
2. під керуванням Mesos;

3. в хмарі на AWS або інших хмарних сервісах;
4. повністю автономно.

Він же підтримує кілька розподілених систем зберігання: HDFS, OpenStack Swift, NoSQL-СУБД, Cassandra, Amazon S3, Kudu, MapR-FS.

У 2009 році група аспірантів з Каліфорнійського університету Берклі розробила систему управління кластером з відкритим вихідним кодом – Mesos. Щоб показати всі можливості свого продукту і те, як легко управляти фреймворком на базі Mesos, ця група розпочала роботу над Spark. За задумом творців, Spark повинен був стати не просто альтернативою Hadoop, а й перевершити його. Основна відмінність двох фреймворків – спосіб звернення до даних. Hadoop зберігає дані на жорсткий диск на кожному кроці алгоритму MapReduce, а Spark робить всі операції в оперативній пам'яті. Завдяки цьому Spark може запускати програми до 100х швидше, ніж Hadoop MapReduce в пам'яті або 10 разів швидше на диску. У 2010 році проект був опублікований під ліцензією BSD, а в 2013 році перейшов під ліцензію Apache Software Foundation, який спонсорує і розробляє перспективні проекти. Mesos також привернув увагу Apache і перейшов під його ліцензію, але не став таким же популярним, як Spark.

Крім того, код на Spark пишеться швидше, оскільки тут у вашому розпорядженні буде більше 80 високорівневих операторів.

При вивченні Apache Spark варто відзначити ще один важливий аспект: тут надається готова інтерактивна оболонка (REPL). За допомогою REPL можна протестувати результат виконання кожного рядка коду без необхідності спочатку програмувати і виконувати всі завдання цілком. Тому написати готовий код вдається набагато швидше, крім того, забезпечується ситуативний аналіз даних.

Крім того, Spark має наступні ключові риси:

- в даний час надає API для Scala, Java і Python, також готується підтримка інших мов (наприклад, R);
- добре інтегрується з екосистемою Hadoop і джерелами даних (HDFS, Amazon S3, Hive, HBase, Cassandra , etc.);

– може працювати на кластерах під керуванням Hadoop YARN або Apache Mesos, а також працювати в автономному режимі.

Висновки до розділу 2

Таким чином, було наведено приклади, де у сучасному світі використовуються великі дані та також як всі процеси на нашій планеті між собою взаємопов'язані. Це було доцільно проаналізувати, так як робота з великими даними особливо у соціальних мережах, де весь час спілкуються люди є дуже важливою, тому що там кожну секунду великі данні відіграють велику роль, також щоб у майбутньому мати змогу відштовхуватись від вже створених методів та підходів роботи з Big Data.

Програми обробки Big Data не дозволяють контролювати етапи введення даних, збирати статистику і підбирати оптимальні структури для зберігання індексів, оптимізувати розміщення даних на диску для забезпечення високої швидкості введення / виводу, для виконання аналітичних запитів немає можливості провести глибокий статистичний аналіз і виробити оптимальний план виконання. Найважливіше значення для масштабованості і швидкодії додатків мають характеристики мереж ЦОД – оптимізація якості обслуговування (QoS) вимагає активного обміну інформацією між обчислювальними вузлами. Однак більшість нинішніх ЦОД не здатні забезпечити високі швидкості переносу даних, зіставні з показниками комунікаційних мереж високопродуктивних комп'ютерів.

Що стосується методів аналізу для обробки Big Data, існуючі на сьогодні інструменти і найбільш поширені методи аналізу масивів даних поки не повністю задовольняють вимогам додатків обробки Big Data. В одному випадку вони не придатні для обробки Big Data , в іншому – важко їх застосовність при побудові автоматичної класифікації безлічі об'єктів в умовах відсутності апріорної інформації про кількість класів, в третьому – алгоритм має високу трудомісткість.

Щодо підходів до обробки BigData, виділено два основних, потокова обробка та пакетна обробка. Саме ці подходи й будуть порівняні й досліджені в даній роботі.

Також був проведений аналіз існуючих програмних засобів що дозволяють та допомогають обробляти великі обсяги текстової і не тільки інформації. Спочатку була зібрана статистика з використання мов програмуввання для BigData. Java є

найвикористовуванішою мовою згідно кількості проектів на github.com. Тому був проведений аналіз кількості використання існуючих фреймворків для Java.

Найбільш використовуваними є Apache Spark серед інструментів потокової обробки та Apache Hadoop серед інструментів потокової обробки даних, саме вони й були обрані для порівняння.

3. АНАЛІЗ АЛГОРИТМІВ ОБРОБКИ ПОВІДОМЛЕНЬ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ПОШУКУ ТРЕНДІВ У ПОВІДОМЛЕННЯХ

3.1 Типові задачі обробки природної мови

Люди спілкуються за допомогою різних форм мови і використовують текст або мовлення. Щоб комп'ютери могли взаємодіяти з людьми, вони повинні розуміти природну мову, якою люди розмовляють. Обробка природної мови – просцесс під час якого комп'ютери розуміють, оброблють та використовують природні мови.

Обробка природної мови (далі NLP) - область, що знаходяться на перетині інформатики, мистецького інтелекту та лінгвістики. Термін NLP вкладається в обробку та "розуміння" натуральної мови для перекладу тексту та відповіді на питання.

З розвитком голосових інтерфейсів та чат-ботів, НЛП стала однією з найважливіших технологій мистецького інтелекту. Нове повне розуміння та виведення словесної мови - надзвичайно сложна задача, так як людський мова має особливості:

Натуральна мова – спеціально сконструйована система передачі змісту сказаного або написаного. Це не просто екзогенний сигнал, а особиста передача інформації. Крім того, мова кодирується так, що навіть маленькі дети можуть швидко вивести його.

Натуральна мова -– дискретна, символічна або категоріальна сигнальна система, що володіє надійністю.

Категоріальні символи мови кодуються як сигнали для спільності за кількома каналами: звук, жести, письмо, зображення та так далі. При цьому мова здатна виражатися будь-яким способом.

Де застосовується NLP

Сьогодні швидко зростає кількість полезних додатків у цій галузі:

- пошук (письмовий або устний);
- показ відповідної інтернет-реклами;
- автоматичний (або при содействії) перевод;

- аналіз настрою для задач маркетингу;
- розпізнавання речі та чат-боти,
- голосові довідники (автоматизована допомога покупцелю, замовлення товарів та послуг).

NLP – один з напрямків штучного інтелекту, який працює з аналізом, розумінням і генерацією живих мов, для того, щоб взаємодіяти з комп'ютерами і усно, і письмово, використовуючи природні мови замість комп'ютерних.

Для поставлених задач будуть використані такі алгоритми аналызу живої мови як токенізація, лематизація, пошук n-грамм.

3.2 Токенізація

Для обробки письмової природної мови необхідно розбити текст на більш дрібні одиниці, які називаються токенами. Для подальшоъ обробки тексту комп’ютером спочатку він повинний бути поділений на окремі об'єкти – токени. Токени це зазвичай прості слова, які є найменшими незалежними одиницями людської мови. Токенізація може бути виконана по реченням та по словам. Оскільки Повідомлення в мережі твіттер є короткими, буде викоримтана токенізація по словам.

На сьогоднішній день існує безліч способів токенізувати текст, щоб розбити електронний текст на окремо значущі одиниці для подальшої комп'ютерної обробки цих одиниць. Існують різні програми-токенизатори, наприклад, Stanford Tokenizer або спеціальні бібліотеки, наприклад, NLTK. Зачастую у розробниках лінгвістичних корпусів, що відповідають певним вимогам до токенізації тексту, які не можуть задовольнити існуючі програми та бібліотеки. У цій статті ми розглянемо найбільш гнучкий інструмент для токенізації тексту -— регулярні висловлювання, покажемо переваги цього інструменту.

Простий токенізатор слів може бути реалізований на багатьох мовах. У даному додатку буде використаний Stanford tokenizer. Розглянемо приклад роботи з фразою “the best fox is running”(рис. 3.1):

Рисунок 3.1 – Результат роботи токенізатора

Використовуючи маркери, можна створити так звані n-грами, які вказують набір токенів з довжиною n. «Грама» – це грецьке слово, що позначає букву або знак. Коли мова йде про набір з n букв в словах, мова йде про символи n грамів.

3.3 Лематизація

Для кращого розуміння лематизації спочатку розглянемо базовий типовий алгоритм NLP – стемінг.

Кількість коректних словоформ, значення яких схожі, але написання відрізняються суфіксами, приставками, закінченнями і іншим, дуже велике, що ускладнює створення словників і подальшу обробку. Стемінг дозволяє привести слово до його основній формі. Суть підходу в знаходженні основи слова, для цього з кінця і початку слова послідовно відрізаються його частини. Правила відсікання для Стеммер створюються заздалегідь, і найчастіше являють собою регулярні вирази, що робить даний підхід трудомістким, так як при підключенні чергового мови потрібні нові лінгвістичні дослідження. Другим недоліком підходу є можлива втрата інформації при відрізанні частин, наприклад, ми можемо втратити інформацію про частини мови.

І стеммінг і лематизація приводять слово до його початкової форми, але з однією відмінністю: в даному випадку корінь слова буде існуючим в мові словом. Наприклад, слово "riding" перетвориться в "ride", а не "rid", як в стеммізації. Оскільки лематизація використовує словники, вона може зіставити «best» з леммою «good» , що можна побачити на рис. 3.2

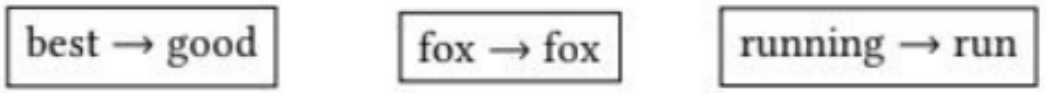


Рисунок 3.2 – Приклад роботи лематизації

3.3 Пошук n-грам

Для підвищення якості пошуку слід застосовувати нові методи добування інформації, щоб забезпечити належний аналіз в документації по програмному забезпеченню. Для цієї мети підходять методи обробки природної мови. Одним з таких методів є модель N-грам. Він був покладений в основу алгоритму для пошуку неточних повторів в інструменті Duplicate Finder.

Текст розглядається як набір пропозицій. Для кожної пропозиції модель N-грам включає в себе всі послідовності (N-грамми), що складаються з n слів, де кожне слово безпосередньо відповідає слову в початковому реченні. Тому кожен N-грам є підрядком відповідної пропозиції. Наприклад, якщо ми хочемо виявити той факт, що дві пропозиції схожі, ми порівнюємо їх безлічі N-грам.

N-грами можуть застосовуватися в широкій області наук: теоретичної математики, біології, картографії, музиці, генетиці, а також для кластеризації серії супутникових знімків Землі з космосу, в комп'ютерному стисненні, для індексування даних в пошукових системах.

Цей підхід є відносно простим, але ефективним, допускає орфографічні і граматичні помилки і не вимагає складної попередньої обробки вхідного тексту. Іноді необхідна тільки базова фільтрація: видалення пробілів і / або розділових знаків.

3.4 Використання алгоритмів NLP для обробки тексту та пошуку найпопулярніших словосполучень

Для заданих задач обробки повідомлень з соціальних мереж були обрані наступні алгоритми:

- токенізація
- лематизація

- пошук n-грам.

Спочастку отримані повідомленя будуть фільтровані на рахунок слів сполучників. Потім текст буде поділений на токени, і будуть знайдені всі n-грами. Серед знайдених n-грам будуть знайдені найпопулярніші словосполучення серед твітів користувачів соціальної мережі.

Обрані алгоритми є одними з типових для обробки природної мови, тому добре підходять для тестування ефективності програмних засобів.

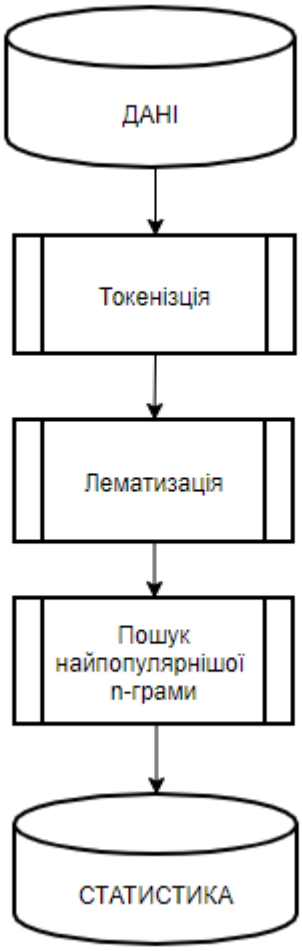


Рисунок 3.3 – Блок схема обробки даних

Розглянемо на прикладі речення “I am writing my master's thesis” алгоритм вище.

Першим етапом є токенізація. Під час токеніцаціі програмний засіб розділяє речення на об'єкти. Після першого етапи маємо такі вихідні дані, що можна побачити на рис. 3.4

Рисунок 3.4 – Результат токенізації

Другим етапом є лемматизація. В даному етапі кожен об'єкт приведений до нормальної форми. Результатом є Json леммами


```
{
  "lemma": {
    "I": 1,
    "be": 1,
    "writing": 1,
    "thesis": 1,
    "master": 1
  }
}
```


Третім етапом є пошук n-грам, та виявлення найбільш вживаних. Результатом є таблиця з ними та частотою вживання в реченні.

На рис 3.5 можна побачити результат обробки заданого речення програмним додатком.

No.	N-gram	Frequency
1	i be	1
2	be writing	1
3	writing master	1
4	master thesis	1

Рисунок 3.5 – Результат пошуку n-грам

Висновки по розділу 3

У даному розділі були розглянуті алгоритми обробки природної мови, такі як токенізація, лематизація, пошук n-грам. Саме ці алгоритми й використані для поставленої задачі – обробки повідомлень соціальних мереж.

Ці алгоритми є типовими, тому добре підходять для аналізу продуктивності програмних засобів.

Задачі обробки природної мови з кожним роком стають більш актуальними, а в поєднанні з інструментами BigData вони є ще більш ефективними для досягнення нових цілей у науці, бізнесі тощо.

4. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ТЕХНЛОГІЇ BIGDATA ДЛЯ СТРУКТУРИЗАЦІЇ ІНФОРМАЦІЇ ІЗ СОЦІАЛЬНИХ МЕРЕЖ ТА ПОШУК ТРЕНДІВ

4.1 Середовище тестування

4.1.1. Хмарне середовище для Big Data

Хмарні обчислення - це надання обчислювальних служб (в тому числі серверів, сховища, баз даних, мереж, програмного забезпечення, аналітики та інтелектуального аналізу) через Інтернет ("хмара"). Такі служби прискорюють впровадження інновацій, підвищують гнучкість ресурсів і забезпечують економію завдяки високій масштабованості. Ви зазвичай платите тільки за хмарні служби, які дозволяють скоротити експлуатаційні витрати, а також підвищити ефективність управління інфраструктурою і масштабування в міру зміни потреб бізнесу.

Найбільш зручним та розвпосюдженним шляхом виконання розподілених обчислень є хмарні обчислення. Основними перевагами є:

- зниження вимог до обчислювальної потужності ПК (неодмінною умовою є тільки наявність доступу в інтернет);
- відмовостійкість;
- безпеку;
- висока швидкість обробки даних;
- зниження витрат на апаратне і програмне забезпечення, на обслуговування і електроенергію;
- економія дискового простору (і дані, і програми зберігаються в інтернеті).

Саме тому для виконання експерименті в даній роботі була обрана саме концепція хмарних обчислень.

Згідно з дослідженням ресурсу https://www.srgresearch.com/ компанія Amazon є лідером у галузі забезпечення хмарними сервісами. На рисунку 4.1 можна побачити результати дослідження даного ресурсу.

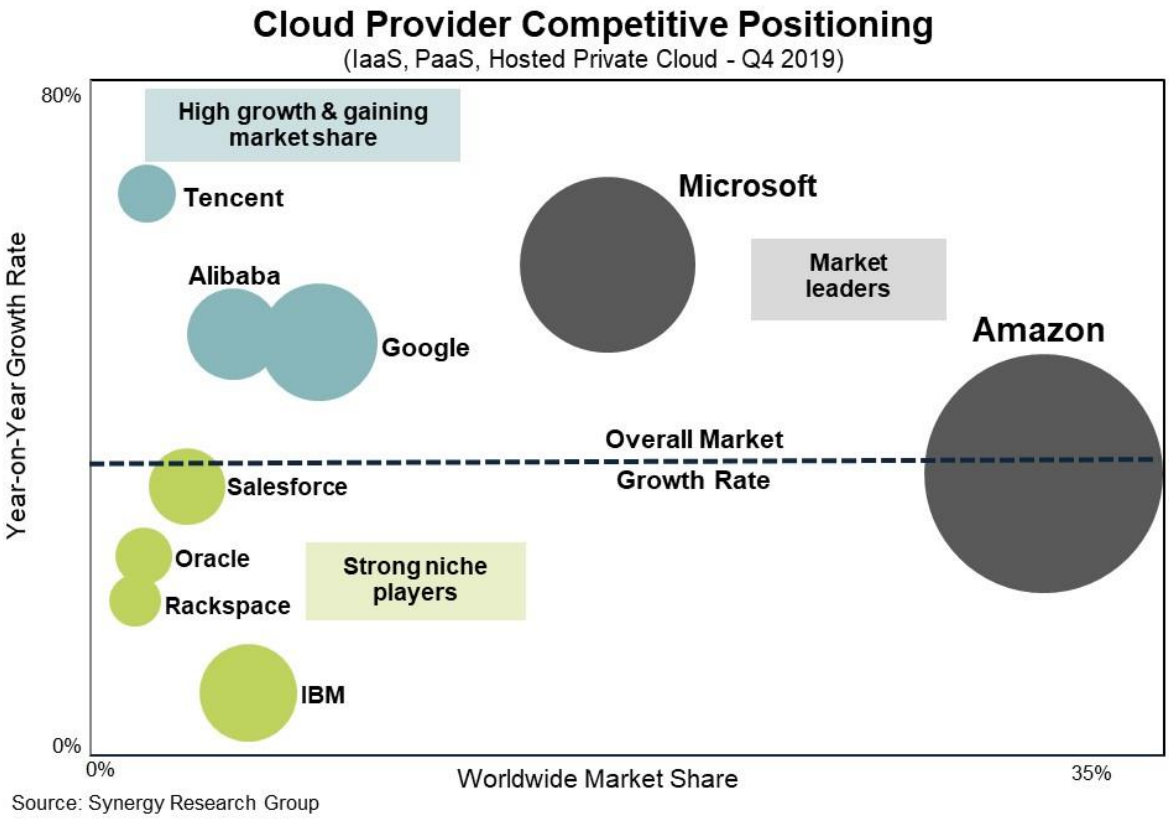


Рисунок 4.1 – Результати дослідження Synergy Research Group

4.1.2. Конфігурація кластерів

Hadoop

У кластері Hadoop є такі види служб:

1. NameNode – основний вузол, який веде облік кожного блоку кожного файлу, через який здійснюються всі операції, при відключенні якого стає недоступним весь HDFS;

2. SecondaryNameNode – додатковий вузол, який веде облік змін (на ділі не дає відмовостійкості, тому що не замінює собою NameNode);

3. DataNode – основні вузли кластера Hadoop, що зберігають дані, які очікують команд від головного вузла Name Node;

4. NodeManager – вузол системи YARN, що знаходиться в парі з кожним вузлом DataNode, що виконує контроль над виконанням завдань з даними, що зберігаються на вузлі;

5.ResourceManager – основний вузол системи YARN, що запускає обчислювальні завдання.

Для створення і налаштування кластера необхідно виконати наступну послідовність дій:

1. Розподіл служб Hadoop по вузлах;
2. Установка віртуалізації LXC;
3. Створення контейнера;
4. Установка і налаштування Hadoop;
5. Клонування контейнера;
6. Запуск кластера

4.2. Виконання експериментів

На рисунку 4.2 зображена схема кластера Hadoop що був використаний для проведення експериментів в рамках даної роботи.

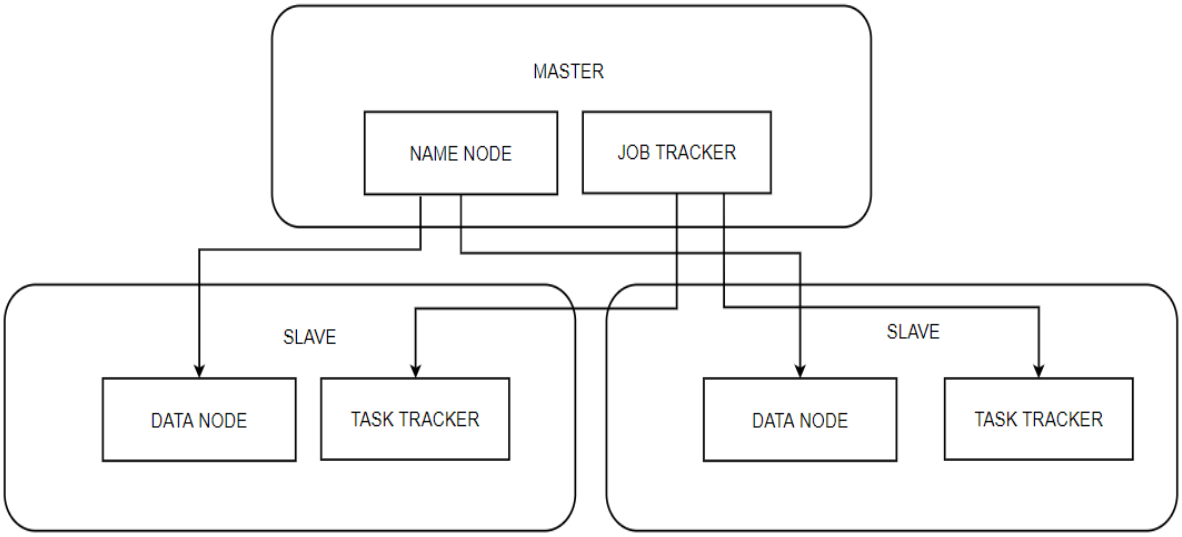


Рисунок 4.2 – Схема кластера Hadoop

Spark

Spark запускається як масив незалежних процесів на кластері, координацією яких займається об'єкт SparkContext у вашій програмі (називається вона driver program).

У режимі кластера, SparkContext може працювати в парі з одним з декількох менеджерів кластера (власний менеджер менеджер кластера, Mesos або YARN), який виділяє ресурси для додатків. Одного разу приєднавшись, Spark отримує виконавців (executors) на ноді в кластері, які обробляють обчислення і зберігають дані для вашої

програми. У підсумку, SparkContext відсилає завдання (tasks) до виконавців для обробки.

На рисунку 4.3 зображена схема кластера Hadoop що був використаний для проведення експериментів в рамках даної роботи.

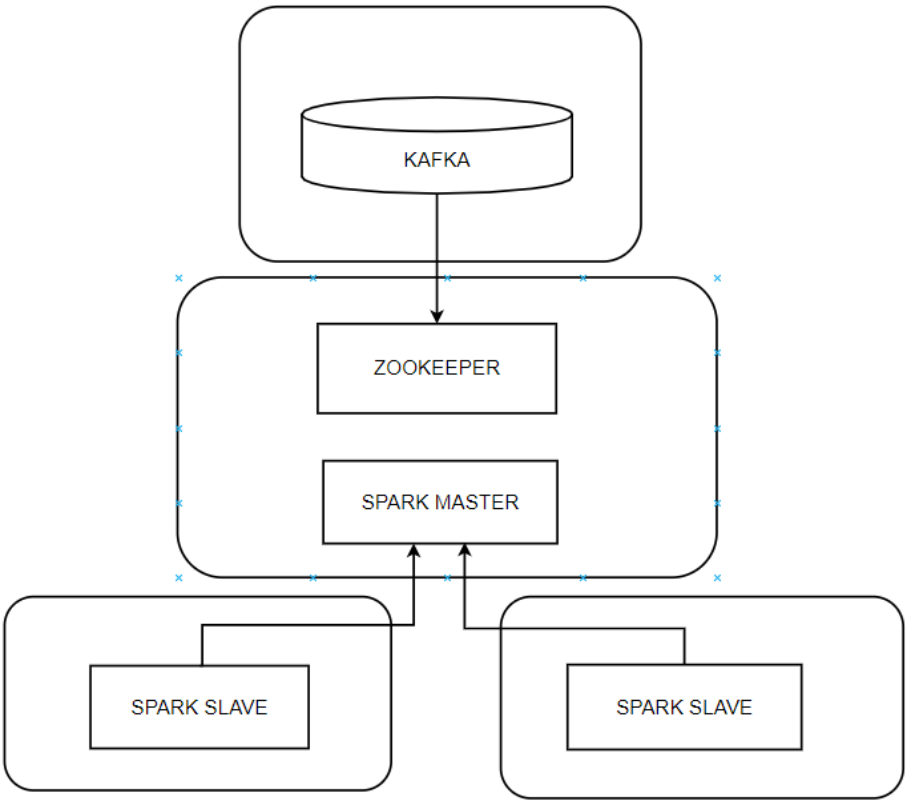


Рисунок 4.3 – Схема кластера Spark

Для виконання експериментів використовуєсься база даних з текстовими даними, тому для імітації потококого надходження потрібно використати чергу повідомлень. В даному випадку був використаний програмний продукт Apache Kafka. На рис. 4.4 можна побачити схему імітації надходження потокових даних.

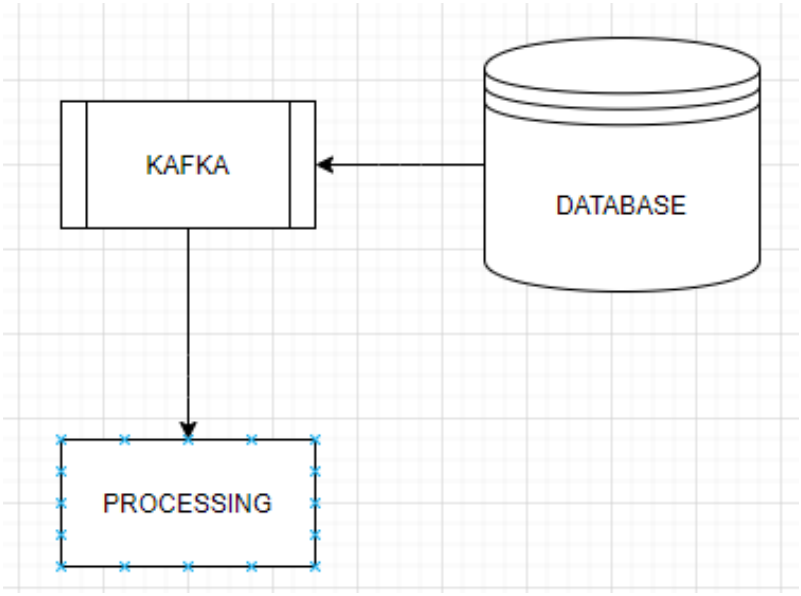


Рисунок 4.4 – Використання Apache Kafka

Для частини експериментив з обробки пакетами буда використана та сама база даних, проте дані були використані напряму, без брокера повідомлень, що можна побачити на рис. 4.5.

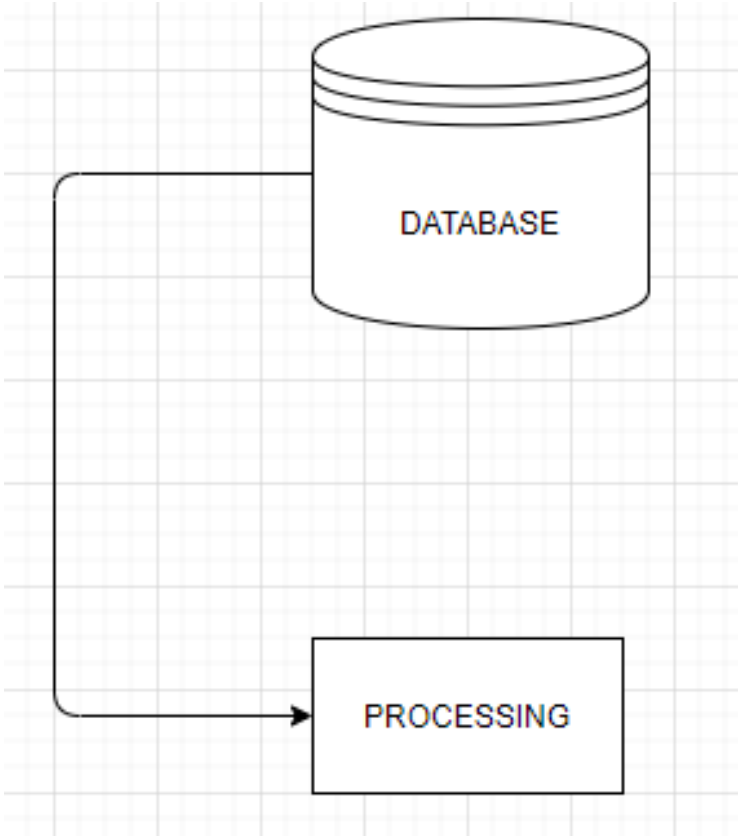


Рисунок 4.5 – Використання Apache Kafka

Тож, з конфігурацій двух кластерів можна зробити висновок що модель Master-Slave використовується в кластерах обох програмних продуктів.

4.2 Виконання експериментів

4.2.1 Планування експериментів

Планування експерименту – це процедура вибору числа і умов проведення дослідів, необхідних і достатніх для вирішення поставленого завдання з необхідною точністю.

Тут істотно наступне:

- прагнення до мінімізації загального числа дослідів;
- одночасне варіювання всіма змінними, що визначають процес, по
- спеціальними правилами - алгоритмами;
- використання математичного апарату, формалізує багато дій;
- вибір чіткої стратегії, що дозволяє приймати обгрунтовані рішення після кожної серії експериментів.

Завдання, для вирішення яких може використовуватися планування експерименту, надзвичайно різноманітні. До них відносяться: пошук оптимальних умов, побудова інтерполяційних формул, вибір істотних чинників, оцінка та уточнення констант теоретичних моделей, вибір найбільш прийнятних з деякого безлічі гіпотез про механізм явищ, дослідження діаграм склад - властивість і т.д.

Суть експериментів виявити кількісні показники швидкості обробки текстових даних використовуючи задані алгоритми обробки природної мови для двох програмних засобів — Apache Hadoop та Apache Spark.

Під час виконання роботи планування експериментів було поділено на 5 етапів:

- встановлення цілі експеримента
- уточнення умов проведення експериментів
- вибір вхідних та вихідних параметрів
- встанослення потрібної точності експериментів
- складання плану і встановлення потрібної кількості експериментів

Ціль експеримента

Основними цілями проведених експериментів були виявлення кращого програмного засоба, за такими характеристиками як продуктивність. Тобто експерименти за своєю природю порівняльні.

Умови проведення

Умовами проведення експериментів є використання однакових конфігурацій хмарних обчислень, в даному випадку це кластери з використанням машин t2.micro і t2.small, що мають однакові процессори та 1 та 2 ГБ оперативної памяті відповідно. Кожна з конфігурацій має один master і 2 slave ноди.

Вхідні дані

Для обох експериментів вхідними параметрами для виконання обробки тексту буде тестова база що містить 25 мільйонів повідомлень користувачів соціальних мереж. Вихідними параметрами є список n кількості найпопулярніших n-грамм. Часовий проміжок виконання експериментів 2,5 години. Згідно ресурсу hootsuite.com саме цей період є середнім для відвідування середньостатистичного користувача сої мережі твіттер. Щодо пакетної обробки, загальний час обробки повідомлень також буде 2,5 години, але виконання обробки виконується пакетами даних що були зібрані за періоди 5, 10, 15, 30, 90, 120 хв. щоб можливо було оцінити залежність продуктивності інструменту від розміру пакетів.

Встанослення потрібної точності експериментів

Оскільки ціллю даного дослідження є порівняльна характеристика то достатня кількість у 10 експериментів, щоб отримати середню кількість та виключити аномалії. Точність до 10 повідомлень за хвилину достатня для порівняння.

При обробці результатів вимірювань в науці і техніці зазвичай припускають нормальний закон розподілу випадкових похибок вимірювань. Воно завжди проявляється тоді, коли сумарна похибка є результатом неврахованого спільного впливу безлічі причин, кожна з яких дає малий внесок у похибка. Причому зовсім неважливо, за яким законом розподілений кожен з вкладів в окремо. Тому при виконанні даних експериментів також було припущено що похибки будуть розподілені за нормальним законом.

Завершенням обробки даних багаторазового прямого вимірювання при заданій ймовірності є два числа: середнє значення виміряної величини і його похибка. Обидва числа є остаточним результат багаторазового вимірювання і повинні бути спільно записані в стандартній формі $x = x \pm \Delta x$ яка містить тільки достовірні, тобто надійно виміряні, цифри цих чисел. Рещультами вимірювань будуть дані, на основі яких й будуть виявлені ці величини та будуть використані у порівняльній характеристиці.

Попередньо було виявлено що порядок виконання експериментів не важливий. Тому вони будуь виконані у такому порядку:

1. Обробка повідомлень у Spark з t2.micro за 2.5 години
2. Обробка повідомлень у Spark з t2.medium за 2.5 години
3. Обробка повідомлень у Spark з t2.micro за 2.5 години пакетами
   3.1.Період збору даних 5 хв.
   3.2.Період збору даних 10 хв.
   3.3.Період збору даних 15 хв.
   3.4.Період збору даних 30 хв.
   3.5.Період збору даних 60 хв.
   3.6.Період збору даних 90 хв.
   3.7.Період збору даних 150 хв.
4. Обробка повідомлень у Spark з t2.micro за 2.5 години пакетами
   4.1.Період збору даних 5 хв.
   4.2.Період збору даних 10 хв.
   4.3.Період збору даних 15 хв.
   4.4.Період збору даних 30 хв.
   4.5.Період збору даних 60 хв.
   4.6.Період збору даних 90 хв.
   4.7.Період збору даних 150 хв.

Результати будуть приведені у вигляді таблиць та графіків.

4.2.2 Виконання експериментів

Експерименти були виконані згідно плану що був встановлений в пункті 4.2.1.

Apache Spark

Згідно плану експериментів було виконано по 10 вимірів для кожного типу машин на платформи хмарних обчислень AWS, що були обрані для експериментів. У таблицях приведені перзультати вимірів, на яких можна побачити кількість оброблених повідомлень програмним додатком за 2.5. години. Також нижче представлена діаграма результатів, для візуалізації та можливості побачити аномальні значення.

Таблиця 2.1 – Результати вимірів

Номер експерименту	1	2	3	4	5
Кількість повідомлень t2.micro	13570855	13580753	13292890	13123368	13126645
Кількість повідомлень t2.small	20899117	20914359	20471051	20209986	23627961
Номер експерименту	6	7	8	9	10
Кількість повідомлень t2.micro	12888288	12874055	13124648	13968373	13274705
Кількість повідомлень t2.small	19847963	19826045	20211957	21511294	20443046

На рисунку 4.6 можна побачити результати виконання вимірів пунктів Apache Spark. Діаграма показує результати вимірів всіх 10 вимірів для кожного з типів машин.

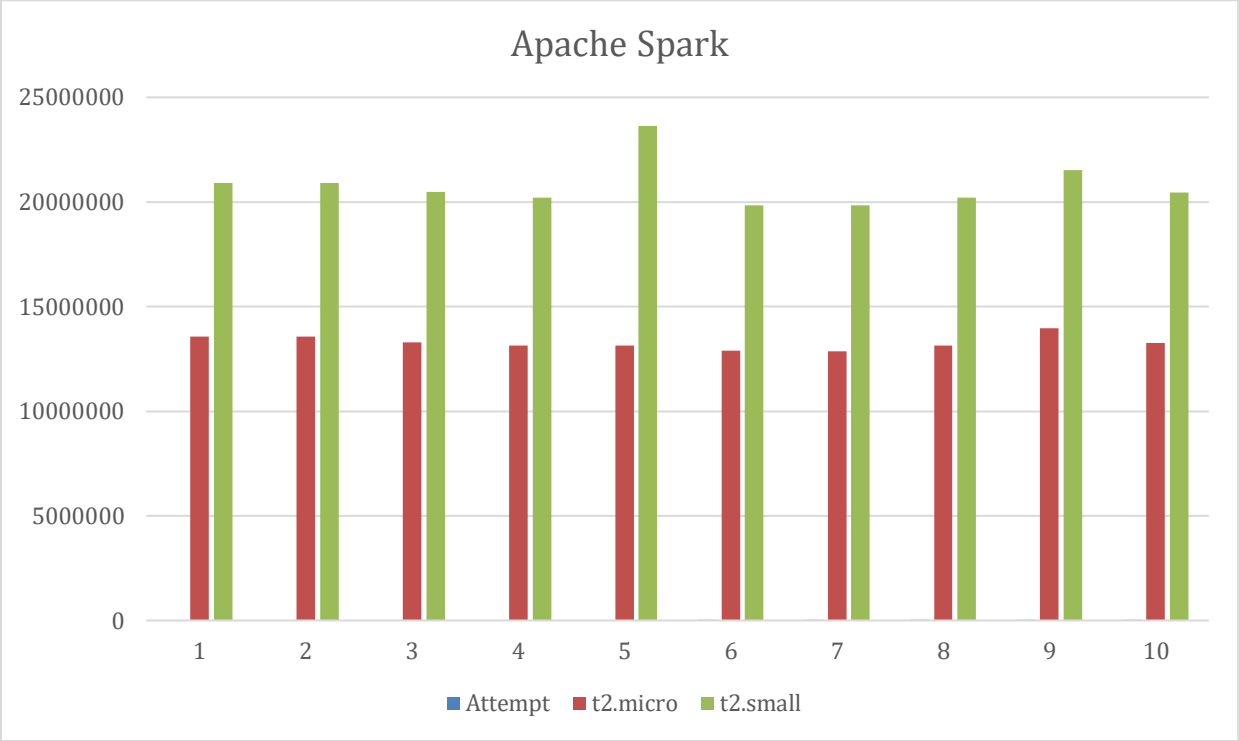


Рисунок 4.6 – Результати виконання вимірів пунктів Apache Spark.

Apache Hadoop

Згідно з планом експериментів виміри Apache Hadoop будуть проведені в декілька етапів з різним періодом збору даних, але сумарним часом 2,5 години.

Для кожного періоду проведено по три експерименти для виключення аномальних значень, якщо такі будуть присутні.

В рамках проведення даних експериментыв аномальними значеннями ваажаємо результати з відхиленям від середнього більше 10 відсотків. Такого показника достатньо щоб виключити ті результати що будуть не дійсними через причини нестабільності середовища, зв'язку, тощо.

Таблиця 2.2 – Результати вимірів з періодом 5 хвилин

Номер експеримента	1	2	3
К-сть повідомлень t2.micro	5803452	5803768	5803011
К-сть повідомлень t2.small	6847783	6848233	6848681

Таблиця 2.3 – Результати вимірів з періодом 15 хвилин

Номер експеримента	1	2	3
К-сть повідомлень t2.micro	6345923	6345994	6345751
К-сть повідомлень t2.small	7489164	7489189	7489668

Таблиця 2.4 – Результати вимірів з періодом 30 хвилин

Номер експеримента	1	2	3
К-сть повідомлень t2.micro	6834634	6834634	6834634
К-сть повідомлень t2.small	8064118	8064118	8064118

Таблиця 2.5 – Результати вимірів з періодом 60 хвилин

Номер експеримента	1	2	3
К-сть повідомлень t2.micro	7703257	7704862	7704911
К-сть повідомлень t2.small	9089543	9089643	9089183

Таблиця 2.6 – Результати вимірів з періодом 90 хвилин

Номер експеримента	1	2	3
К-сть повідомлень t2.micro	8506456	8506424	8506991
К-сть повідомлень t2.small	10037789	10037122	10037118

Таблиця 2.7 – Результати вимірів з періодом 150 хвилин

Номер експеримента	1	2	3
К-сть повідомлень t2.micro	914011	9122541	9123233
К-сть повідомлень t2.small	10765140	10765850	10765204

Нижче на рис 4.6 представлена діаграма з результатами вимірювання програмного додатку з Apache Hadoop для можливості візульного виявлення залежності продуктивності від розміру пакетів.

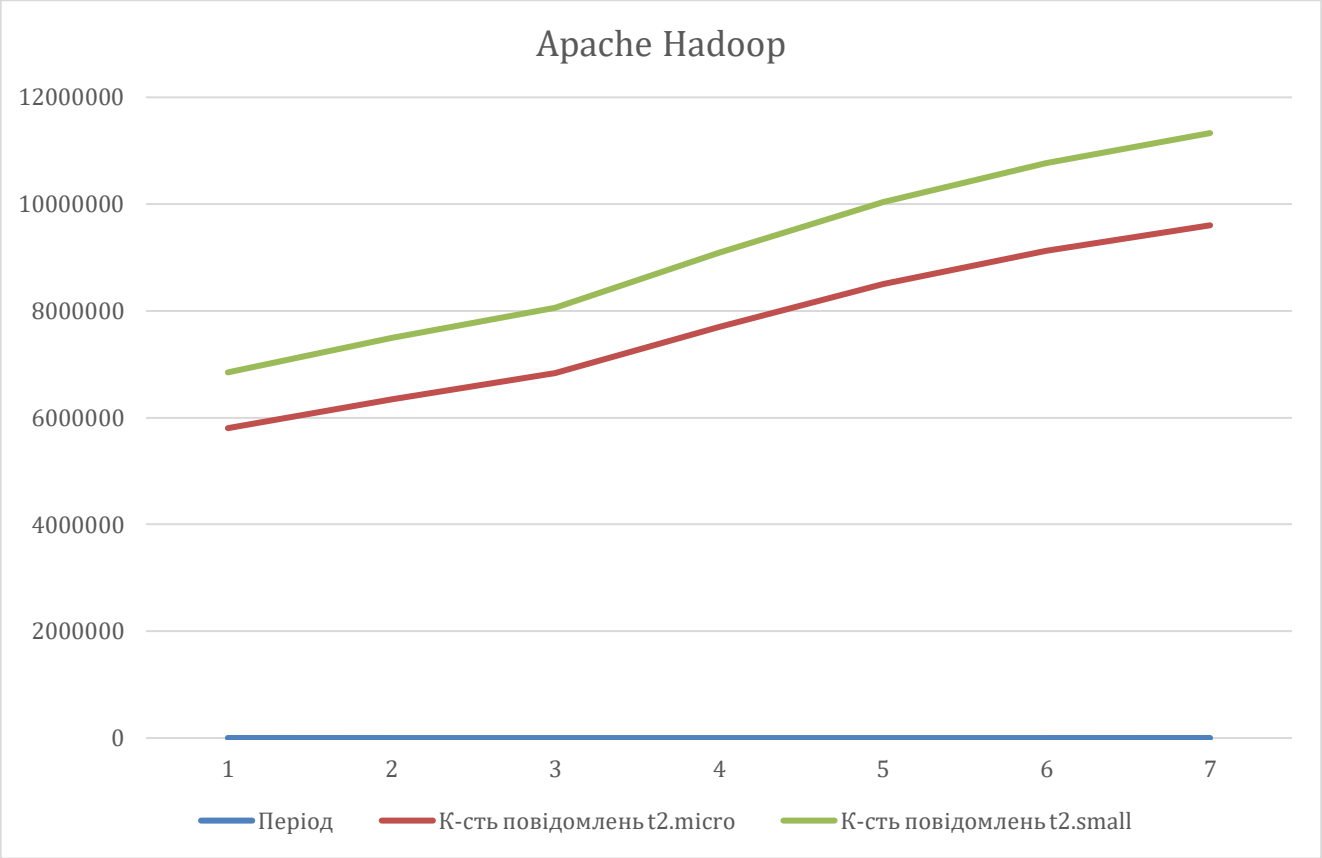


Рисунок 4.6 – Результати виконання вимірів пунктів Apache Spark.

4.3 Аналіз результатів та порівняльна характеристика

Згідно з результатами проведених експериментів для задач обробки невеликих повідомлень використовуючи типові алгоритми NLP більш ефективним виявився Apache Spark. В середньому за 2.5 години інструмент потокової обробки даних обробив на 40 відсотків більше повідомлень ніж інструмент пакетної обробки. Аномальних значень під час виконання дослідів не виявлено.

При дослідженні продуктивності Apache Hadoop було проведено досліди з різним періодом збору інформації, таким чином було виявлено що за рахунок збільшення розміру пакету, збільшується продуктивність даного програмного засобу.

Опираючись на результати експериментів можна сказати що Apache Spark має меньшу дисперсію результатів (до 6%) ніж Apache Hadoop (до 10%). Дані досліди не були спрямовані на зясування стабільності, але опираючись на результати можна сказати що для даної задачі засоб потокової обробки виявився більш стабільним.

Також експерименти були проведені з використанням двох типів апаратного забезпечення. Кластери були сконфігуровані з використанням хмарної платформи AWS, а саме EC2 машини типів t2.micro і t2.small.

Порівняльна характеристика програмних засобів для обробки повідомлень

В ході виконання роботи було виконане дослідження сучаних засобів та підходів обробки BigData для задач аналізу текстових даних з соціальних мереж. Були порівняні дві основні моделі – потокова обробка та обробка пакетами.

В якості програмного засобу для обробки пакетами був обраний Apache Hadoop, що реалізує MapReduce.

Для потокової обробки був використаних Apache Spark.

Найбільш ефективним засобом для даної задачі виявився Spark і потокова обробка даних.

Результати експериментів показали що стабільніше працює Hadoop. Через використання оперативної памяті Spark показав відхилення від середнього результату до 6%.

Під час проведення експериментів ні одна з систем не мала перебоїв. Hadoop має високу стійкість до перебоїв за рахунок файлової системи HDFS та реплікацій. Spark за рахунок розподілених датасетів RDD.

В рамках проведених експериментів було сконфігуровано по 2 кластери для кожної з тестуємих моделей обробки даних. Використана була найпопулярніша платформа Amazon Web Services.

В результаті експериментів було виявлено що за рахунок збільшення оперативної памяті в 2 рази Apache Spark став ефективніший в середньому на 55%, проти 18% у Hadoop.

За рахунок використання оперативної памяті потокова модель обробки даних дорожча.В рамках проведеного експерименту було виявлено що різниця у вартості кластеру AWS була ~20% за рахунок використання ресурсів.

Висновки по розділу 4

Насамперед моє дослідження присвячено існуючим методам для розв’язання проблеми, щоб мати змогу швидко використовувати прямі дані про сучасні тренди. Проаналізовано основні галузі, які використовувалися у роботі роботу з великою кількістю даних. Можна сказати що існуючі алгоритми та програми обробки Big Data усі базуються на різних платформах та дозволяють контролювати етапи введення даних, збирати статистику про користувачів соціальних мереж та підбирати для них доцільні пропозицій. В майбутньому даних стане ще більше, як само і різноманітних пристроїв, методологій та підходів для того щоб обробляти їх у найшвидший час.

Також проводився аналіз отриманих результатів, а саме порівняння продуктивності. У цьому розділі наведено моделі та методи, які використовуються у роботі. Наведено формалізований опис задачі

В ході роботи було проведено дослідження популярності програмних засобів для обробки великих даних, та виявлені основні підхои. В ході експериментів порівнювались Apache Spark, як представник потоковоъ обробки і Apache Hadoop як представник пакетної.

Найбільш ефективним засобом для даної задачі виявився Spark і потокова обробка даних.

Він має переваги і у швидкості обробки даних, і в ефективності використання ресурсів, і в стабільності.

Результати експериментів показали що стабільніше працює Hadoop. Через використання оперативної памяті Spark показав відхилення від середнього результату до 6%.

Під час проведення експериментів ні одна з систем не мала перебоїв. Hadoop має високу стійкість до перебоїв за рахунок файлової системи HDFS та реплікацій. Spark за рахунок розподілених датасетів RDD.

В рамках проведених експериментів було сконфігуровано по 2 кластери для кожної з тестуємих моделей обробки даних. Використана була найпопулярніша платформа Amazon Web Services.

В результаті експериментів було виявлено що за рахунок збільшення оперативної памяті в 2 рази Apache Spark став ефективніший в середньому на 55%, проти 18% у Hadoop.

Також важливим недоліком Hadoop є те що дані доставляються з затримкою. Оскільки у нас є якийсь період обчислень, то на цей період ми завжди відстаємо від реального часу. І чим більше ітерації, тим сильніше ми відстаємо. Таким чином, ми отримуємо затримку за часом, що в деяких випадках критично.

Через одночасовий великий обсяг даних створюється пікове навантаження на залізо. Якщо ми дуже багато обчислюємо в пакетному режимі, у нас після закінчення періоду (дня, тижня, місяця) спостерігається пік навантаження, адже порахувати потрібно багато всього. До чого це призводить? По-перше, починаємо упиратися в ліміти, які, як відомо, не нескінченні. В результаті система періодично працює на межі можливостей, що нерідко закінчується збоями. По-друге, так як всі ці job'и починаються одночасно, вони конкурують і розраховуються досить повільно, тобто на швидкий результат розраховувати не доводиться.

Однак і у потокової обробки є суттєві недоліки. Найважливіший пов'язаний з обробкою помилок. Він пов'язаний з відкатами. Якщо job'а не відпрацювала, і стався збій, то дуже важко вловити саме той момент, де все зламалося. І рішення проблеми потребують більше зусиль і ресурсів у порівнянні з пакетною обробкою.

Тож основним висновком є те що для обробки статичних текстових даних, великими пакетами є більш привабливим Apache Hadoop за рахунок невеликої вартості і достатньої ефективності в даному випадку. Але для обробки даних з соціальних мереж, які постійно оновлюються і додаються потокова обробка є більш ефективною разом з Apache Spark.

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Вимоги безпеки при виконанні робіт на робочому місці

За для забезбечення безпеки та попередження шкідливих впливів кожен працівник та роботодавець повинні дотримуватить відповідних положень.

Для розробки програмного додатку, що передбачений дослідження у даній магістерській роботі достатньо одного інженера-програміста, відповідно й розрахунки будуть вестись для одного робітника.

Вимоги щодо безпеки виконання робіт на рабочому місці описані в наступних законат та нормах України:

1. Закон України «Про охорону праці» прийнятий від 14 жовтня 1992 року № 2695-12 [41];
2. Типове положення про порядок проведення навчання і перевірки знань з питань охорони, НПАОП 0.00-4.12-05, початок дії від 14.04.2017 [42];
3. ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки, затверджено та введено в дію наказом міністерства економічного розвитку і торгівлі України 14.10.2013 № 1231[43];
4. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, ДСанПін 3.3.2.007-98. затверджено постановою головного санітарного лікаря України від 10 грудня 1998 року [44];
5. Державні санітарні норми виробничої загальної та локальної вібрації, ДСН 3.3.6.039-99, початок дії від 01.12.1999 [45];
6. ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення [46];
7. Державні будівельні норми України «Природне і штучне освітлення» ДБН В.2.5-28:2018, затверджені наказом Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 03.10.2018 № 264, введено в дії з 01.03.2019 [48];

8. ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища, введено в дію з 01.01.2006 [47];
9. ДСН 3.3.6.042-99 Державні санітарні норми мікроклімату виробничих приміщень, затверджені Постановою головного санітарного лікаря України № 42 від 1 грудня 1999 року [49];
10. Типове положення про службу охорони праці, затверджене наказом Державного комітету України з нагляду за охороною праці від 15.11.2004 № 255 і зареєстроване у Міністерстві юстиції України 01.12.2004 за № 1526/10125 [50];
11. Наказ про «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою», зареєстрований в Міністерстві юстиції України 7 липня 2014 р.за № 775/25552 [51];
12. НАПБ А.01.001-2014 Правила пожежної безпеки в Україні, затверджений наказом Міністерства внутрішніх справ України від 30.12.2014 № 1417 і зареєстрований у Міністерстві юстиції України 05.03.2015 за № 252/26697 [52] ;
13. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджені Постановою головного санітарного лікаря України № 42 від 1 грудня 1999 року [49];

5.2 Шкідливі виробничі фактори на робочому місці

Задля підтримання достойного рівня продуктивності працівника треба створити правильні умови праці. Робоче місце - це частина простору, в якому інженер здійснює трудову діяльність, і проводить велику частину робочого часу.

При правильній організації робочого місця продуктивність праці інженера зростає з 8 до 20 відсотків.

Основними шкідливими факторами є тривале сидяче положення, навантаження на очі, перевантаження суглобів рук та навіть стресс.

Тривале сидяче положення призводить до напруження м'язів шиї, голови, рук і плечей, остеохондрозу, у дітей - ще й до сколіозу. Тривале сидяче положення ще

призводить до застою крові в тазових органах і, як наслідок, до простатиту і геморою. Не секрет, що малорухливий спосіб життя призводить до ожиріння. Наслідки можуть бути найрізноманітнішими, від болів в спині і кінцівках, до паралічу кінцівок і смерті. Одна з поширених причин остеохондрозу - дистрофія м'язів спини. Людина, що веде в основному сидячий спосіб життя, цілком може захворіти на остеохондроз.

Навантаження на зір. Людське око реагує на найдрібнішу вібрацію тексту і на мерехтіння екрану. М'язи очі, керуючі кришталиком, знаходяться в постійній напрузі, що обов'язково призводить до втрати гостроти зору. Важливе значення для профілактики зорових дисфункцій надають: правильний або рекомендований підбір кольору, шрифтів, компоновки вікон у використовуваних додатках, орієнтація дисплея монітора.

Перевантаження суглобів кистей рук призводить головним чином до такого явища, як синдром зап'ястного каналу.

Тож для профілактики шкідливих впливів що описані вище потрібні регулярні фізичні активності та перерви у роботі. А для зменшення цих впливів робоче місце повинне бути правильним чином організоване.

Конструкція робочого місця і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця програміста повинні бути дотримані наступні основні умови:

- оптимальне розміщення устаткування, що входить до складу робочого місця;
- достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення;
- необхідно природне і штучне освітлення для виконання поставлених завдань;
- рівень акустичного шуму не повинен перевищувати допустимого значення.

5.2.1. Характеристика робочого місця

Згідно з даними у розділі 5.1. розрахунки будуть вестись для одного проацівника.

Згідно правил охорони праці під час експлуатації електронно-обчислювальних машин (НПАОП 0.00-1.31-99)[53] площа, виділена для одного робочого місця

з відеотерміналом або персональною ЕОМ, повинна складати не менше 6 кв.м, а обсяг - не менше 20 куб.м, а робочі місця відносно світлових
прорізів повинні розміщуватися так, щоб природне світло падало збоку, переважно зліва.

Робоче місце у якому розроблявся програмний додаток має такі характеристики:

1. Площа 12 кв.м.
2. Об'єм приміщення 34 куб.м.
3. Габарити робочого столу 1200x800x800 мм
4. Природне світло падає зліва

Тож можна стверждувати що робоче місце відповідає стандартам та підхоодить для розробки програмного забезпечення однією людиною.

5.2.2. Освітлення

Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини. У зв'язку з цим раціональне природне і штучне освітлення в житлових приміщеннях і громадських будівлях, на робочих місцях має велике значення для забезпечення нормальної життєдіяльності та працездатності людини. Світло не тільки забезпечує нормальну життєдіяльність організму людини, але і визначає життєвий тонус і ритм. Недостатнє освітлення робочого місця ускладнює тривалу роботу, викликає підвищене стомлення і сприяє розвитку короткозорості. Занадто низькі рівні освітленості викликають апатію і сонливість, а в деяких випадках сприяють розвитку почуття тривоги.

Освітленість (Е) визначається як світловий потік, який припадає на одиницю площі поверхні, що освітлюється. Одиниця виміру - люкс (лк), 1 лк - освітленість поверхні в 1 м2, на яку подає світловий потік в 1 лм.

Рівень штучного освітлення при використанні ламп розжарювання повинен складати 150 лк і 300 лк при люмінесцентних лампах.

На використаному робочому місці освітленність дорівнює 280 лк що є близько до норми.

5.2.3 Мікроклімат

Згідно з Державними санітарними нормами мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [45] температура повітря повинна становити 22–25°С, відносна вологість повітря — 40–60%, швидкість руху повітря — не більше 0,1 м/с.

Під час розробки програмного додатку ці показники відповідали нормам. Виміряна температура дорівнює 22 °С, вологість 54%, а швидкість руху повітря 0 м/с.

Для підтримання нормальних показників рекомендовано використовувати відповідну кліматичну техніку.

5.2.4 Шум та вібрація

Встановлено, що шум погіршує умови праці, погано впливаючи на організм людини. При тривалому впливі шуму на людину відбуваються небажані явища: знижується гострота зору, слуху, підвищується кров'яний тиск, знижується увага. Сильний тривалий шум може стати причиною функціональних змін серцево-судинної і нервової систем.

Для забезпечення дотримання допустимих рівнів шуму на робочих місцях застосовуються засоби звукопоглинання, вибір яких обґрунтовується спеціальними інженерно-акустичними розрахунками.

Під час розробки програмного додатку рівень шуму становив 20-30 дБ що є в межах норми згідно з Державними санітарними нормами виробничої загальної та локальної вібрації, ДСН 3.3.6.039-99 [46]

5.3 Дії працівників в надзвичайних ситуаціях

Під час роботи з електричною технікою, такою як комп'ютер, комп'ютерна перифирія, тощо інжерен програміст може отпримати електротравму.

Дія електричного струму на організм людини може викликати ураження, результат яких залежить від багатьох факторів. Небезпека впливу електричного струму на людину велика ще й тому, що він непомітний для ока, не чуємо, не відчувається на відстані, не має запаху, а сприймається лише в момент зіткнення з незахищеними струмопровідних проводів або деталями електроустановок та їх корпусами, які з яких- небудь причин потрапили під напругу.

Існує певний порядок надання першої допомоги ураженому, який зазначений в наказі Міністерства охорони здоров'я України «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою» [51]

Послідовність дій при наданні домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою не медичними працівниками:

1) переконатися у відсутності небезпеки;

2) якщо постраждалий перебуває під дією електричного струму, при можливості припинити його дію: вимкнути джерело струму, відкинути електричний провід за допомогою сухої дерев'яної палиці чи іншого електронепровідного засобу;

3) провести огляд постраждалого, визначити наявність свідомості, дихання;

4) викликати бригаду екстреної (швидкої) медичної допомоги;

5) якщо у постраждалого відсутнє дихання, розпочати проведення серцево-легеневої реанімації;

6) якщо постраждалий без свідомості, але дихання збережене, надати постраждалому стабільного положення;

7) накласти на місця опіку чисті, стерильні пов'язки;

8) забезпечити постійний нагляд за постраждалим до приїзду бригади екстреної (швидкої) медичної допомоги;

9) при погіршенні стану постраждалого до приїзду бригади екстреної (швидкої) медичної допомоги повторно зателефонувати диспетчеру екстреної медичної допомоги.

Що не можна робити при електротравмі:

1) Торкатися до потерпілого мокрими і не ізольованими руками і предметами, якщо джерело струму не відключений. Братися за одяг потерпілого, якщо вона мокра або не відділяється від тіла.
2) Залишати травмованого на самоті, навіть на хвилину.
3) Поїти хворого гарячими напоями, давати йому каву, алкоголь.
4) Відмовлятися від госпіталізації, якщо потерпілий відчуває себе відносно добре. Найчастіше ураження електричним струмом, навіть легке, дає

відстрочені ускладнення, тому важливо отримати кваліфіковане лікування і перебувати під наглядом медиків стільки, скільки потрібно.

Висновки до розділу 5

Був проведений аналіз вимог та рекомендацій до робочого місця працівника для покращення його стану, продуктивності та запобігання можливих шкідливих впливів.

Також проведений аналіз правил першої допомоги робітнику з електротравмою.

Тож для попередження шкідливих впливів, розвитку різноманітних хвороб та підвищення продуктивності роботодавцю та робітнику необхідно організовувати робочий простір та робочий процес згідно з нормами та законами, наведеними у цьому розділі.

ВИСНОВКИ

Таким чином, епоха Big Data вже настала - обсяги даних, що генеруються в науці, бізнесі, індустрії та управлінні IT, ростуть експоненціально. Однак існуючі програми обробки Big Data не дозволяють контролювати етапи введення даних, збирати статистику і підбирати оптимальні структури для зберігання індексів, оптимізувати розміщення даних на диску для забезпечення високої швидкості введення / виводу, для виконання аналітичних запитів немає можливості провести глибокий статистичний аналіз і виробити оптимальний план виконання.

Найважливіше значення для масштабованості і швидкодії додатків мають характеристики мереж ЦОД - оптимізація якості обслуговування (QoS) вимагає активного обміну інформацією між обчислювальними вузлами. Однак більшість нинішніх ЦОД не здатні забезпечити високі швидкості переносу даних, зіставні з показниками комунікаційних мереж високопродуктивних комп'ютерів.

Що стосується методів аналізу для обробки Big Data, існуючі на сьогодні інструменти і найбільш поширені методи аналізу масивів даних поки не повністю задовольняють вимогам додатків обробки Big Data. В одному випадку вони не придатні для обробки Big Data , в іншому – важко їх застосовність при побудові автоматичної класифікації безлічі об'єктів в умовах відсутності апріорної інформації про кількість класів, в третьому – алгоритм має високу трудомісткість.

В даний момент можна прогнозувати високошвидкісну доставку даних з розподілених джерел, оптимізацію перенесення даних можна здійснити, наприклад, за допомогою розвиваються зараз методів управління ресурсами з дотриманням гарантій якості обслуговування (QoS). У промисловій сфері можна прогнозувати апаратні засоби зі спеціалізованими датчиками для точного зняття показників даних, а також розвиток програм, які будуть ці дані збирати, обробляти і структурувати, передавати в центри обробки, візуалізувати для легкого сприйняття, що, в свою чергу, полегшить прийняття правильного рішення. В роботі розроблена паралельна реалізація алгоритму k-середніх для кластеризації данних використання послуг

телеком оператора на моделі Map Reduce. Також проведенні дослідження щодо можливостей системи обробляти данні різного об’єму та на різних кількостях вузлів.

У роботі було виконано дослідження особливостей використання технології BigData для структуризації інформації із соціальних мереж та пошуку нових трендів.

Реалізовано програмне середовище, здатне використовувати технології BigData для структуризації інформації із соціальних мереж та пошуку нових трендів.

У ході розробки проекту використовувалися новітні технології. Завдяки дотриманню принципів об’єктно-орієнтованого дизайну та широкому використанню шаблонів проектування, систему буде легко супроводжувати та вдосконалювати.

Під час аналізу предметної сфери було висвітлено основні проблеми використання технології BigData для структуризації інформації із соціальних мереж та пошуку нових трендів, виконано широкий огляд програмних аналогів та їх порівняння.

Були розглянуті алгоритми обробки природної мови, такі як токенізація, лематизація, пошук n-грам. Саме ці алгоритми й використані для поставленої задачі – обробки повідомлень соціальних мереж.

Задачі обробки природної мови з кожним роком стають більш актуальними, а в поєднанні з інструментами BigData вони є ще більш ефективними для досягнення нових цілей у науці, бізнесі тощо.

Проаналізовано технологічні платформи, наведено аргументацію використання обраних програмних рішень, зібрана статистика використання цих технологій.

Були проведені виміри продуктивності обраних програмних засобів потокової та пакетної обробок даних. Висновком є те що для обробки статичних текстових даних, великими пакетами є більш привабливим Apache Hadoop за рахунок невеликої вартості і достатньої ефективності в даному випадку. Але для обробки даних з соціальних мереж, які постійно оновлюються і додаються потокова обробка є більш ефективною разом з Apache Spark.

Проектування системи відбувалося з використанням об’єктно-орієнтованого підходу. Використання шаблонів проектування та окремі дизайнерські рішення розробника дозволили зробити систему гнучкою, що легко піддається модифікації, зрозумілою та надійною.

Використання сучасних гнучких методологій розробки дозволить і надалі розширювати функціональність системи або вносити зміни до її поточної функціональності.

Проектування та розробка програми проводилась з використанням сучасних технологій та підходів до вирішення цих питань, також була спроектована база даних серверної частини.

Після закінчення розробки, була написана вся необхідна програмна документація. Наявність якісного керівництва користувача дозволить користувачам, що будуть використовувати додаток, швидко вивчити можливості програми та почати її використовувати. Задокументований опис та текст програми спростить подальше супроводження системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Акатова Н.А. Комплексность проектирования интегрированных информационных систем административно-территориального и муниципального управления [Текст] / Акатова Н.А., Черкасов Ю.М. // Вестник Университета: серия «Информационные системы управления». –, 2000. – № 1. – С.23-35.
2. Большие данные [Електронний ресурс] Режим доступу: http://sewiki.ru/ index.php?title=Большие_данные&oldid=3075
3. Большие_данные (Big_Data) ? [Електронний ресурс] Режим доступу: http://www.tadviser.ru/index.php/Статья:Большие_данные (Big_Data)
4. Большие_данные [Електронний ресурс] Режим доступу: https://ru.wikipedia. org/wiki/Большие_данные
5. Большие_данные_(Big_Data) [Електронний ресурс] Режим доступу: http://www.tadviser.ru/index.php/Статья:Большие_данные_(Big_Data)
6. Бутакова М.А. Мера информационного подобия для анализа слабоструктурированной информации [Електронний ресурс] / Бутакова М.А., Климанская Е.В., Янц В.И. // Современные проблемы науки и образования. – 2013. – № 6. – Режим доступу: http://www.science-educa- tion.ru/113-11307.
7. Горчаков, А. Математичний апарат для інвестора. Аналіз та прогнозування часових рядів [Текст] / А. Горчаков // Фінансовий ринок України. – 2007. – № 12. – С. 38–42.
8. Згуровський М.З. Основи системного аналізу [Текст] / Згуровський М.З., Панкратова Н.Д. – К.: Видавнича група BHV, 2007. – 544 с.: іл.
9. Ильин И. В., Леонова О. Г., Розанов А. С. Теория и практика политической глобалистики. – М.: Издательство Московского Университета, 2013. 296 с.
10. Карелов С. Долой Шерлока Холмса. Как победить на рынке BIG DATA // Slon. URL: http://slon.ru/biz/1036183/.
11. Кісь Я.П. Інтелектуальні геоінформаційні системи. Міжнародний досвід та шляхи розвитку в Україні [Текст] / Я.П.Кісь, Н.Б.Шаховська, О.Б.Вальчук //

Вісник Національного університету «Львівська політехніка». –, 2008. –№653: Інформаційні системи та мережі. – С. 139–145.

12. Лычкина Н.Н. Компьютерное моделирование социальноэкономического развития регионов в системах поддержки принятия решений [Текст] / Н.Н. Лычкина // Материалы III Международной конференции «Идентификация систем и задачи управления» SICPRO04, 28-30 января 2004, Москва. – М.: ИПУ РАН, 2004. – С. 1377–1402.
13. Майер-Шенбергер В., Кукьер К. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим / Пер. с англ. Гайдюк И. – М.: Манн, Иванов и Фербер, 2014. 240 с.
14. Найдич А. Большие данные: насколько они большие? [Електронний ресурс] Режим доступу: http://compress.ru/article.aspx?id=23469
15. Свами М. Графы, сети и алгоритмы / М. Свами, К. Тхуласираман. –М.: Наука, 1984. – 256 с.
16. Смородин Г. Н. Рынок трудовых ресурсов 2020 // Академический форум корпорации EMC (EMC Academic Forum Russia & CIS). Сборник тезисов участников конференции – Москва, факультет ВМК МГУ им. М.В. Ломоносова. М.: МАКС Пресс, 2014 г. С. 3–5.
17. Технологии Big Data и их применение на современном промышленном предприятии [Електронний ресурс] Режим доступу: http://engjournal.ru /articles/1228/1228.pdf
18. Что такое Big Data? [Електронний ресурс] Режим доступу: https://postnauka. ru/faq/46974
19. Шенбергер В. М., Кукьер К. Большие данные. Революция, которая изменит то, как мы живем, работаем и мыслим. – М.: Манн, Иванов и Фербер, 2014.
20. Ширикov А. Big Data – самый надоевший термин 2013 года // Slon. URL: http://slon.ru/biz/1026116/.
21. Big Data [Електронний ресурс] Режим доступу: https://intellect.ml/big-data-6821
22. Big Data [Електронний ресурс]. Режим доступу: http://www.tadviser.r u/index.php/Статья:Большие_данные_%28Big_Data%29#cite_note-g-6 .

23. Big Data Brighten BI Future. eWeek [Electronic Resours]. – Access mode: http://www.eweek.com/c/a/Data-Storage/TBA-Hadoop-Yahoo-BigData-Brightens-BI-Future-254079.
24. Big data the next frontier for innovation [Електронний ресурс] Режим доступу: http://www.mckinsey.com/insights/business_technology/big_data_the_next_fro ntier_for_innovation
25. Big Data в промышленности: инновации, к которым придется привыкать [Електронний ресурс] Режим доступу: http://www.ogcs.com.ua/index.php/articles/121-big-data-v-promyshlennosti-innovatsii-k-kotorym-pridetsya-privykat
26. Bigtable: A Distributed Storage System for Structured Data [Electronic Resours] / Chang Fay, Dean Jeffrey, Ghemawat Sanjay, Hsieh Wilson C., Wallach, Deborah A., Burrows Michael, Chandra Tushar, Fikes Andrew, Gruber Robert E. Access mode: http://static.googleusercontent.com/media/r esearch.google.com/ru//archive/bigtable-osdi06.pdf.
27. Chernyak L. Big Data - the new theory and practice. Open the system [Electronic Resours] / Leonid Chernyak. – M. : Open Systems, 2011. – № 10. – Access mode: http://www.osp.ru/os/2011/ 10/13010990.
28. Exploiting technical terminology for knowledge management / Rinaldi F, Yuste E., Schneider G and others // Proceedings of ECAI and EKAW, 2004. – Режим доступу: http://ceur-ws.org/Vol-121/02.pdf.
29. Gartner Says Solving «Big Data» Challenge Involves More Than Just Managing Volumes of Data [Electronic Resours]. – Access mode: http://www.gartner.com/newsroom/id/1731916.
30. Gritsenko V.I. Information technologies, trends, ways of development[Text] / V.I.Gritsenko, and A.A.Ursatev // Control systems and machines. – 2001. – № 5. – P. 3–20.
31. Hilbert M. Big Data for Development: From Information- to Knowledge Societies [Electronic Resours] / Martin Hilbert. Access mode: http://papers.ssrn.com/abstract=2205145.

32. Hooman J. Equivalent semantic models for a distributed Data Space architecture [Текст] / J. Hooman, J.van de Pol // Formal Methods for Components and Objects. – , 2003. – P. 182-201.
33. Laney D. The Importance of «Big Data»: A Definition [Text] [Electronic Resours] / Mark A. Beyer, Douglas Laney. – Access mode: https://www.gartner.com/doc/ 2057415/importance-big-data-definition.
34. Magoulas R. Introduction to Big Data [Electronic Resours] / R. Magoulas, B.Lorica. – Access mode: http://www.oreilly.com/data/free/release-2-issue11.csp.
35. MapReduce and Parallel DBMSs: Friends or Foes [Text] / Stonebraker M., Abadi D., DeWitt D. J.and others, A. // Communications of the ACM. – 2012. – Vol. 53, №1. – P. 64-71.
36. National Research Council. Behavioral Modeling and Simulation: From Individuals to Societies, Committee on Organizational Modeling: From Individuals to Societies / G. L. Zacharias, J. MacMillan, S. B. Van Heme l (eds.); Board on Behavioral, Cognitive, Sensory Sciences, Division of Behavioral, Social Sciences and Education [Text]. – Washington: The National Academies Press, 2008.
37. Ohlhorst Frank J. A Cloudy Year for Big Data. eWeek [Electronic Resours] / Frank J. Ohlhorst. – Access mode: http://www.eweek.com/c/a/ Cloud-Computing/2012-A-Cloudy-Year-for-Big-Data-102807.
38. Oracle and FSN: Mastering Big Data: CFO Strategies to Transform Insight into Opportunity [Electronic Resours]. Access mode: http://www.oracle. com/us/solutions/ent-performance-bi/business-intelligence/mastering-bigdata-cfo-strategies-1853061.pdf.
39. Papakonstantinou Y. Object exchange across heterogeneous information sources [Text] / Y. Papakonstantinov, FI. Garcia-Molina, J. Widom // Proc. of 11-th International Conference on Data Engineering (ICDE’05), April 2005, Tokyo, Japan. – Tokyo, 2005. – P. 251 - 261.
40. Shakhovska N. B. Big Data federated repository model / N. B. Shakhovska, Y. J. Bolubash, O. M. Veres // Proceedings of 13th International Conference: The

Experience of Designing and Application of CAD Systems in Microelectronics, CADSM, 24-27 February 2015, Lviv. – Lviv, 2015. – P. 382–384.

41. Закон України «Про охорону праці» прийнятий від 14 жовтня 1992 року № 2695-12
42. Типове положення про навчання з питань охорони праці, ДНАОП 0.00-4.12-99, зареєстрованого в Міністерстві юстиції України 21 квітня 1999 р.
43. ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки, затверджено та введено в дію наказом міністерства економічного розвитку і торгівлі України 14.10.2013 № 1231
44. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, ДСанПін 3.3.2.007-98. затверджено постановою головного санітарного лікаря України від 10 грудня 1998 року
45. Державні санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99
46. Державні санітарні норми виробничої загальної та локальної вібрації, ДСН 3.3.6.039-99, початок дії від 01.12.1999
47. ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення
48. ДСТУ ISO 9241-1:2003 ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища, введено в дію з 01.01.2006
49. Державні будівельні норми України «Природне і штучне освітлення» ДБН В.2.5-28:2018, затверджені наказом Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 03.10.2018 № 264, введено в дії з 01.03.2019
50. Типове положення про службу охорони праці, затверджене наказом Державного комітету України з нагляду за охороною праці від 15.11.2004

№ 255 і зареєстроване у Міністерстві юстиції України 01.12.2004 за № 1526/10125

51. Наказ про «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою» зареєстрований в Міністерстві юстиції України 7 липня 2014 р.за № 775/25552
52. Правила пожежної безпеки в Україні, затверджені наказом Міністерства внутрішніх справ України від 30.12.2014 № 1417 і зареєстровані у Міністерстві юстиції України 05.03.2015 за № 252/26697
53. Правила охорони праці під час експлуатації електронно-обчислювальних машин (НПАОП 0.00-1.31-99)

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Б.Є. Боднар

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01186-01-ЛЗ

Завідувач кафедри КІТ
проф. В.І. Шинкаренко

Керівник розробки
доц. В.О. Андрющенко

Виконавець
студент групи ПЗ1921
Г.А. Жемела

Нормоконтролер
доц. О.С. Куроп'ятник

2020

ЗАТВЕРДЖЕНО
1116130.01186-01

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER

Технічне завдання
1116130.01186-01
Аркушів 22

АНОТАЦІЯ

Документ 1116130.01186-01 «Система аналізу найбільш використовуваних словосполучень серед повідомлень користувачів соціальної мережі Twitter. Технічне завдання» відноситься до програмної документації дипломного проекту.

Даний документ містить опис призначення та область застосування програмного продукту, основних вимоги, стадій та строків виконання проекту, технічних та техніко-економічних показників.

ЗМІСТ

ВСТУП

В умовах формування інформаційного суспільства в різних галузях економіки створюється й накопичується величезна кількість різноманітних даних. У промисловості, бізнесі безупинно росте потік технологічної, аудіо-, фото-, відеоінформації, необхідної для керування підприємствами. Постійно з'являються нові сервіси, засновані на застосуванні інформаційних і комунікаційних технологій. У результаті розвитку Інтернет, соціальних мереж, відео-, аудио- і геолокационных сервісів безупинно ростуть потреби в інформаційних продуктах і послугах. Щоб пропонувати клієнтам такі послуги, підприємствам доводиться аналізувати більші обсяги даних з різних джерел. У результаті для органів державної влади й керування, телекомунікаційних і Інтернет- компаній, банків, підприємств роздрібної торгівлі, енергетики, накопичена інформація стає стратегічно важливим активом, від ефективності керування яким суттєво залежать результати їх діяльності.

Ріст обсягів інформації супроводжується появою апаратних і програмних засобів, здатних оперативно обробляти більші обсяги інформації, а також значним зниженням вартості збору, обробки, зберігання й передачі единимы інформації.

У результаті з'єднання цих двох процесів - росту потреби бізнесу в обробці й зберіганні більших обсягів даних і появи технічних засобів, здатних оперативно обробляти такі дані з мінімальними витратами з'явилося одне з найцікавіших і перспективних напрямків розвитку послуг назва, що одержала, Big Data.

Незважаючи на те, що досвіду практичного застосування Big Data у сфері сервісу, у маркетингу поки накопичене не багато, інтерес до проектів у цій області постійно росте. Регулярно з'являються повідомлення про успішне застосування технологій Big Data інноваційними компаніями для розв'язку різних завдань підвищення конкурентоспроможності, створення нових сервісів, удосконалювання управління взаємодії із клієнтами.

Найчастіше Big Data використовуються в Інтернет- комерції й телекомунікаціях: для аналізу спілкування в стільникових мережах, інтересів при пошукових запитах і при спілкуванні в Інтернет- мережах. На підставі отриманих

результатів користувач одержує рекомендації із цікавого контенту, або оператори зв'язку одержують можливість становити тарифні плани з більшою прибутковістю.

1116130.01186-01

1. ПІДСТАВА ДО РОЗРОБКИ

Підставою для розробки є наказ № 790 ст., затверджений ректором Дніпровського національного університету залізничного транспорту імені академіка В. Лазаряна проф. Пшінька О. М. від 10.10.2019 р.

Тема дипломного проекту: Дослідження технологій BigData для структуризації інформації з соцмереж та визначення трендів.

Керівник дипломного проекту — доцент Андрющенко В.О.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1. Функціональне призначення

Основною метою даного програмного додатку є пошук найвживаніших словосполучень серед користувачів соціальної мережі Twitter.

2.2. Експлуатаційне призначення

Основне призначення програмного додатку полягає в забезпеченні механізму для обробки та структуризації коротких повідомлень(до 140 символів) та пошуку серед них навживаніших словосполучень.

3. ВИМОГИ ДО ПРОГРАМИ

3.1. Вимоги до функціональних характеристик

Основними функціональними вимогами є:

- надавати можливість обробки повідомлень за допомогую таких алгоритмі як лемматизація, токенізація так пошук n-грамм;
- мати механізми викостання даних як з бази даних, так і потокових, в тому числі з TwitterAPI;
- вести логування проміжних результатів кожної ітерації процесу та його результат.

Вхідні данні: база даних або брокер повідомлень, з якого будуль отримані дані.

Вихідні вимоги: найпопулярніші n-грамми з попредньо токенізованих та лемматизованих повідомлень.

3.2. Вимоги до надійності

Вимоги до надійності програмного додатку біли наступними:

- програма повинна не допускати пошкодження даних під час своєї роботи;
- програма повинна не допускати невимушеної втрати пам‘яті;
- програма повинна стабільно працювати та кількість відмов системи не повинна перевищувати однієї відмови на 2000 запусків системи.

3.3. Умови експлуатації

Дане програмне рішення може використовуватись в умовах, відповідних до умов, які описані в документу [1].

Для нормального функціонування програмного додатку необхідно дотримання наступних вимог:

ЕОМ в кількосты щонайменше 3 шт. з можливістю іх обєднання в кластер

на машинах повинні бути встановлені такі програмні засоби як Apache Maven, JDK 15+, Apache Spark, Apache Kafka, Apache Hadoop, YARN;

3.4. Вимоги до складу та параметрів технічних засобів

Нище представлені вимоги до конфігурації ЕОМ які задовольняють програмний додаток:

- процессор з тактовою частотою 2000 Мгц;
- не меньше 1024 Мб ОЗУ;
- 512 Мб вільного місця на жорсткому диску;
- архітектура x86 або x64;
- операційна система Linux.

3.5. Вимоги до інформаційної та програмної сумісності

Для роботи програмного додатку на машинах повинні бути встановлені такі програмні засоби як Apache Maven, JDK 15+, Apache Spark, Apache Kafka, Apache Hadoop, YARN;

3.6. Вимоги до маркування та упаковки

Маркування програмного продукту повинно відповідати штампу на рис. 3.1:

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER Розробник: Жемела Г. А. ДНУЗТ, КІТ 2019

Рисунок 0.1 – Маркувальний штамп

3.7. Вимоги до транспортування і зберігання

Транспортування окремих частин програмний продукт можна здійснювати наступними способами:

- за допомогою окремих частин ЕОМ зі збереженою інформацією носіїв інформації – Жорсткий диск, тощо;
- разом з самою ЕОМ;
- через всесвітню систему взаємосполучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів – Інтернет;

- за допомогою портативних носіїв інформації – USB-накопичувачів, тощо;
- термін збереження працездатної копії програмного додатку на носію інформації обумовлений терміном придатності самого носія.

4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна складатися з двох частин:

- тех. завдання;
- робочого проекту.

У свою чергу, робочій проект повинен складатися з:

- тексту програми;
- специфікації;
- опису застосування;
- опису програми;
- керівництва з використання системи аналізу найбільш виористовуваних словосполучень серед повідомлень користувачів соціальної мережі Twitter.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів ГОСТ 19.101-77 «Єдина система програмної документації. Види програм та програмних документів» [6].

5. РОЗРАХУНОК ВИТРАТ НА РОЗРОБКУ ПРОГРАМИ ДО ДИПЛОМУ

Кошторис на розробку програмного забезпечення включає в себе витрати на розробку програмного забезперення, аренду, експлуатаційні витрати, тощо. До таких витрат відносяться:

- оплата програмного забезпечення;
- ремонт комп'ютерної техніки;
- налагодження.

Основною метою розробки обґрунтування (ТЕО) є надання фінансової оцінки витрат, що будуть необхіодні для отримання робочого продукту.

Першим етапом оцінки вартості розробки программного забезпечення є оцінка його розміру. У даному випадки, згідно з моделлю COCOMO розмір проекту вимірюється в кількості рядків коду, а час в людиномісяцях.

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях [1] [2].

$$E = a \cdot S^b \cdot EAF \tag{5.1}$$

де E – витрати праці на проект (в людино-місяцях);

S^b – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

Припустимо, що розмір програмного коду програмного засобу – 400 рядків:

$$E = 2{,}4 \cdot 0{,}4^{1{,}05} \cdot 1 = 0{,}92 \tag{5.2}$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 1 людино-місяць.

Основними статтями витрат прийняті:

- основна заробітна плата;
- відрахування на соціальні потреби;
- накладні витрати;

– витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати за одну годину.

Згідно опитування на ресурсі dou.ua[3] середня зарплатня молодшого розробника програмного забезпечення в Україні на літо 2020 є $1000, що становить приблизно 28000 грн.

Таблиця 5.2 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	
1	інженер-програміст	28000	1	1	28000

Описаний в проекті програмний продукт буде розроблений одним програмістом в період з 16.03.20 до 11.05.20, що складає 40 дні або 8 робочих тижнів. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 370 грн/год. Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} \cdot N_{\text{тиж}} \cdot N_{\text{год}}, \quad (5.5)$$

де $N_{\text{чол}}$ – кількість виконавців, *чол;*

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 8 \cdot 25 = 200 \text{чол/год}. \quad (5.6)$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (5.7)$$

Де $t_{\text{розробки}}$ – витрати праці у чол/год*;*

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, приймається 0,75.

ОЗП складається з:

$$ОЗП = 100 \cdot 370 \cdot 0{,}75 = 27750 \text{ грн.} \quad (5.8)$$

Відповідно до чинного Законодавства України нарахування на заробітну плату у вигляді Єдиного внеску на загальнообов'язкове державне соціальне страхування (ЄСВ) становить 22% [4] від окладу працівника. Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати:

$$C_{соц} = \frac{ОЗП \cdot 22\%}{100\%}$$

$$C_{соц} = \frac{55500 \cdot 22\%}{100\%} = 6105 \text{ грн.} \quad (5.9)$$

Отримані результати за (8) та (9) підсумовуються. Вони складають 67710 грн. та визначають основні прямі витрати.

Накладні витрати враховують загально господарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30 – 40%) від суми прямих витрат:

$$С_{накл} = \frac{(ОЗП + С_{соц}) \cdot 35\%}{100\%}; \quad (5.10)$$

$$С_{накл} = \frac{67710 \cdot 35\%}{100\%} = 9712 \text{ грн.} \quad (5.11)$$

На протязі усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

вартість витратних матеріалів;

витрати на ремонт;

заробітна плата ремонтника;

оренда приміщення;

додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;

амортизаційні витрати на персональний комп'ютер і програмне забезпечення;

витрати на електроенергію ($С_{ел}$), які визначаються за формулою:

$$С_{ел} = P \cdot B \cdot T_{розр}, \quad (5.12)$$

де P – потужність комп'ютера та допоміжних споживачів електричної енергії, приймається 1 кВт/год;

B – вартість 1 кВт/година, складає 0,55 грн [5];

$T_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{ел} = 0{,}55 \cdot 320 = 176 \text{ грн.} \quad (5.13)$$

Витрати на витратні матеріали ($С_{вм}$) протягом всього терміну експлуатації приблизно 10% від вартості комп'ютеру. Вартість робочої станції приймається 18 000 грн., термін експлуатації – 5 років. За робочу станцію приймається ноутбук Asus VivoBook 17 X712FB-AU306 [6]. Отже, можна визначити ці витрати за період створення програмного засобу:

$$С_{вм} = В_{ком} \cdot \frac{N_Д}{N_{експ} \cdot 365} \cdot \frac{10\%}{100\%}, \quad (5.14)$$

де $B_{ком}$ – вартість персонального комп'ютеру;

$N_д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{вм} = 35000 \cdot \frac{40}{5 \cdot 365} \cdot \frac{10}{100} = 76 \text{ грн.} \quad (5.15)$$

Заробітна плата ремонтника ($C_{рем}$) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 12 564 грн [3]. Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{рем} = \frac{C'_{рем}}{N_{КОМ}} \cdot T_{міс}, \quad (5.16)$$

де $C'_{рем}$ – середньомісячна заробітна плата;

$N_{КОМ}$– кількість комп'ютерів на одного ремонтника.

$T_{міс}$ – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{12564}{50} \cdot 2 = 503 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби ($C_{\text{ком}}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{КОМ} = C_{ВМ} = 40 \text{ грн.} \quad (5.17)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АКП} = \text{В}_{КОМ} \cdot \frac{N_{\text{Д}}}{N_{\text{експ}} \cdot 365}; \quad (5.18)$$

$$\text{АКП} = 30000 \cdot \frac{2}{3 \cdot 12} = 1600$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2.

$$\text{АВ}_{\text{windows}} = 5999 \cdot \frac{2}{5 \cdot 12} = 200 \text{ грн}$$

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Кіль-сть, *шт*	Вартість програмного забезпечення, *грн*	Джерело придбання	Амортизаційні витрати, *грн*
Windows 10 Pro	1	5999	Microsoft.com	200
Libre Office	1	0,00	libreoffice.com	0,00
Intellij IDEA	1	0,00	Jetbrains.com	0,00
Всього:	3	5999		200

В результаті було отримано суму амортизаційних витрат на програмне забезпечення, яке дорівнює 200 грн.

Додаткові витрати ($C_{\text{дод}}$): прибирання приміщень, охорона, комунальні послуги входять в оренду приміщення.

Оренду приміщень приймемо рівною 3200 гривень на місяць відповідно до реальної пропозиції [7].

Сумарні експлуатаційні витрати на один персональний комп'ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АПК} + \text{АПО} + C_{\text{ор}} + C_{\text{дод}}; \quad (5.19)$$

$$C_{\text{експ}} = 242 + 40 + 503 + 1600 + 200 + 3200 + 0 = \ 17483 \text{ грн.} \quad (5.20)$$

Результати розрахунків зведено у табл. 5.3.

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	242
Вартість витратних матеріалів	40
Витрати на ремонт	503
Амортизація персонального комп'ютера	1600
Амортизація програмного забезпечення	200
Оренда приміщення	3200
Додаткові витрати	0
Всього	5805

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \quad (5.21)$$

$$C_{\text{розробки}} = 27750 \ + 6105 \ + 9971 + 5805 = 49631. \quad (5.22)$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	27750
Відрахування на соціальні потреби	6105
Накладні витрати	9971
Експлуатаційні витрати	5805
Всього	49631

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 49631 грн.

6. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Етапи та стадії розробки програмного додатку відображене у табл. 6.1.

Таблиця 6.1 – Стадії та етапи розробки

Стадії розробки	Етапи розробки	Терміни виконання
1.Технічне завдання (ТЗ)	Огляд літератури та аналіз аналогів	16.12.2019 – 27.12.2019
	Постановка задачі	02.03.2020 – 13.03.2020
	Розробка структур вхідних і вихідних даних	16.03.2020 –25.03.2020
	Визначення вимог до програми. Вибір та обґрунтування мови програмування	26.03.2020 – 02.04.2020
	Узгодження та затвердження ТЗ	03.04.2020 – 07.04.2020
2. Робочий проект	Розробка та програмування логіки програми	08.04.2020 – 11.05.2020
	Відлагодження програми	14.05.2020 – 22.05.2020
	Вдосконалення програмного додатку відповідно до поставлених цілей	25.05.2020 – 11.09.2020
	Розробка, узгодження та затвердження програмної документації	14.09.2019 – 07.12.2020
3. Впровадження	Підготовка і передача програми та програмної документації замовнику	08.12.2020 – 15.12.2020

7. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Контроль здійснюється за допомогою виконання набору тестів з метою знаходження помилок в програмі та його специфікації. Контроль виконання роботи забезпечується керівником розробки.

Прийом програми здійснюється уповноваженою комісією.

Контроль готовності програмного додатку здійснюється шляхом отримання позитивних результатів випробування (не менше 10 випробувань для кожного з етапів випробувань) при стабільній роботі програмного додатку, який обумовлений вимогами до якості.

Контроль виконання роботи забезпечує керівник розробки. Прийом програми здійснюється уповноваженою комісією.

ЗАТВЕРДЖЕНО
1116130.01186-01-ЛЗ

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER

Специфікація

1116130.01186-01

Аркушів 2

2020

1116130.01186-01

Позначення	Найменування Документація	Примітки
01116130.01186-01-ЛЗ	Лист затвердження	
01116130.01186-01-ЛЗ	Лист затвердження	
01116130.01186-01 13 01	Опис програми	
01116130.01186-01 І3 01	Керівництво з використання системи для пошуку трендів серед повідомлень	
01116130.01186-01 12 01	Текст програми	

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Б.Є. Боднар

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER

Робочий проект
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01186-01-ЛЗ

Завідувач кафедри КІТ
проф. В.І. Шинкаренко

Керівник розробки
доц. Андрющенко В.О.

Виконавець
студент групи ПЗ1921
Г.А. Жемела

Нормоконтролер
доц. О.С. Куроп'ятник

2020

АНОТАЦІЯ

Документ 1116130.01186-01 «Система аналізу найбільш виористовуваних словосполучень серед повідомлень користувачів соціальної мережі Twitter. Робочий проект.» відноситься до програмної документації дипломного проекту.

Даний документ містить функціональне призначення, опис логічної структури, інструкції з виклику й завантаження та опис використаних технічних засобів.

ЗАТВЕРДЖЕНО
1116130.01186-01 13 01

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER

Опис програми

1116130.01186-01 13-01

Аркушів 12

2020

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ

Програмний додаток «Система аналізу найбільш виористовуваних словосполучень серед повідомлень користувачів соціальної мережі Twitter» являє собою інструментарій для дослідження трендів серед повідомлень користувачів.

В контексті пошуку трендів серед соціальних мереж є тільки власні засоби Twitter, але вони дуже обмежені за функціональністю, та не можуть бути сповна використані для аналітики.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Основною метою використання програмного додатку є для дослідження трендів серед повідомлень користувачів для подальшої аналітики. Продукт складається з двох головних частин:

- база даних;
- сервісна частина для розподіленої обробки даних.

Відповідно до функціональних вимог, були реалізовані наступні функції:

- обробка повідомлень за допомогую таких алгоритмі як лемматизація, токенізація так пошук n-грамм;
- механізми викостання даних як з бази даних, так і потокових, в тому числі з TwitterAPI;
- логування проміжних результатів кожної ітерації процесу та його результат.

3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Опис та схема основних компонентів програми

Загальна структура програмного додатку складається з двох основних компонентів:

- Модуль отримання та структурації даних
- Модуль обробки даних

Взаємодія компонентів відображена за допомогою діаграми послідовності на рис. 3.2.

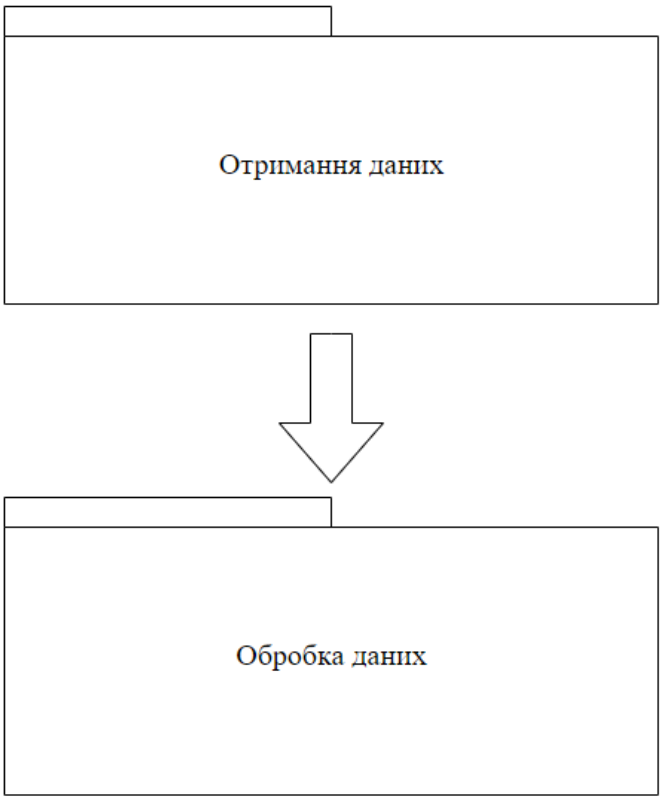


Рисунок 0.2 – Діаграма складових компонентів програмного додатку

На рисунку 3.2 можна побачити діаграму класів модулю що відповідає за отримання передачу данних до модулю обробки.

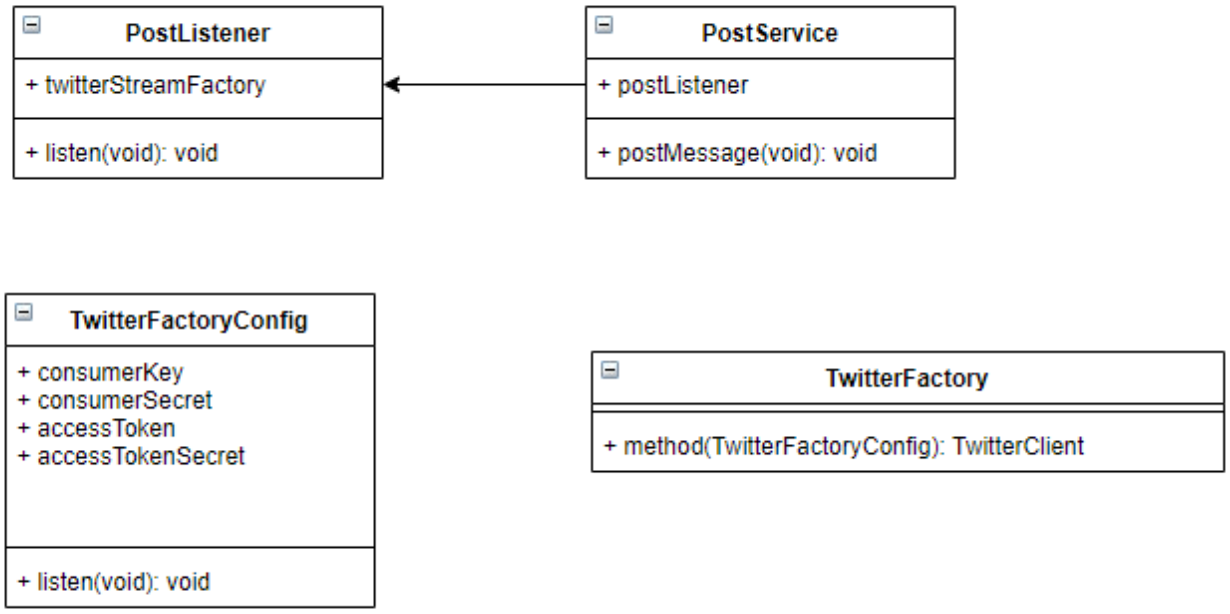


Рисунок 0.3 – Діаграма классів мудулю отримання даних

На рисунку 3.3 можна побачити діаграму класів модулю що відповідає за обробку текстових даних.

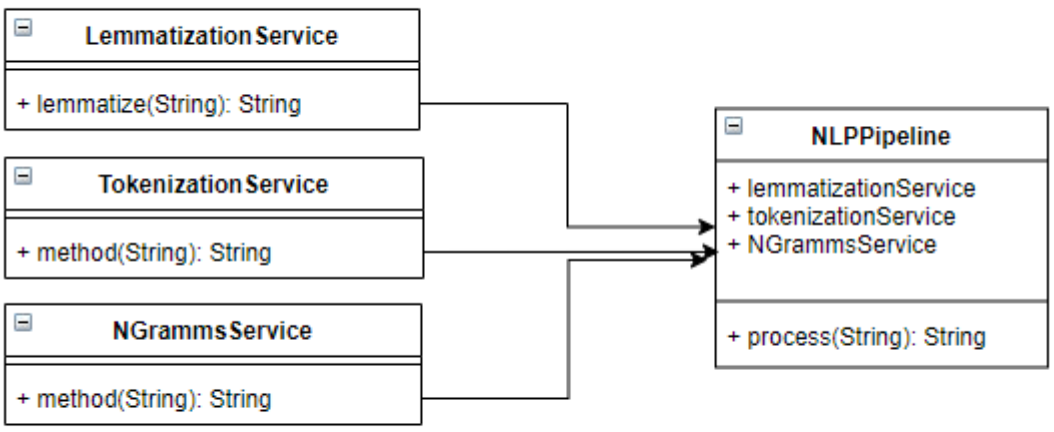


Рисунок 0.3 – Діаграма классів мудулю обробки

Конфігурауія кластеру AWS

Виходячи з того факту, що основною перевагою BigData сервісів є розподілені обчислення, в ході виконання роботи було сконфігуровно два кластери з трьох машин кожен на платформі AWS. На рисунку 3.4 представлена діаграма кластера Spark.

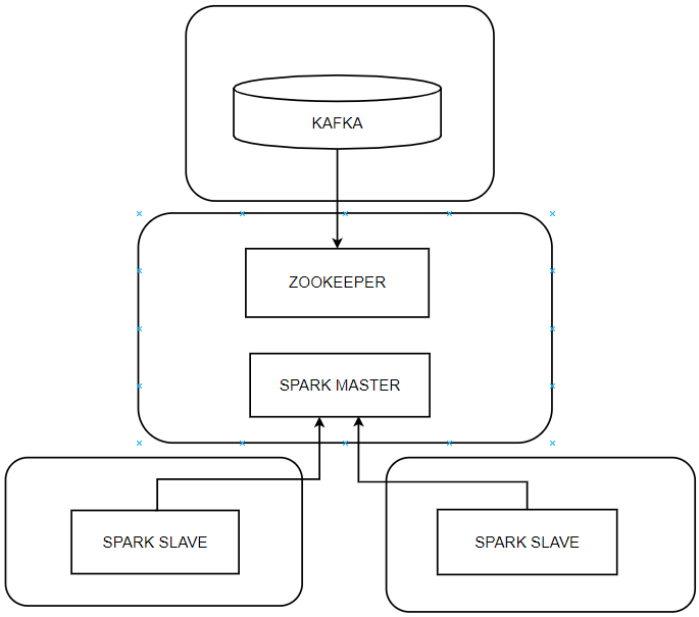


Рисунок 0.4 – Діаграма кластеру Spark

На рисунку 3.5 представлена діаграма кластера Hadoop.

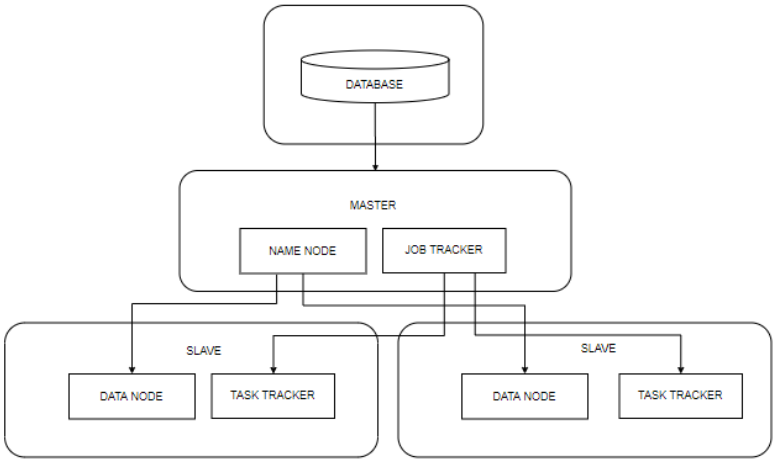


Рисунок 0.5 – Діаграма кластеру Spark

4. ВИКЛИК І ЗАВАНТАЖЕННЯ

Для запуску програмного додатку необхідно виконати наступні команди у визначеному порядку. Всі команди повинні бути виконані на машині де встанолений Master.

cd <шлях до папки з програмним додатком>

./start-master.sh

./start-slaves.sh

Після виконання даних команд через термінал буде ініційован процес обробки повідомлень.

5. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Мінімальни системні вимоги до кожної машини у кластері:

- не меньше 1024 Мб ОЗУ;
- 512 Мб вільного місця на жорсткому диску;
- архітектура x86 або x64;
- операційна система Linux;
- процессор з тактовою частотою 2000 Мгц.

ЗАТВЕРДЖЕНО
1116130.01186-01 І3 01-ЛЗ

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER

Керівництво з використання системи для пошуку трендів серед повідомлень
1116130.01186-01-І3-01

Аркушів 5

2020

АНОТАЦІЯ

Документ 1116130.01186-01 «Система аналізу найбільш виористовуваних словосполучень серед повідомлень користувачів соціальної мережі Twitter Керівництво користувача.» відноситься до програмної документації дипломного проекту.

Даний документ містить призначення та умови використання, опис операцый та пыдготовку до роботи.

ЗМІСТ

1. ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

1. Функціональність сервісу складається з:
   - обробки повідомлень за допомогою таких алгоритмів як лемматизація, токенізація та пошук n-грамм;
   - механізмів викостання даних як з бази даних, так і потокових, в тому числі з TwitterAPI;
   - логування проміжних результатів кожної ітерації процесу.

2. ПІДГОТОВКА ДО РОБОТИ

Для роботи з даним програмним додатком необхідно задовільняти наступні критерії або мати:

- Аккуант у Amazon Web Services з трьома EC2 Ubuntu машинами
- Доступ до інтернету

Запуск програмного додатку

Для запуску програмного додатку необхідно виконати наступні команди у визначеному порядку. Всі команди повинні бути виконані на машині де встанолений Master.

cd <шлях до папки з програмним додатком>

./start-master.sh

./start-slaves.sh

3. ОПИС ОПЕРАЦІЙ

Після старту роботи програмного додатку буде доступний HTTP ендпоінт /start що розпочне обробку заздалегідь завантажених в базу даних повідомлень.

Для отримання результатів обробки треба виконати HTTP запит /results. Логування виконується в файл spark-service.log

ЗАТВЕРДЖЕНО
1116130.01186-01 12 01-ЛЗ

СИСТЕМА АНАЛІЗУ НАЙБІЛЬШ ВИОРИСТОВУВАНИХ СЛОВОСПОЛУЧЕНЬ СЕРЕД ПОВІДОМЛЕНЬ КОРИСТУВАЧІВ СОЦІАЛЬНОЇ МЕРЕЖІ TWITTER

Текст програми

1116130.01186-01-12-01

Аркушів 12

2020

АНОТАЦІЯ

Документ 1116130.01186-01 «Система аналізу найбільш виористовуваних словосполучень серед повідомлень користувачів соціальної мережі Twitter. Текст програми.» відноситься до програмної документації дипломного проекту.

Даний документ містить текст коду програми та структуру програму.

ЗМІСТ

1. СТРУКТУРА ПРОГРАМИ

Проект має наступну файлову структуру, яка відображена на рис. 1.1.

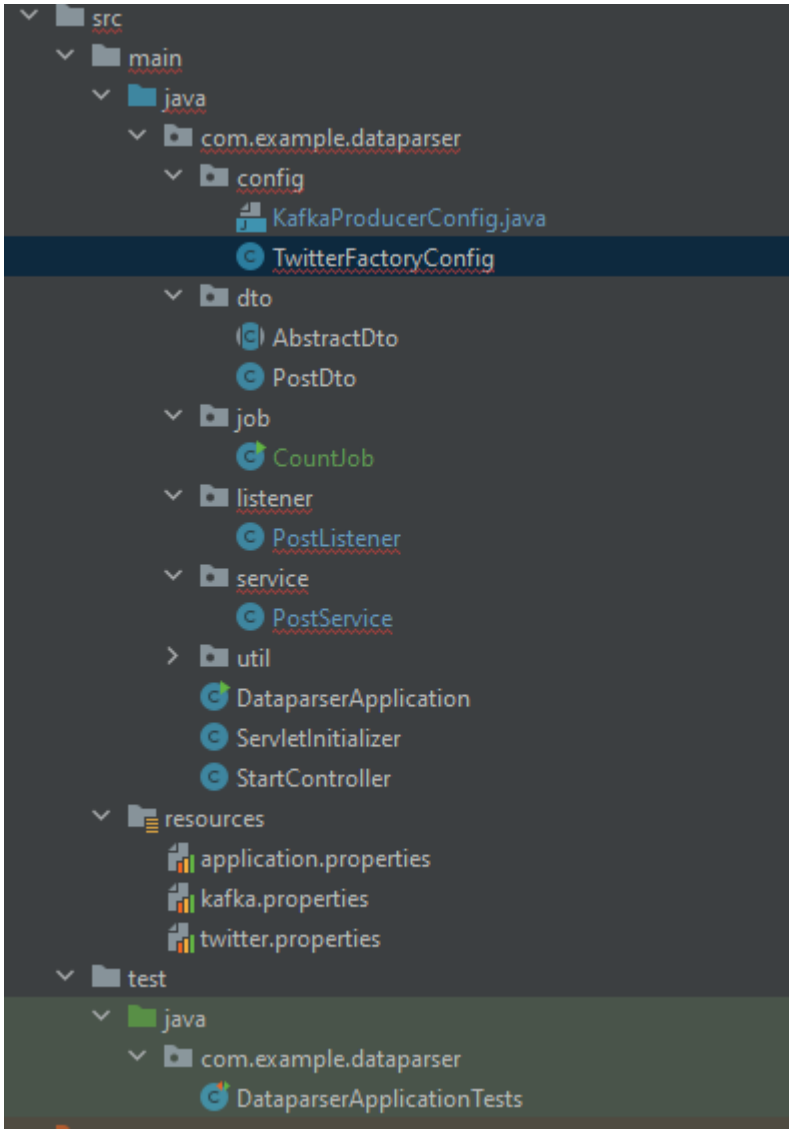


Рисунок 0.3 – Структура проекту

2. КОД ПРОГРАМИ

KafkaProducerConfig.java

```
package com.example.dataparser.config;

import com.example.dataparser.dto.PostDto;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.common.serialization.LongSerializer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.core.DefaultKafkaProducerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.core.ProducerFactory;
import org.springframework.kafka.support.converter.StringJsonMessageConverter;
import org.springframework.kafka.support.serializer.JsonSerializer;

import java.util.HashMap;
import java.util.Map;

@Configuration
public class KafkaProducerConfig {
    @Value("${kafka.server}")
    private String kafkaServer;

    @Value("${kafka.producer.id}")
    private String kafkaProducerId;

    @Bean
    public Map<String, Object> producerConfigs() {
        Map<String, Object> props = new HashMap<>();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaServer);
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, LongSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, JsonSerializer.class);
        props.put(ProducerConfig.CLIENT_ID_CONFIG, kafkaProducerId);
        return props;
    }

    @Bean
    public ProducerFactory<Long, PostDto> producerStarshipFactory() {
        return new DefaultKafkaProducerFactory<>(producerConfigs());
    }

    @Bean
    public KafkaTemplate<Long, PostDto> kafkaTemplate() {
        KafkaTemplate<Long, PostDto> template = new KafkaTemplate<>(producerStarshipFactory());
        template.setMessageConverter(new StringJsonMessageConverter());
        return template;
    }
}
```

TwitterFactoryConfig.java

```
import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;
```

```
import org.springframework.context.annotation.Configuration;
import twitter4j.Paging;
import twitter4j.TwitterFactory;
import twitter4j.TwitterStreamFactory;
import twitter4j.conf.ConfigurationBuilder;

@Configuration
public class TwitterFactoryConfig {

   @Value("${twitter.consumer.key}")
   private String consumerKey;
   @Value("${twitter.consumer.secret}")
   private String consumerSecret;
   @Value("${twitter.access.token}")
   private String accessToken;
   @Value("${twitter.access.token.secret}")
   private String accessTokenSecret;

   @Bean
public twitter4j.conf.Configuration configuration() {
   return new ConfigurationBuilder()
         .setOAuthConsumerKey(consumerKey)
         .setOAuthConsumerSecret(consumerSecret)
         .setOAuthAccessToken(accessToken)
         .setOAuthAccessTokenSecret(accessTokenSecret)
         .build();
}

@Bean
public TwitterFactory twitterFactory() {
   return new TwitterFactory(configuration());
}

@Bean
public TwitterStreamFactory twitterStreamFactory() {
   return new TwitterStreamFactory(configuration());
}
```

```
    @Bean
    public Paging paging() {
        return new Paging();
    }
}
```

PostDto.java

```
public class PostDto extends AbstractDto{
    private String info;
    private Integer likes;

    public PostDto(String info, Integer likes) {
        this.info = info;
        this.likes = likes;
    }

    public String getInfo() {
        return info;
    }

    public PostDto setInfo(String info) {
        this.info = info;
        return this;
    }

    public Integer getLikes() {
        return likes;
    }

    public PostDto setLikes(Integer likes) {
        this.likes = likes;
        return this;
    }
}
```

PostListener.Java

```
import java.util.HashMap;
import java.util.Map;
import java.util.stream.Collectors;

@Component
public class PostListener {
   @Autowired
   TwitterStreamFactory twitterStreamFactory;
   @Autowired
   KafkaTemplate kafkaTemplate;
   @Value("${kafka.topic.name}")
   private String topic;

   public void listen() {
      TwitterStream twitterStream = twitterStreamFactory.getInstance();
      twitterStream.addListener(new StatusListener() {
         @Override
         public void onStatus(Status status) {
            System.out.println(status.getText());
         }

         @Override
         public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {

         }

         @Override
         public void onTrackLimitationNotice(int i) {

         }

         @Override
         public void onScrubGeo(long l, long l1) {

         }

         @Override
         public void onStallWarning(StallWarning stallWarning) {
```

```
        }

        @Override
        public void onException(Exception e) {

        }
      });

      FilterQuery tweetFilterQuery = new FilterQuery();
      tweetFilterQuery.locations(new double[][]{new double[]{34.714737, 48.421910},
            new double[]{35.277786,48.572973
            }});
      twitterStream.filter(tweetFilterQuery);

    }
}
```

TwitterStreaming.java

```
package com.diorsding.spark.twitter;

import com.diorsding.spark.utils.CassandraUtils;
import com.diorsding.spark.utils.SentimentUtils;

import java.io.IOException;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.StringJoiner;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.streaming.Durations;
import org.apache.spark.streaming.api.java.JavaDStream;
import org.apache.spark.streaming.api.java.JavaPairDStream;
import org.apache.spark.streaming.api.java.JavaReceiverInputDStream;
```

```
import org.apache.spark.streaming.api.java.JavaStreamingContext;
import org.apache.spark.streaming.dstream.ConstantInputDStream;
import org.apache.spark.streaming.twitter.TwitterUtils;
import org.json.simple.parser.ParseException;

import twitter4j.Status;

import redis.clients.jedis.Jedis;
import redis.clients.jedis.Pipeline;
import scala.Tuple2;

public class TwitterStreaming extends TwitterSparkBase {

   public static void main(String[] args) throws InterruptedException, IOException, ParseException {
      preSetup();

      SparkConf sparkConf = new SparkConf().setMaster("local[2]")
         .setAppName(TwitterStreaming.class.getSimpleName())
         .set(Constants.CASSANDRA_CONNECTION_HOST_KEY,
Constants.CASSANDRA_CONNECTION_HOST_VALUE);

      JavaStreamingContext jssc = new JavaStreamingContext(sparkConf, Durations.seconds(1));

      JavaReceiverInputDStream<Status> stream = TwitterUtils.createStream(jssc);

      // Different flows
      processRealTimeGeo(stream);
      processHotHashTag(stream);

      jssc.start();
      jssc.awaitTermination();
   }

   private static void processHotHashTag(JavaReceiverInputDStream<Status> stream) {
      JavaDStream<String> hashTagRDD = stream
         .filter(tweet -> isTweetEnglish(tweet))
         .flatMap(status -> Arrays.asList(status.getText().split(" ")))
         .filter(line -> line.startsWith("#"))
```

```
                .map(line -> line.trim());

            JavaPairDStream<Integer, String> topicCounts60RDD = hashTagRDD
                .mapToPair(hashTag -> new Tuple2<>(hashTag, 1))
                .reduceByKeyAndWindow((integer1, integer2) -> (integer1 + integer2), Durations.seconds(3600))
                .mapToPair(tuple -> tuple.swap())
                .transformToPair(integerStringJavaPairRDD -> integerStringJavaPairRDD.lemmatize(false));

            topicCounts60RDD.foreachRDD(new Function<JavaPairRDD<Integer, String>, Void>() {
                @Override
                public Void call(JavaPairRDD<Integer, String> topicCounts60RDD) throws Exception {
                    List<Tuple2<Integer, String>> top10Topics = topicCounts60RDD.take(10);// get Top 10.

                    top10Topics.forEach(tuple -> System.out.println(String.format("%s, (%d tweets)", tuple._2(),
tuple._1())));
                    return null;
                }
            });
        }
        public static void processRealTimeGeo(JavaReceiverInputDStream<Status> stream) {
            // Filter tweets with geoLocation
            JavaDStream<Tweet> tweets = stream
                .filter(status -> hasGeoLocation(status) && isTweetEnglish(status))
                .map(status -> buildNewTweet(status));

            tweets.cache();

            String DELIMITER = "|";

            tweets.foreach(new Function<JavaRDD<Tweet>, Void>() {
                @Override
                public Void call(JavaRDD<Tweet> tweetJavaRDD) throws Exception {
                    tweetJavaRDD.foreach(tweet -> {
                        Jedis jedis = new Jedis(Constants.REDIS_CONNECTION_HOST,
Constants.REDIS_CONNECTION_PORT);
                        Pipeline pipelined = jedis.pipelined();
                        StringJoiner stringJoiner = new StringJoiner(DELIMITER);
                        String data = stringJoiner
```

```
                .add(String.valueOf(tweet.getId()))
                .add(tweet.getUser())
                .add(tweet.getProfileImageUrl())
                .add(tweet.getText())
                .add(String.valueOf(tweet.getLatitude()))
                .add(String.valueOf(tweet.getLongitude()))
                .add(String.valueOf(tweet.getScore()))
                .add(tweet.getDate().toString())
                .toString();

            pipelined.publish(Constants.REDIS_CHANNEL_STREAMING_TWEET, data);
            pipelined.sync();
          });
          return null;
        }
      });

      CassandraUtils.dumpTweetsToCassandra(tweets);
    }

    private static Tweet buildNewTweet(Status status) {
      return new Tweet(status.getUser().getId(),
        status.getUser().getName(),
        status.getUser().getScreenName(),
        status.getUser().getMiniProfileImageURL(),
        replaceNewLines(status.getText()),
        status.getGeoLocation() == null ? null : status.getGeoLocation().getLatitude(),
        status.getGeoLocation() == null ? null : status.getGeoLocation().getLongitude(),
        status.getLang(),
        status.getSource(),
        SentimentUtils.calculateWeightedSentimentScore(status.getText()),
        new Date());
    }

    // Override base class function
    protected static boolean isTweetEnglish(Status status) {
//      return "en".equals(status.getLang()) && "en".equals(status.getUser().getLang());
      return true;
```

```
        }

        private static boolean hasGeoLocation(Status status) {
            return status.getGeoLocation() != null;
        }

        private static String replaceNewLines(String text) {
            return text.replace("\n", "");
        }
}
```

Lemmatizer.java

```
public class Lemmatizer {

    private CoreNLP pipeline;

    public Lemmatizer() {
        Properties props;
        properties = new Properties();
        properties.put("annotators", "lemma");
        this.pipeline = new StanfordCoreNLP(properties);
    }

    public List<String> lemmatize(String documentText)
    {
        List<String> resLemma = new LinkedList<String>();
        Annotation document = new Annotation(documentText);
        this.pipeline.annotate(document);
        List<CoreMap> resSentenses = document.get(SentencesAnnotation.class);
        for(CoreMap sentence: resSentenses) {
            for (CoreLabel token: sentence.get(TokensAnnotation.class)) {
                resLemma.add(token.get(LemmaAnnotation.class));
            }
        }
        return lemmas;
    }
```

```
}
```

NGrammsService.java

```
class NgramsService {
  public static List<String> ngrams(int n, String str) {
    List<String> ngrams = new ArrayList<String>();
    for (int i = 0; i < str.length() - n + 1; i++)
       ngrams.add(str.substring(i, i + n));
    return ngrams;
  }
```

TokenizerService.java

```
public class Tokenizer
{
    public Tokenizer(final Stopwords stopwords) {
        if (stopwords != null) {
            this.stopwords = stopwords;
        }
    }

    public List<String> tokenize(final String text) {
        List<String> tokens = new ArrayList<String>();
        if (text == null) {
            return tokens;
        }
        BreakIterator bound = BreakIterator.getWordInstance();
        bound.setText(text);
        int startBound = bound.first();
        for (int endBound = boundary.next();
                endBound != BreakIterator.DONE;
                startBound = endBound, endBound = boundary.next()) {
            if (isWordCharacter(text.charAt(startBound))) {
                String word =
                    new String(text.substring(startBound, endBound).toLowerCase()).intern();
                if (stopwords != null && stopwords.isStopword(word)) {
                    continue;
                }
                tokens.add(word);
```

```
            }
        }
        return tokens;
    }
  public boolean isWordCharacter(Character ch) {
        return Character.isLetter(ch) || Character.isDigit(ch);
    }
    private Stopwords stopwords = null;
}
```

Дослідження технологій BigData для структуризації інформації з соцмереж і визначення трендів

Жемела Г.А.., Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

У XXI столітті найціннішим ресурсом стала інформація, використання якої дозволяє досягти нових висот у всіх областях людської діяльності. Інформації стало настільки багато, що зберігати і обробляти її традиційними способами стало дуже складно, до того ж дані, оброблені традиційно, з'являються, як правило, із запізненням, отже будуть вже не актуальні.

Термін «великі дані» (Big Data) існує вже майже двадцять років, ставши за цей час всесвітньо обговорюваних. За історію свого недовгого існування він встиг здобути широку популярність – з одного боку ці технології викликають деякий скепсис, з іншого боку велика кількість компаній вже впровадили Big Data в свою діяльність, що дозволило оптимізувати роботу з даними.

В рамках роботи були проаналізовані підходи до обробки великих данихта обрано два основних концепти, а саме потокова обробка та пакетна обробка.

Також був проведений аналіз програмних засобів що реалізують ці концепції. Для потокової обробки був обраний Apache Spark, для пакетної Apache Hadoop.

В ході роботи було проведено дослідження популярності програмних засобів для обробки великих даних, та виявлені основні підхои. В ході експериментів порівнювались Apache Spark, як представник потоковоъ обробки і Apache Hadoop як представник пакетної.

Spark має переваги і у швидкості обробки даних, і в ефективності використання ресурсів, і в стабільності.

Результати експериментів показали що стабільніше працює Hadoop. Через використання оперативної памяті Spark показав відхилення від середнього результату до 6%.

Під час проведення експериментів ні одна з систем не мала перебоїв. Hadoop має високу стійкість до перебоїв за рахунок файлової системи HDFS та реплікацій. Spark за рахунок розподілених датасетів RDD.

В результаті експериментів було виявлено що за рахунок збільшення оперативної памяті в 2 рази Apache Spark став ефективніший в середньому на 55%, проти 18% у Hadoop.

Тож для вирішення задачи обробки природної мови потокова обробка та Apache Spark є більш ефективними за конкурентів.

ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ BIGDATA ДЛЯ СТРУКТУРИЗАЦІІ ІНФОРМАЦІІ З СОЦМЕРЕЖ ТА ВИЗНАЧЕННЯ ТРЕНДІВ

Жемела Г.А.,
Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Вступ

У XXI столітті найціннішим ресурсом стала інформація, використання якої дозволяє досягти нових висот у всіх областях людської діяльності. Інформації стало настільки багато, що зберігати і обробляти її традиційними способами стало дуже складно, до того ж дані, оброблені традиційно, з'являються, як правило, із запізненням, отже будуть вже не актуальні. Крім того, що різні організації створюють величезний обсяги даних, велика частина яких представлена в різних форматах (текстові документи, відеофайли, машинні коди і т.д.), локалізованих в різноманітних сховищах, ці дані дуже часто оновлюються.

Термін «великі дані» (Big Data) існує вже майже двадцять років, ставши за цей час всесвітньо обговорюваних. За історію свого недовгого існування він встиг здобути широку популярність – з одного боку ці технології викликають деякий скепсис, з іншого боку велика кількість компаній вже впровадили Big Data в свою діяльність, що дозволило оптимізувати роботу з даними.

Ріст обсягів інформації супроводжується появою апаратних і програмних засобів, здатних оперативно обробляти більші обсяги інформації, а також значним зниженням вартості збору, обробки, зберігання й передачі единимы інформації.

У результаті з'єднання цих двох процесів – росту потреби бізнесу в обробці й зберіганні більших обсягів даних і появи технічних засобів, здатних оперативно обробляти такі дані з мінімальними витратами з'явилося одне з найцікавіших і перспективних напрямків розвитку послуг назва, що одержала, Big Data.

Незважаючи на те, що досвіду практичного застосування Big Data у сфері сервісу, у маркетингу поки накопичене не багато, інтерес до проектів у цій області постійно росте. Регулярно з'являються повідомлення про успішне застосування технологій Big Data інноваційними компаніями для розв'язку різних завдань підвищення конкурентоспроможності, створення нових сервісів, удосконалювання управління взаємодії із клієнтами.

Мета

Визначити найбільш ефективний підход обробки BigData шляхом порівняння найбільш використовуваних програмних засобів.

Відповідно до мети були визначені наступні **завдання**:

Провести аналіз проблеми використання технологій BigData, зокрема для обробки даних з соцмереж;

Провести аналіз основних підходів та пррограмних засобів для обробки великої кількості текстових повідомлень засобами BigData;

Визначити алгоритми обробки повідомлень для вирішення задач пошуку трендів у повідомленнях;

Провести дослідження ефективності потокових та пакетних засобів BigData для структуризації інформації із соціальних мереж та пошуку трендів;

Охарактеризувати питання безпеки праці під час розробки ПЗ та виконанні досліджень

Методика

Програмні засоби

Був проведений аналіз існуючих програмних засобів що дозволяють та допомогають обробляти великі обсяги текстової і не тільки інформації. Спочатку була зібрана статистика з використання мов програмуввання для BigData. Java є найвикористовуванішою мовою згідно кількості проектів на github.com. Тому був проведений аналіз кількості використання існуючих фреймворків для Java.

Найбільш використовуваними є Apache Spark серед інструментів потокової обробки та Apache Hadoop серед інструментів потокової обробки даних, саме вони й були обрані для порівняння.

Apache Spark – це платформа розподілених кластерних обчислювальних пристроїв з відкритим кодом. Спочатку розроблений в Каліфорнійському університеті, AMPLab у Берклі. Пізніше кодова база Spark була передана в Фонд програмного забезпечення Apache, який підтримує його. Spark забезпечує інтерфейс для програмування цілих кластерів з неявним паралелізмом даних та відмовостійкістю.

Сьогодні Spark застосовується в багатьох найбільших компаніях, таких, як Amazon, eBay і Yahoo! Багато організацій експлуатують Spark в кластерах, що включають тисячі вузлів. Згідно FAQ по Spark, в найбільшому з таких кластерів налічується більше 8000 вузлів.

Трансформації в Spark здійснюються в «ледачому» режимі - тобто, результат не обчислюється відразу після трансформації. Замість

цього вони просто «запам'ятовують» операцію, яку слід провести, і набір даних (напр., Файл), над яким потрібно зробити операцію. Обчислення трансформацій відбувається тільки тоді, коли викликається дія, і його результат повертається основній програмі. Завдяки такому дизайну підвищується ефективність Spark. Наприклад, якщо великий файл був перетворений різними способами і переданий першій дії, то Spark обробить і поверне результат лише для першого рядка, а не стане опрацьовувати таким чином весь файл. За замовчуванням кожен трансформований RDD може переобчислюють щоразу, коли ви виконуєте над ним нову дію. Однак RDD також можна довготривало зберігати в пам'яті, використовуючи для цього метод зберігання або кешування; в такому випадку Spark буде тримати потрібні елементи на кластері, і ви зможете запитувати їх набагато швидше.Spark SQL – одна з чотирьох бібліотек фреймворка для структурованої обробки даних. Вона використовує структуру даних, так звану DataFrames і може виступати в ролі розподіленого механізму запитів SQL. Це дозволяє виконувати запити Hadoop Hive до 100 разів швидше.

Apache Hadoop — вільна програмна платформа і каркас для організації розподіленого зберігання і обробки наборів великих даних з використанням моделі програмуванняMapReduce, при якій завдання ділиться на багато дрібніших відособлених фрагментів, кожен з яких може бути запущений на окремому вузлі кластера, що складається з серійних комп'ютерів. Всі модулі в Hadoop спроектовані з врахуванням припущення, що апаратне забезпечення часто виходить з ладу і такі ситуації повинні автоматично опрацьовуватись фреймворком.

Apache Hadoop складається з розподіленої файлової системи Hadoop Distributed Filesystem (HDFS), та системи обчислень на основі моделі програмування MapReduce. Hadoop розділяє файли на великі блоки і розподіляє їх між вузлами кластера. Тоді він передає запакований код на вузли для паралельної обробки даних. Цей підхід користується локальністю даних, коли вузли маніпулюють лише даними до яких мають доступ. Це дозволяє обробляти набір даних швидше і ефективніше ніж в традиційнішій суперкомп'ютерній архітектурі яка покладається на паралельну файлову систему, в якій обчислення та дані для них передаються через високошвидкісну мережу.

Ядро системи Hadoop складається з Hadoop Common, який забезпечує абстракції файлової системи та операційної системи, MapReduce і розподілену файлову систему Hadoop (HDFS). У загальному пакеті Hadoop містяться файли та сценарії для Java-архіва (JAR), необхідні для запуску Hadoop. Для ефективного планування роботи кожна Hadoop-сумісна файлова система повинна забезпечувати інформацію про місцезнаходження – ім'я стійки (або, точніше, мережевого перемикача), де є робочий вузол. Програми Hadoop можуть використовувати цю інформацію для виконання коду на вузлі, де є дані, а у разі відсутності -– на тій же стійці / комутаторі, щоб зменшити магістральний трафік. HDFS використовує цей метод при реплікації даних для надлишковості даних на декількох стійках. Цей підхід зменшує наслідки відключення живлення або перемикання живлення в стойці; якщо відбудеться яке-небудь з цих апаратних збоїв, дані залишатимуться доступними. Багатокомпонентний кластер Hadoop Невеликий Hadoop кластер включає в себе єдиний майстер і кілька робочих вузлів. Головний вузол складається з Job Tracker, Task Tracker, NameNode і DataNode. Рабський або робочий вузол виступає як DataNode, так і TaskTracker, хоча це можливо лише для робочих вузлів лише для даних і обчислень. Вони зазвичай використовуються тільки в нестандартних програмах. Hadoop вимагає Java Runtime Environment (JRE) 1.6 або новішої версії. Стандартні скрипти запуску та завершення роботи вимагають встановлення безпечної оболонки (SSH) між вузлами кластера.

На рис. 1 можна побачити типову конфігурацію кластеру Hadoop що містить всі вищеописані компоненти.

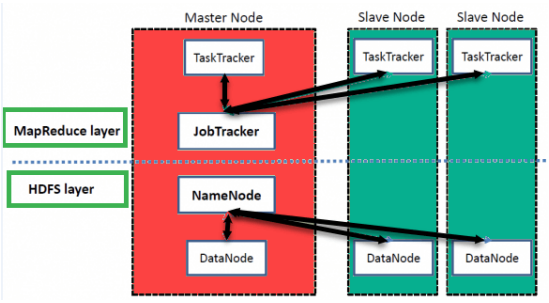


Рисунок 1 – Типовий кластер Hadoop

Алгоритми обробки природної мови

Також був проведений аналіз типових алгоритмів обробки натульної мови та обрані ті що відповідають поставленій задачі, такі як токенізація, лематизація, пошук n-грам. Саме ці алгоритми й використані для поставленої задачі – обробки повідомлень соціальних мереж.

Ці алгоритми є типовими, тому добре підходять для аналізу продуктивності програмних засобів.

Задачі обробки природної мови з кожним роком стають більш актуальними, а в поєднанні з інструментами BigData вони є ще більш ефективними для досягнення нових цілей у науці, бізнесі тощо.

Спочатку отримані повідомленя будуть фільтровані на рахунок слів сполучників. Потім текст буде поділений на токени, і будуть знайдені всі n-грами. Серед знайдених n-грам будуть знайдені

найпопулярніші словосполучення серед твітів користувачів соціальної мережі.

Обрані алгоритми є одними з типових для обробки природної мови, тому добре підходять для тестування ефективності програмних засобів.

Кінцевий алгоритм можна побачити на рис.2

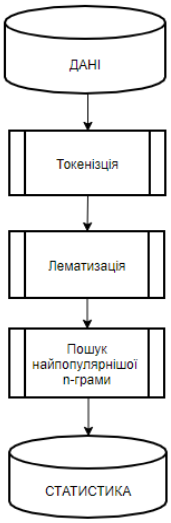


Рисунок 2 – Кінцевий алгоритм обробки даних

Розглянемо на прикладі речення “I am writing my master's thesis” алгоритм вище.

Першим етапом є токенізація. Під час токеніцації програмний засіб розділяє речення на об'єкти. Після першого етапи маємо такі вихідні дані, що можна побачити на рис. 3

Рисунок 3 – Результат токенізації

Другим етапом є лемматизація. В даному етапі кожен об'єкт приведений до нормальної форми. Результатом є Json леммами

```
{
  "lemma": {
    "I": 1,
    "be": 1,
    "writing": 1,
    "thesis": 1,
    "master": 1
  }
}
```

Третім етапом є пошук n-грам, та виявлення найбільш вживаних.

Результатом є таблиця з ними та частотою вживання в реченні.

На рис 4 можна побачити результат обробки заданого речення програмним додатком.

No.	N-gram	Frequency
1	i be	1
2	be writing	1
3	writing master	1
4	master thesis	1

Рисунок 4 – Результат пошуку n-грам

Хмарне середовище виконання

Хмарні обчислення - це надання обчислювальних служб (в тому числі серверів, сховища, баз даних, мереж, програмного забезпечення, аналітики та інтелектуального аналізу) через Інтернет ("хмара"). Такі служби прискорюють впровадження інновацій, підвищують гнучкість ресурсів і забезпечують економію завдяки високій масштабованості. Ви зазвичай платите тільки за хмарні служби, які дозволяють скоротити експлуатаційні витрати, а також підвищити ефективність управління інфраструктурою і масштабування в міру зміни потреб бізнесу.

Найбільш зручним та розвпосюдженним шляхом виконання розподілених обчислень є хмарні обчислення. Основними перевагами є:

- зниження вимог до обчислювальної потужності ПК (неодмінною умовою є тільки наявність доступу в інтернет);
- відмовостійкість;
- безпеку;
- висока швидкість обробки даних;
- зниження витрат на апаратне і програмне забезпечення, на обслуговування і електроенергію;
- економія дискового простору (і дані, і програми зберігаються в інтернеті).

Саме тому для виконання експерименті в даній роботі була обрана саме концепція хмарних обчислень.

Конфігурація кластерів

У кластері Hadoop є такі види служб:

1. NameNode – основний вузол, який веде облік кожного блоку кожного файлу, через який здійснюються всі операції, при відключенні якого стає недоступним весь HDFS;

2. SecondaryNameNode – додатковий вузол, який веде облік змін (на ділі не

дає відмовостійкості, тому що не замінює собою NameNode);

3. DataNode – основні вузли кластера Hadoop, що зберігають дані, які очікують команд від головного вузла Name Node;

4. NodeManager – вузол системи YARN, що знаходиться в парі з кожним вузлом DataNode, що виконує контроль над виконанням завдань з даними, що зберігаються на вузлі;

5.ResourceManager – основний вузол системи YARN, що запускає обчислювальні завдання.

Для створення і налаштування кластера необхідно виконати наступну послідовність дій:

1. Розподіл служб Hadoop по вузлах;
2. Установка віртуалізації LXC;
3. Створення контейнера;
4. Установка і налаштування Hadoop;
5. Клонування контейнера;
6. Запуск кластера

4.2. Виконання експериментів

На рисунку 5 зображена схема кластера Hadoop що був використаний для проведення експериментів в рамках даної роботи.

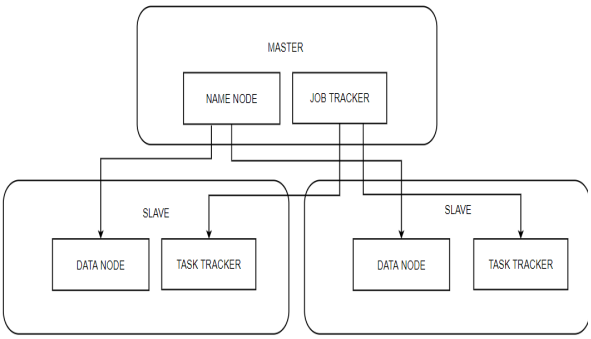


Рисунок 5 – Конфігурація Hadoop кластера

Spark запускається як масив незалежних процесів на кластері, координацією яких займається об'єкт SparkContext у вашій програмі (називається вона driver program).

У режимі кластера, SparkContext може працювати в парі з одним з декількох менеджерів кластера (власний менеджер менеджер кластера, Mesos або YARN), який виділяє ресурси для додатків. Одного разу приєднавшись, Spark отримує виконавців (executors) на ноді в кластері, які обробляють обчислення і зберігають дані для вашої програми. У підсумку, SparkContext відсилає завдання (tasks) до виконавців для обробки.

На рисунку 6 зображена схема кластера Hadoop що був використаний для проведення експериментів в рамках даної роботи.

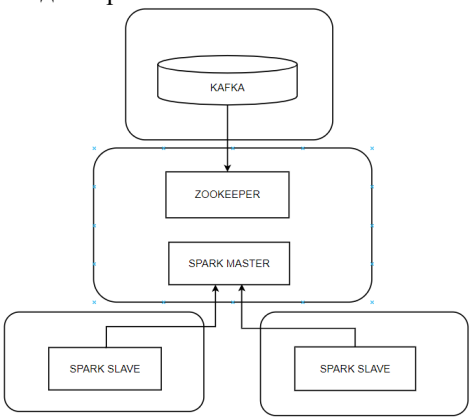


Рисунок 6 – Конфігурація кластера Spark

Для виконання експериментів використовуєсься база даних з текстовими даними, тому для імітації потококого надходження потрібно використати чергу повідомлень. В даному випадку був використаний програмний продукт Apache Kafka. На рис. 7 можна побачити схему імітації надходження потокових даних.

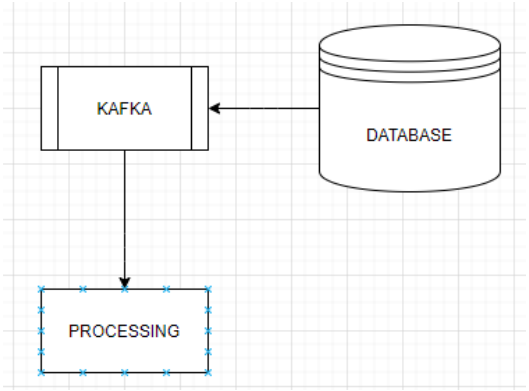


Рисунок 7 – Використання Apache Kafka

Для частини експериментив з обробки пакетами буда використана та сама база даних, проте дані були використані напряму, без брокера повідомлень, що можна побачити на рис. 8

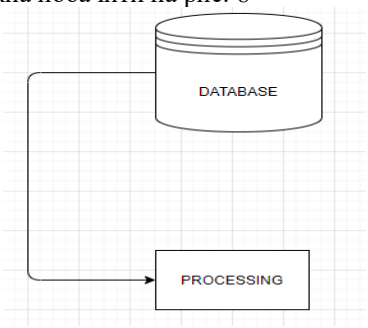


Рисунок 8 – Використання Apache Kafka

Тож, з конфігурацій двух кластерів можна зробити висновок що модель Master-Slave використовується в кластерах обох програмних продуктів.

Результати

Для обох експериментів вхідними параметрами для виконання обробки тексту буде тестова база що містить 25 мільйонів повідомлень користувачів соціальних мереж. Вихідними параметрами є список n кількості найпопулярніших n-грамм. Часовий проміжок виконання експериментів 2,5 години. Згідно ресурсу hootsuite.com саме цей період є середнім для відвідування середньостатистичного користувача сої мережі твіттер. Щодо пакетної обробки, загальний час обробки повідомлень також буде 2,5 години, але виконання обробки виконується пакетами даних що були зібрані за періоди 5, 10, 15, 30, 90, 120 хв. щоб можливо було оцінити залежність продуктивності інструменту від розміру пакетів.

Попередньо було виявлено що порядок виконання експериментів не важливий. Тому вони будуь виконані у такому порядку:

Обробка повідомлень у Spark з t2.micro за 2.5 години

Обробка повідомлень у Spark з t2.medium за 2.5 години

Обробка повідомлень у Spark з t2.micro за 2.5 години пакетами

Обробка повідомлень у Spark з t2.micro за 2.5 години пакетами

Результати будуть приведені у вигляді таблиць та графіків.

Apache Spark

Згідно плану експерименті було виконано по 10 вимірів для кожного.

У діаграмі на рис.9 приведені результати.

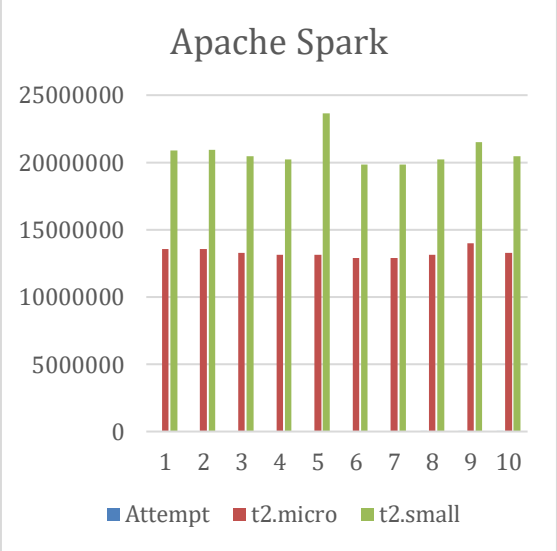


Рисунок 9 – Результати Apache Spark

Hadoop

Згідно з планом експериментів виміри Apache Hadoop будуть проведені в декілька етапів з різним періодом збору даних, але сумарним часом 2,5 години.

Для кожного періоду проведено по три

Номер експеримента	1	2	3
t2.micro	5803452	5803768	5803011
t2.small	6847783	6848233	6848681

експерименти для виключення аномальних значень, якщо такі будуть присутні.

В рамках проведення даних експериментыв аномальними значеннями ваажаємо результати з відхиленям від середнього більше 10 відсотків. Такого показника достатньо щоб виключити ті результати що будуть не дійсними через причини нестабільності середовища, зв'язку, тощо.

У таблиці 1 приведені результаті виміру з найбільшими пакетами у 150 хв.

Таблиця 1 – Результати вимірів

У діаграмі на рис.10 приведені результати.

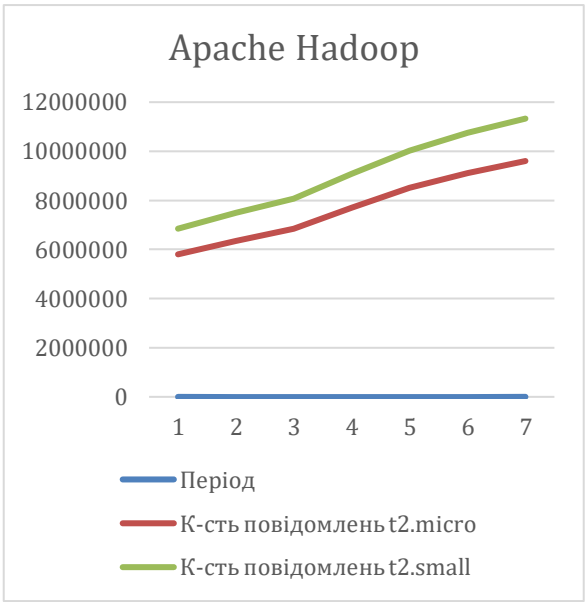


Рисунок 10 – Результати Apache Hadoop

Згідно з результатами проведених експериментів для задач обробки невеликих повідомлень використовуючи типові алгоритми NLP більш ефективним виявився Apache Spark. В середньому за 2.5 години інструмент потокової обробки даних обробив на 40 відсотків більше повідомлень ніж інструмент пакетної обробки. Аномальних значень під час виконання дослідів не виявлено.

При дослідженні продуктивності Apache Hadoop було проведено досліди з різним періодом збору інформації, таким чином було виявлено що за рахунок збільшення розміру пакету, збільшується продуктивність даного програмного засобу.

Опираючись на результати експериментів можна сказати що Apache Spark має меньшу дисперсію результатів (до 6%) ніж Apache Hadoop (до 10%). Дані досліди не були спрямовані на зясування стабільності, але опираючись на результати можна сказати що для даної задачі засоб потокової обробки виявився більш стабільним.

Також експерименти були проведені з використанням двох типів апаратного забезпечення. Кластери були сконфігуровані з використанням

хмарної платформи AWS, а саме EC2 машини типів t2.micro і t2.small.

Порівняльна характеристика програмних засобів для обробки повідомлень

В ході виконання роботи було виконане дослідження сучаних засобів та підходів обробки BigData для задач аналізу текстових даних з соціальних мереж. Були порівняні дві основні моделі – потокова обробка та обробка пакетами.

В якості програмного засобу для обробки пакетами був обраний Apache Hadoop, що реалізує MapReduce.

Для потокової обробки був використаних Apache Spark.

Найбільш ефективним засобом для даної задачі виявився Spark і потокова обробка даних.

Результати експериментів показали що стабільніше працює Hadoop. Через використання оперативної памяті Spark показав відхилення від середнього результату до 6%.

Під час проведення експериментів ні одна з систем не мала перебоїв. Hadoop має високу стійкість до перебоїв за рахунок файлової системи HDFS та реплікацій. Spark за рахунок розподілених датасетів RDD.

В рамках проведених експериментів було сконфігуровано по 2 кластери для кожної з тестуємих моделей обробки даних. Використана була найпопулярніша платформа Amazon Web Services.

В результаті експериментів було виявлено що за рахунок збільшення оперативної памяті в 2 рази Apache Spark став ефективніший в середньому на 55%, проти 18% у Hadoop.

За рахунок використання оперативної памяті потокова модель обробки даних дорожча.В рамках проведеного експерименту було виявлено що різниця у вартості кластеру AWS була ~20% за рахунок використання ресурсів.

Висновки

В ході роботи було проведено дослідження популярності програмних засобів для обробки великих даних, та виявлені основні підхои. В ході експериментів порівнювались Apache Spark, як представник потоковоъ обробки і Apache Hadoop як представник пакетної.

Тож основним висновком є те що для обробки статичних текстових даних, великими пакетами є більш привабливим Apache Hadoop за рахунок невеликої вартості і достатньої ефективності в даному випадку. Але для обробки даних з соціальних мереж, які постійно оновлюються і додаються потокова обробка є більш ефетивною разом з Apache Spark.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Акатова Н.А. Комплексность проектирования интегрированных информационных систем административно-территориального и муниципального управления [Текст] / Акатова Н.А., Черкасов Ю.М. // Вестник Университета: серия «Информационные системы управления». –, 2000. – № 1. – С.23-35.
2. Большие данные [Електронний ресурс] Режим доступу: http://sewiki.ru/ index.php?title=Большие_данные&oldid=3075
3. Большие_данные (Big_Data) ? [Електронний ресурс] Режим доступу: http://www.tadviser.ru/index.php/Статья:Большие_данные (Big_Data)
4. Большие_данные [Електронний ресурс] Режим доступу: https://ru.wikipedia. org/wiki/Большие_данные
5. Большие_данные_(Big_Data) [Електронний ресурс] Режим доступу: http://www.tadviser.ru/index.php/Статья:Большие_данные_(Big_Data)

6. Бутакова М.А. Мера информационного подобия для анализа слабоструктурированной информации [Електронний ресурс] / Бутакова М.А., Климанская Е.В., Янц В.И. // Современные проблемы науки и образования. – 2013. – № 6. – Режим доступу: http://www.science-educa- tion.ru/113-11307.
7. Горчаков, А. Математичний апарат для інвестора. Аналіз та прогнозування часових рядів [Текст] / А. Горчаков // Фінансовий ринок України. – 2007. – № 12. – С. 38–42.