НЕЙРОІНФОРМАТИКА ТА ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ

НЕЙРОИНФОРМАТИКА И ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

NEUROINFORMATICS AND INTELLIGENT SYSTEMS

УДК 004.021

Бойко Н. І.

Канд. екон. наук, доцент, доцент кафедри Інформаційних систем та мереж, Національний університет «Львівська політехніка», Львів, Україна

ПЕРСПЕКТИВНІ ТЕХНОЛОГІЇ ДОСЛІДЖЕННЯ ВЕЛИКИХ ДАНИХ У РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

Актуальність. Розглянуто питання коректної інтерпретації інформаційних потоків у розподілених інформаційних системах. Об'єктом дослідження є методи дослідження просування «великих даних» по кластерах системи.

Мета роботи є дослідження перспективних напрямків та технологій для аналізу структур даних у розподілених інформаційних системах.

Метод. Розглянуто технології обробки великих даних. Проведено аналіз кожної з них. Наведено приклад застосування парадигми MapReduce, завантаження великих обсягів даних на сервер, опрацювання та аналіз неструктурованої інформації та розподілення її у кластеризовану базу даних. В статті узагальнено поняття "великі дані". Наводяться приклади методів по роботі з масивами неструктурованих даних. Виділені наукові спрямування для аналізу великих даних. Сформульовані принципи роботи неструктурованих даних у розподілений інформаційних системах. Приводиться робота платформ Hadoop MapReduce та Apache Spark. Аналізуються їхні властивості та приводяться відмінності. Наводиться порівняльний аналіз продуктивності обох платформ у відношенні – час виконання до кількості ітерацій. Розглядаються способи створення RDD: розпаралелення переданої колекції в програмі та посилання на зовнішню файлову систему в Hadoop. Також наводиться приклад розпаралеленої системи RDD. Пропонується робота класу одинак для основних операцій з базою даних: підключення до бази даних, створення таблиці, знищення таблиці, отримання рядка по id, повернення усіх елементів бази даних, оновлення, видалення та створення рядка.

Результати. Проведений аналіз моделей Spark та Hadoop MapReduce для поеапної побудови розподіленої інформаційної системи. Побудований SparkConf об'єкт, який містить інформацію про аплікацію і є кінцевим варіантом експеременту.

Висновки. Проведені експерименти підтвердили працездатність запропонованих методів, які здатні обробляти горизонтальні масиви даних, що розпаралелені через неякісний спосіб представлення інформації. Такі перспективні напрямки роботи аналізують структуру даних з метою прогнозу результатів та створюють алгоритми передових кореляцій, що сприяють новому розумінню діяльності розподілених інформаційних систем. Подальші дослідження можуть полягати в широкому застосуванні інформаційних систем, які би забезпечували повний комплекс технологічного процесу адаптації інформаційних потоків у кластери.

Ключові слова: система, технологія, великі дані, інформація, методика, база даних, веб-аплікація, моделювання, обробка, аналіз.

НОМЕНКЛАТУРА

IT – інформаційні технології;
SN-архітектура – Shared Nothing Architecture;
NoSQL – Not only Structured query language;
БД – база даних;
СКБД – Система керування базами даних;
GPS – Global Positioning System;
API – Application Program Interface;
RDD – Resilient Distributed Datasets;
UI – User Interface.

ВСТУП

Останніми роками перспективною областю наукових досліджень є область IT. Донедавна великі системи складалися із десятків серверів та терабайтів інформації. Сьогодні вони використовують хмаркову кластерну модель, яка складається із тисячі мульти-ядерних процесорів та петабайтів інформації.

Такий розвиток подій сприяв створенню нового наукового напряму – «великі дані», які вже знайшли своє відображення в академічних вузівських програмах. З позиції наукових досліджень до «великих даних» відносять інформацію: структуровану чи неструктуровану дані, медіа та випадкові процеси, які на практиці важко обробити традиційними методами. На зміну традиційним монолітним системам приходять нові асинхронні та паралельні рішення, які спрощують роботу з «великими даними». Дані системи складаються з декількох незалеж-

DOI 10.15588/1607-3274-2017-4-8

них блоків, які ефективно обробляють інформацію в умовах її неперервного накопичення та відповідного її розподілення по численних вузлах кластера. В таких системах, обсяги інформації зростають за експоненціальним законом і значну їх частину складають неструктуровані дані. Тому питання коректної інтерпретації інформаційних потоків у системах такого типу являються на сьогодні актуальними та водночас складними.

Отже, процес дослідження просування «великих даних» по кластерних системах являється об'єктом даного дослідження.

Предметом дослідження є методи та засоби побудови, редагування та адаптації інформаційних потоків у розподілених інформаційних системах.

Метою роботи є дослідження перспективних напрямків та технологій для аналізу структур даних у розподілених інформаційних системах.

1 ПОСТАНОВКА ЗАДАЧІ:

1) розглянути методи та принципи роботи з «великими даними»;

2) проаналізувати існуючі технології обробки «великих даних»;

3) здійснити порівняльний аналіз продуктивності методів Hadoop та Spark платформ для обробки неструктурованих обсягів даних на мові Scala.

2 ОГЛЯД ЛІТЕРАТУРИ

Сьогодення диктує нові вимоги до інформаційних систем, які важко досягнути вчорашніми технологіями. Сучасні системи повинні реагувати на проблеми вчасно наскільки це можливо. Дослідження відомих іноземних вчених [1–5] показали важливість даної проблеми, яка повинна бути вирішуватись швидко та ефективно. У роботах [4, 6] акцентується увага на стійкості системи до збоїв та можливості її залишатися реагувати навіть після збою.

Дослідниками [8, 10, 14] являється актуальним питання масштабування кластера, адже кластерні системи вважаються доволі складним. Тому в будь-яких системах є перспективним питання її продуктивності, тому автори [13, 15] виокремлюються два варіанти масштабування: горизонтальне та вертикальне. Вони дослідили, що при застосуванні горизонтальної кластеризації одні компоненти повинні бути ізольованими від інших, і при цьому система повинна реагувати при різних навантаженнях на кластери. Науковці досліджували її здатність збільшувати або зменшувати обсяг ресурсів, який виділяється для її обслуговування. Дослідники вважали, що система повинна покладатися на асинхронну передачу повідомлень, для встановлення розмежування між компонентами. Тим самим вона гарантує слабозв'язаність, ізоляцію та прозорість розташування вузлів кластера, забезпечує засоби для делегування помилок та повідомлень. Іншими авторами [10–12] виявлено, що система повинна управляти навантаженнями, еластичністю, управляти потоками, формувати та моніторити черги повідомлень в системі і застосовувати зворотний тиск при необхідності. Вона повинна мати номери для блокування зв'язку, що дозволить одержувачам споживати тільки активні ресурси, що призведе до менших витрат системи.

Науковцями [10–12] встановлено, що при використанні даного типу систем виникає деяка проблема взаємодії всіх вузлів кластерної системи. До прикладу різним додаткам потрібен доступ до даних з різних вузлів. Це ускладнює роботу кластерної системи, але існує можливість вертикального масштабування даних, що забезпечує доступ до даних всіх вузлів системи. Дослідниками [8] виокремлено, що запропонований вище процес вертикального масштабування інформації дозволяє відділяти кластер від систем, які з'єднують між собою безліч машин.

На основі проведеного аналізу встановлено, що для створення розподіленої інформаційної системи не існує єдиних методів та технологій, які б поєднували всі етапи побудови кластерних систем. Тому робота з детальним описом технологій вертикального масштабування інформації для вирішення такого роду проблем є актуальною.

3 МАТЕРІАЛИ І МЕТОДИ

Існує багато визначень стосовно того, що таке «великі дані». Серед них такі: «великі дані» – це петабайти інформації, які неможливо обробити в MS Excel; також, це дані, які неможливо обробити на одному комп'ютері; «великі дані» – це будь-яка структурована чи неструктурована інформація великих обсягів [1–3].

Узагальнюючи подані визначення, можна виокремити, що «великі дані» – це не самі дані, а методи їх опрацювання, які дозволяють розподілено обробити інформацію. Ці методи можна застосувати до масивів будь-якої розмірності. У роботі наводяться декілька прикладів джерел утворення великих даних та методи, які призначені для роботи з ними: дані про поведінку користувачів в Інтернеті; GPS – сигнали; дані, зібрані з лічильників; дані, зняті із камер спостереження; оцифровані книги із бібліотек; інформація про транзакції усіх клієнтів банку тощо.

З перелічених прикладів можна виділити те, що кількість інформаційних джерел стрімко зростає, відповідно технології їх обробки та методи аналізу великих даних набирають великої популярності.

Тому, серед основних наукових спрямувань для аналізу великих даних слід виділити:

– Data Mining: кластерний аналіз.

– Краудсорсинг: передача певних виробничих функцій невизначеному колу осіб.

– Машинне навчання: використання моделей на базі машинного навчання для отримання комплексних прогнозів на основі базових моделей.

– Імітаційне моделювання: метод, що дозволяє будувати моделі процесів, які описують дійсні інформаційні процеси.

– Візуалізація даних: інтерактивне вивчення візуального представлення абстрактних даних для посилення людського пізнання.

– Штучні нейронні мережі: моделі, побудовані за принципом функціонування біологічних нейронних мереж.

Проаналізувавши діяльність дослідників [1–10] в галузі великих даних, можна сформулювати певні принципи роботи із ними. Серед таких можна виділити:

– Горизонтальне масштабування, коли зростання обсягу даних призводить до розширення системи та збільшення кількості технічних засобів у кластері.

– Відмовостійкіть. При горизонтальному масштабуванні машин в кластері може бути багато. Для прикладу, Hadoop-кластер компанії Yahoo має більше ніж 42000 машин у кластері. Така кількість машин не може безперебійно працювати. Даний принцип оперується методами обробки великих даних, які враховують можливість збоїв та попереджують їх.

– Локальність даних. У розподілених системах дані фізично знаходяться на одному сервері, а обробляються на іншому, тим самим збільшуючи витрати ресурсів системи на передачу даних – до витрат на їх обробку.

Технологічний аспект обробки великих даних повинен слідувати згаданим вище принципам.

Найчастіше в якості базового принципу обробки великих даних вказують SN-архітектуру, яка забезпечує паралельну обробку, масштабовану без деградації на сотні й тисячі вузлів кластера. До основних її технології обробки «великих даних» відносять:

– NoSQL;
– MapReduce;
– Apache Hadoop;
– Apache Spark.

Збільшення обсягів інформації призводить до певних проблем, з якими класичні реляційні архітектури не справляються. Тому виникла потреба спроектувати архітектуру NoSQL, яка здатна адаптуватися до зростання даних та ефективної їх обробки.

До особливостей NoSQL підходу слід віднести:

– високу пропускну здатність;
– необмежене горизонтальне масштабування;
– консистентність у жертву продуктивності.

До класифікаційних ознак NoSQL БД слід віднести:

1. Сховища типу ключ-значення. Модель даних – асоціативний масив або словник, який дозволяє працювати з даними по ключу. Основне завдання таких сховищ – максимальна продуктивність системи, яка не зберігає інформацію про структуру значення. До прикладів таких баз даних можна віднести: Redis, Scalaris, Riak, Tokyo Tyrant.

2. Документні сховища. Модель даних об'єднує множину пар ключ-значення в абстракцію, яку називають «документ». Документи мають вкладену структуру, що об'єднуються в колекції. Робота з документами проводиться по ключу, також існують рішення, які дозволяють виконувати запити по значенням атрибутів. До прикладів даних БД відносять: SimpleDb, CouchDb, MongoDb.

3. Стовпцеві сховища. Модель даних – збереження значень, які не інтерпретовані в байтових масивах та адресуються кортежами. Основою такої моделі даних є стовпчик чи необмежене число стовпців для певної таблиці. Стовпці по ключам об'єднуються у сімейства та набувають властивості певних наборів. До таких БД слід віднести: BigTable, HBase, Cassandra.

4. Графові сховища. Модель даних складається із вершин, ребер і їх властивостей. Робота з даними виконується шляхом обходу графа по ребрах із заданими властивостями. Подібні сховища застосовуються для роботи з даними типу графів (наприклад, соціальна мережа). Прикладом може бути БД Neo4j.

Також існують класичні методи для розробки засобів оброблення неструктурованих даних. Однією із таких – парадигма ***MapReduce.***

Дана модель розподіленої оброблення даних, запропонована компанією Google для обробки надвеликих обсягів даних на комп'ютерних кластерах (рис. 1). Кластер – це декілька незалежних обчислювальних машин, які використовуються спільно та працюють як одна система [5].

Метод MapReduce передбачає організацію даних у вигляді списків, обробка яких відбувається у 3 етапи (рис. 2):

1. Стадія Map, на якій дані обробляються за допомогою функції map(), яку визначає користувач. Робота на цій стадії полягає у переробці та фільтрації даних. Робота операції схожа на метод map() у функціональних мовах програмування – функція застосовується до кожного елементу списку. Функція map приймає список на вході і повертає множину пар ключ-значення (key-value).

2. Стадія Shuffle, на якій функція map «розбирається по корзинам» – кожна корзина відповідає одному ключу стадії map. В подальшому ці корзини на вході послужать функції reduce().

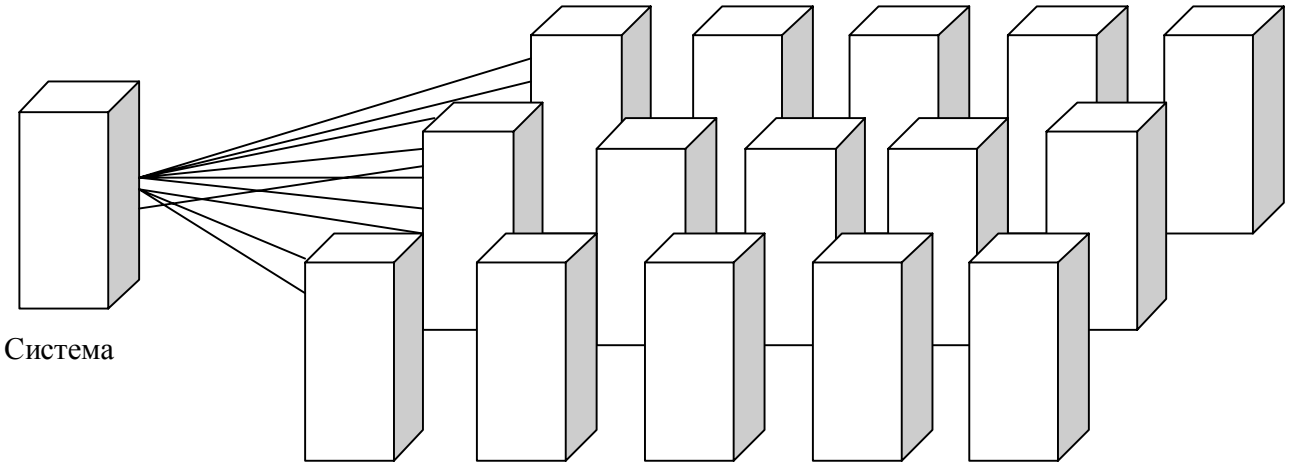

Рисунок 1 – Кластерна модель

3. Стадія Reduce. Функція reduce визначає фінальний результат для окремої «корзини». Множина усіх значень, які повертає функція reduce() є фінальним результатом MapReduce – задачі.

Усі запуски функцій map(), reduce() і shuffle(), що наведені на рис. 2, працюють незалежно та можуть обробляти інформацію паралельно на різних машинах кластера. Така робота методу MapReduce дозволяє виконувати принцип горизонтального масштабування.

На сьогодні, лідером у застосуванні парадигми MapReduce та створенні програмної платформи для організації розподіленої обробки великих обсягів даних є технології Apache Hadoop MapReduce та Apache Spark.

Apache Hadoop MapReduce – це вільна платформа, для організації обробки великих обсягів даних (виміряється у петабайтах) з використанням парадигми MapReduce (рис. 3). Даний метод дозволяє здійснювати поділ відособлених фрагментів, кожен з яких може бути запущений на окремому вузлі кластера. До складу Hadoop входить реалізація розподіленої файлової системи Hadoop HDFS, яка автоматично забезпечує резервування даних та є оптимізованою для роботи з MapReduce. Для спрощення доступу до даних в сховищі Hadoop розроблена SQL-подібна мова Hive, яка є свого роду SQL для MapReduce, і запити якої можуть бути розпаралелені й оброблені кількома Hadoop-платформами.

Apache Spark – це високопродуктивний технічний засіб для обробки даних, які зберігаються в кластері Hadoop (рис. 4). Порівняно з попереднім Hadoop MapReduce, Spark забезпечує в 100 разів більшу продуктивність при обробці даних у пам'яті та у 10 разів більшу – при розміщенні даних на дисках. Даний механізм виконується на вузлах кластера Hadoop за допомогою Hadoop YARN, та у відокремленому режимі. У ньому підтримується обробка даних у сховищах HDFS, Cassandra, Hive та у будь-якому форматі введення Hadoop. У жовтні 2014 року Apache Spark встановив світовий рекорд при сортуванні 100 терабайт даних.

Основна відмінність моделі Spark від Hadoop MapReduce (рис. 3–4) у тому, що Spark зберігає інформацію на пам'ять комп'ютера, тим самим забезпечує вищу продуктивність платформи. В той час як Hadoop зберігає її на диску, забезпечуючи вищий рівень безпеки.

Окрім традиційних можливостей моделі Apache Hadoop MapReduce, тобто обробки неструктурованих обсягів даних, Apache Spark платформа включає в себе Spark Streaming для роботи з асинхронними стрімами, Mlib – бібліотеку для машинного аналізу та GraphX (рис. 5).

Модуль MLib (рис. 5) включає в себе алгоритми машинного навчання (класифікація, регресія, кластеризація, колаборативна фільтрація і т.д.). Перевагою використання технології Spark, модуля MLib для машинного навчання є те, що в Spark дані можна кешувати в оперативній пам'яті. Це дозволяє істотно прискорити обчислення для ітеративних алгоритмів, якими є більшість алгоритмів машинного навчання.

GraphX – інструмент для аналізу та обробки графів (рис. 5). Концепція графа реалізована у вигляді так званого Property Graph – це мультиграф з мітками на вершинах і ребрах. Мультиграф – це орієнтований граф, в якому дозволені кратні ребра та петлі. Тут визначені такі поняття, як вхідна степінь (число вхідних ребер) та вихідна степінь (число вихідних ребер).

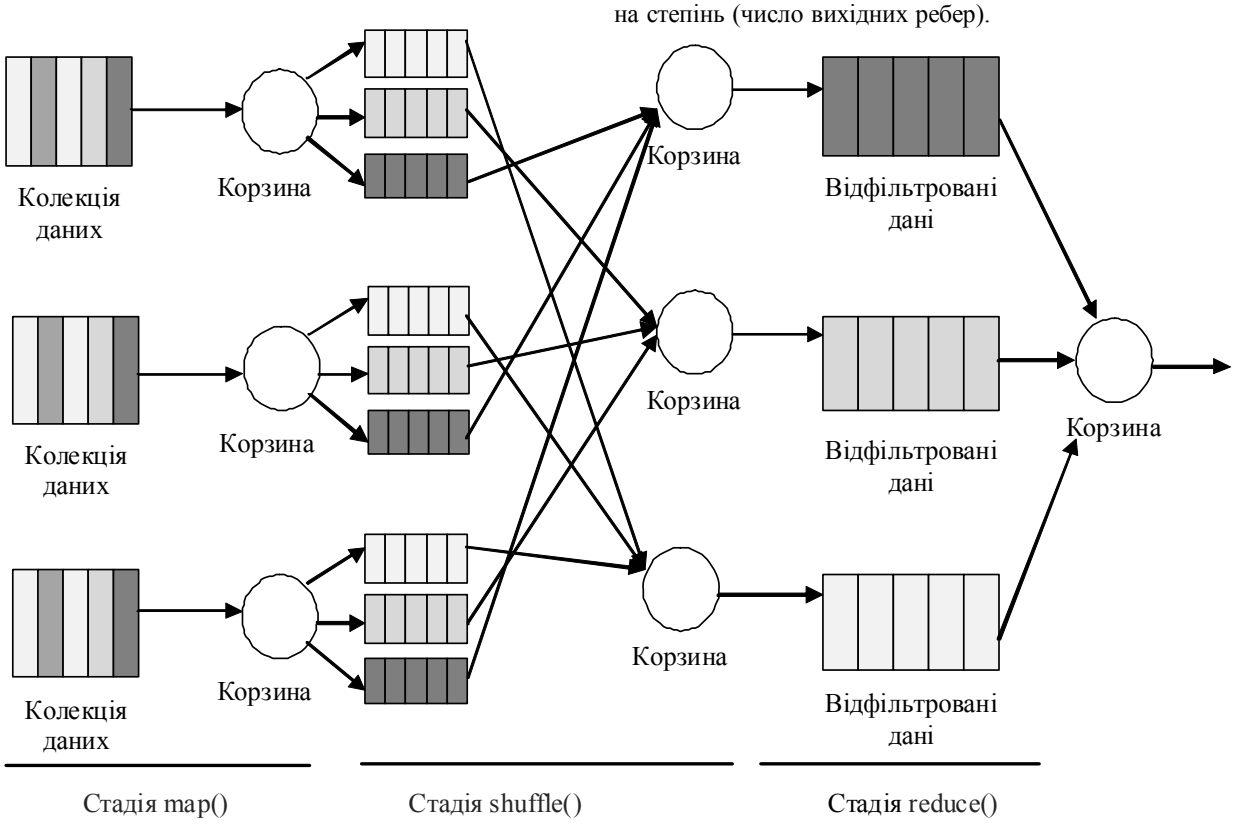

Рисунок 2 – Ілюстрація роботи MapReduce

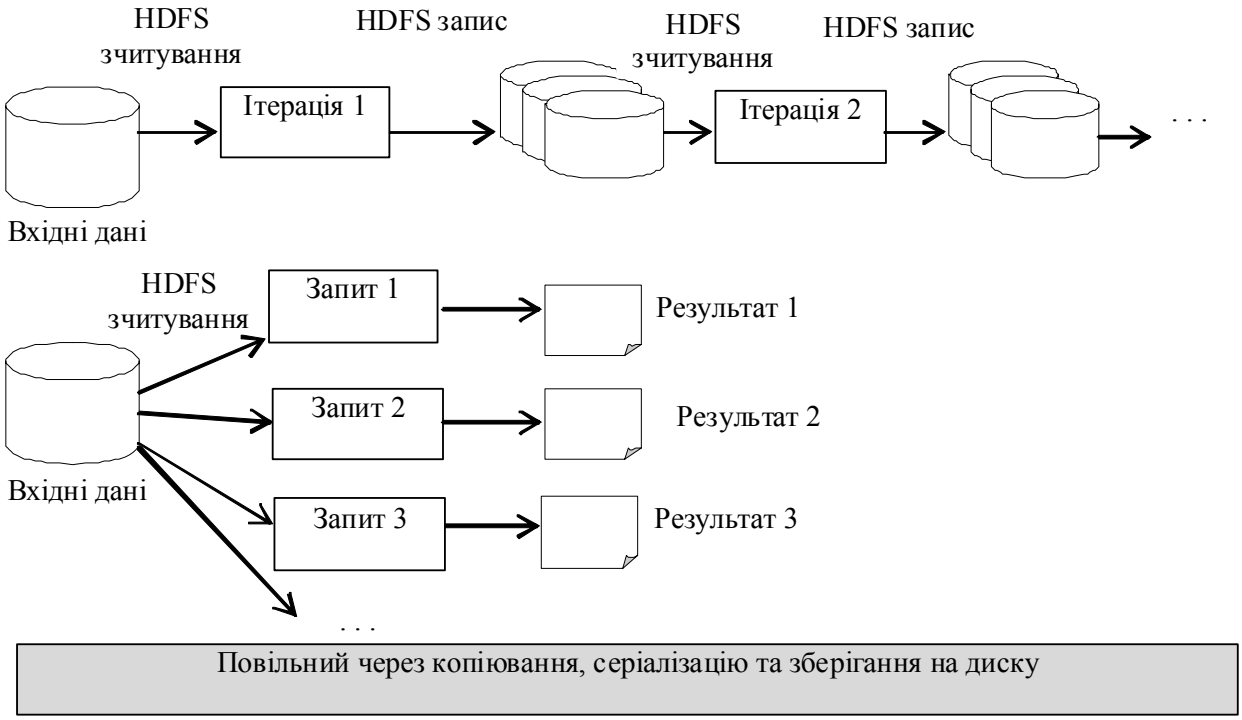

Рисунок 3 – Схема обробки даних у MapReduce

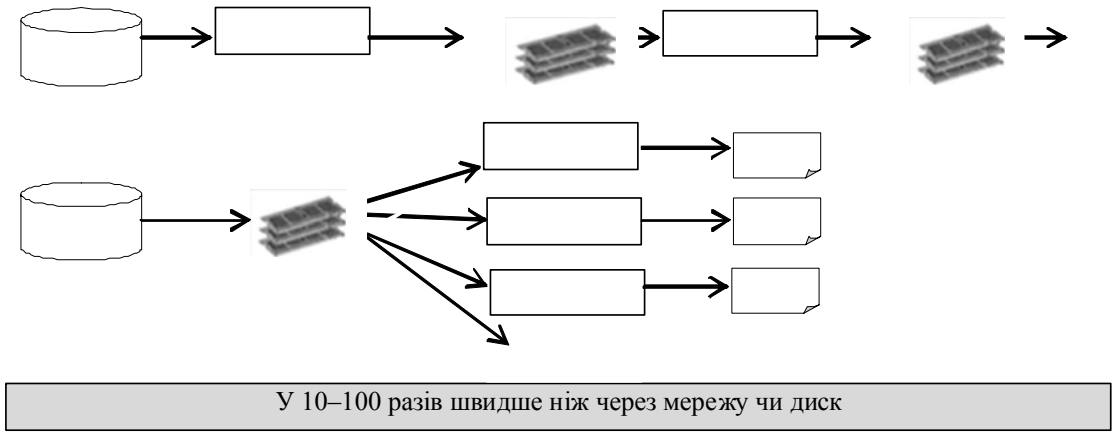

Рисунок 4 – Схема обробки даних у Spark

Spark SQL
SQL та обробка неструктурованих даних
MLlib
Алгоритм машинного навчання
GraphX
Обробка графів
Spark Streaming
Обробка стрім потоків

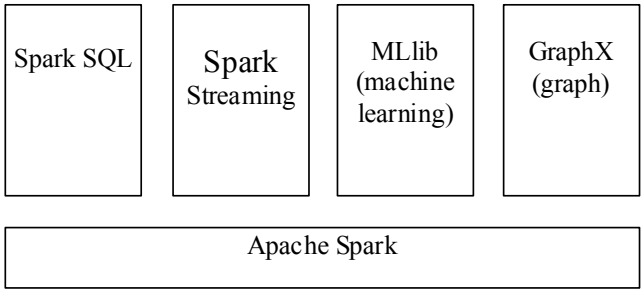

Рисунок 5 – Модулі Spark

Spark Streaming – це один із найцікавіших компонентів Spark. Із Spark Streaming можна створювати канали даних, які обробляються у реальному часі (рис. 5). Крім того, він забезпечує стійкість до збоїв, яка необхідна при роботі з real-time даними.

Нижче наводиться порівняльний аналіз продуктивності обох платформ у відношенні – час виконання до кількості ітерацій (рис. 6).

Spark надає API на Scala, Java, Python, R мовах програмування.

4 ЕКСПЕРЕМЕНТИ

Проведені дослідження методів Spark та Hadoop MapReduce дозволяють зробити наступні висновки, що Spark є продуктивнішою технологією (рис. 6). Адже програма спочатку створює SparkContext об'єкт, який вказує Spark метод доступу до кластера. На основі запропонованих вище методів, для створення SparkContext слід здійснити експеримент – побудувати SparkConf об'єкт, який містить інформацію про аплікацію.

Для детального аналізу слід привести приклад створення SparkContext на Scala (рис. 7):

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
val conf = new SparkConf().
setAppName(appName).setMaster(master)
new SparkContext(conf)
```

Основою Spark є концепція пружного розподіленого набору – RDD. Це відмовостійка колекція елементів, яка

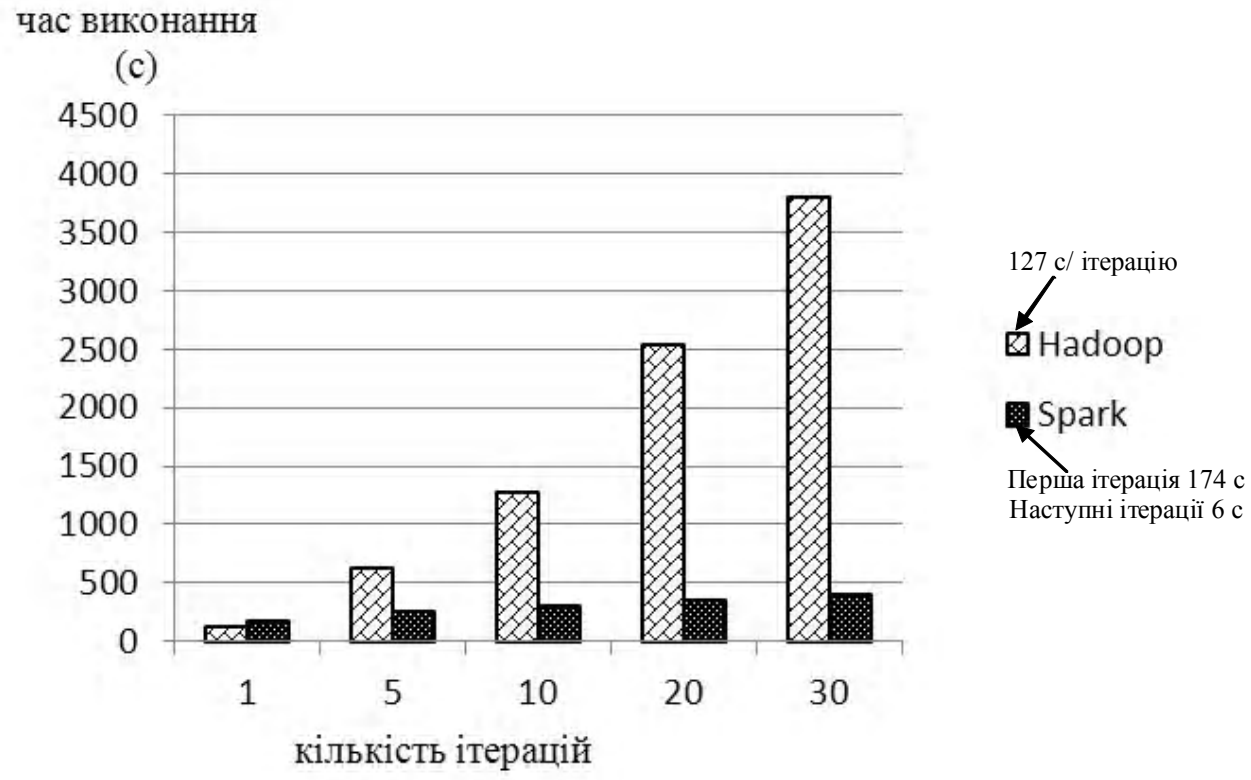

Рисунок 6 – Порівняльний аналіз продуктивності Hadoop та Spark платформ

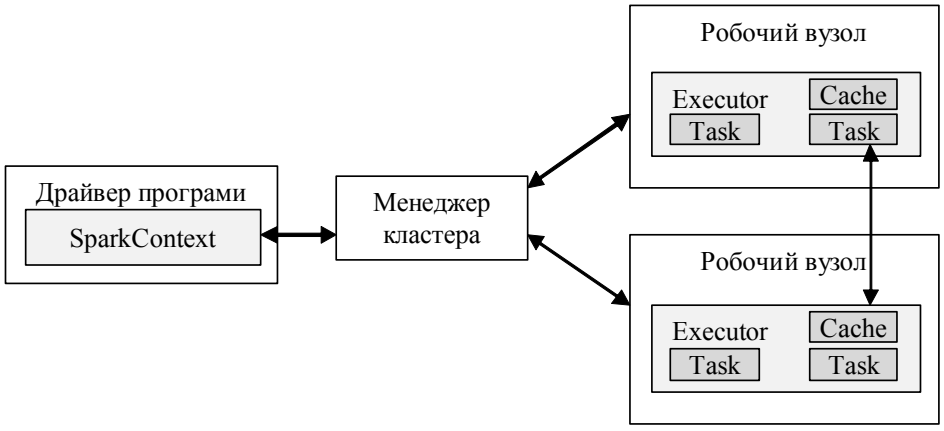

Рисунок 7 – Приклад створення Spark Context

опрацьовується паралельно. Є два способи створення RDD: розпаралелення переданої колекції в програмі та посилання на зовнішню файлову систему, таку як HDFS або будь-яке інше джерело даних в Hadoop.

Для прикладу, в даному розділі аналізується розпаралелена система RDD. Вона повинна приймати csv або exel документ із надвеликою кількістю інформаційних рядків про транзакції. Для цього слід завантажити документ на сервер, передати у Spark RDD, опрацювати та проаналізувати інформацію та розподілити у кластеризовану базу даних.

Для проведення експеременту потрібно розділити структуру сервісу на дві частини: перша – це веб-сторінка, на якій буде UI з формою для відправлення документу на сервер та інтерфейсом із аналізом даних після отримання оброблених даних із сервера; друга – це API нашої системи, яка буде представляти бібліотеку методів для прийому, обробки, аналізу і відправки даних клієнту (нашу веб-сторінку).

Тут слід зосередити увагу на API системи при застосуванні Apache Spark. Приклад такої системи наведений на мові Scala. Для початку, слід задати налаштування конфігурації кластера та створити SparkContext.

```
import org.apache.spark.{SparkContext, SparkConf}
object SparkUtil {
lazy val sparkConf = new SparkConf().setMaster("local[*]").setAppName("spark-app")
lazy val sc = new SparkContext(sparkConf)
```

В коді master URL – це налаштування конфігурації кластера; setMaster("local[*]") – означає запуск Spark локально з визначеним числом інформаційних потоків відповідно до кількості ядер на певній машині; setMaster(spark://HOST:PORT) – конфігурація для з'єднання із зовнішнім кластером.

Також в коді master URL слід застосувати метод, для отримання файлу від клієнта, здійснити перевірку: чи тип файлу відповідає очікуванням (csv або xlsx). Якщо так, тоді файл буде завантажено на сервер та передано його назву в метод parseAttachment(inputFile: String). Інакше – метод поверне попередження (warning).

```
//метод отримання файлу від клієнта
def file = Action(parse.multipartFormData) { request =>
   request.body.file("file").map { file =>
   val filename: String = file.filename
     val contentType : Option[String] = file.contentType
   contentType match {
      case Some("text/csv") => SparkUtil.parseAttachment(filename)
     case Some("application/vnd.openxmlformats-officedocument.spreadsheetml.sheet") =>
       => SparkUtil.parseAttachment(filename)
     case _ => SparkUtil.warning = true;
   }
}
```

На наступному кроці експерименту, кожен елемент файлу передається в конструктор CSVReader, його слід розпарсити та повернути сирий контент у Spark RDD. Цей процес описує метод розпаралелення надісланого файлу та дозволяє розпаралелити опрацювання даних. Далі повертається колекція (list), яка передається в конструктор toTransactions(data), таким чином повертається колекція з транзакції. Після цього процесу, кожен елемент колекції передається у DAO.create(), тобто зберігається у базі даних.

```
//метод розпаралелення отриманого файлу

def parseAttachment(inputFile: String): Unit = {
  val input = sc.wholeTextFiles(inputFile) //наш RDD
    val result = input.flatMap {
       case (_, txt) =>
         val reader = new CSVReader (new StringReader (txt) )
       reader.readAll ()
} //розпарсити csv файл

//розпаралелити розпарсений RDD і повернути колекцію(list)
val data = result.collect().map(_.mkString(",")).toList.tail
val importedData = toTransactions(data)
val importedCount = importedData.size
//зберегти в базу даних кожен елемент колекції
importedData.foreach(U => DAO.create(U))
   }
}
```

Наступним етапом проведення експерименту є застосування класу одинака в коді master URL для здійснення основних операцій з базою даних: підключення до бази даних, створення таблиці, знищення таблиці, отримання рядка по id, повернення усіх елементів бази даних, оновлення, видалення та створення рядка.

```
// клас одинак (scala object) для основних операцій з базою даних
object DAO extends DbSupport {
  class TransactionTable(tag: Tag) extends Table[Transaction](tag, "TRANSACTIONS") {
   def id: Rep[Long] = column[Long]("id", O.AutoInc, O.PrimaryKey, O.SqlType("BIGINT"))
     def account: Rep[String] = column[String]("account", O.SqlType("TEXT"))
    def description: Rep[String] = column[String]("description", O.SqlType("TEXT"))
    def currency_code: Rep[String] = column[String]("currency_code",O.SqlType("VARCHAR(255)"))
     def amount: Rep[BigDecimal] = column[BigDecimal]("amount", O.SqlType("DECIMAL"))
   def * = (id, account, description, currency_code, amount) <> (Transaction.tupled, Transaction.unapply)
}
val transactions = TableQuery[TransactionTable]
def queryById = Compiled(
     (id: Rep[Long]) => transactions.filter(_.id === id))
```

```
def createTable =
      transactions.schema.create
def dropTable =
      transactions.schema.drop
def setup = DBIO.seq(
      createTable)
val runDb = db.run(setup)
def all = {
      db.run(transactions.result)
}
def update(id:Long, transaction : Transaction)
= {

db.run(queryById(id).update(transaction))
}
def delete(id:Long) = {
      db.run(queryById(id).delete)
}
def  create(transaction:Transaction):
Future[Int] = {
      db.run(
           transactions += transaction)}}
```

Наступним етапом експерименту являється створення класу транзакцій, який має 4 поля: id, акаунт(account), опис(desc), код(code) і кількість(amount).

```
//клас транзакцій
case class Transaction(id: Long, account: String, desc:
String, code: String, amount: BigDecimal)
```

Для подальшої роботи, потрібно створити трейт для налаштування бази даних, підключення, створення сесії та її запуску.

```
//скала трейт для запуску бази даних
trait DbSupport {
    val db = Database.forConfig("db")
       implicit val session: Session =
db.createSession()
    def startDb() = {
      DAO.runDb }
```

Після налаштування БД, наступним етапом є створення актора для асинхронної передачі повідомлень, для того, щоб встановити межу між компонентами. Вона гарантує слабозв'язаність, ізоляцію, прозорість розташування та забезпечує засоби для делегування помилок чи повідомлень.

```
import akka.actor.Actor
import models.{Transaction, DAO}
object DbActor {
   case object FetchAll
   case class Update(id: Long, transaction :
Transaction)
   case class Delete(id : Long)
   case class Create(transaction:Transaction)
   case object CreateTable
   case object DropTable
}
class DbActor extends Actor {
     import DbActor._
def receive: Receive = {
  case FetchAll =>
     sender ! DAO.all
  case Update(id: Long, transaction : Transaction) =>
     sender ! DAO.update(id, transaction)
    case Create(transaction : Transaction) =>
      sender ! DAO.create(transaction)
   case Delete(id : Long) =>
     sender ! DAO.delete(id)
   case CreateTable =>
     sender ! DAO.createTable
   case DropTable =>
     sender ! DAO.dropTable}}
```

Останнім етапом в коді master URL є об'єднання усіх елементів для створення основного класу запуску аплікації. У роботі використана скала бібліотека – spray, для запуску сервера і розгортання аплікації. Слід створити конфігурацію, об'єднати її з базою даних, створити сервіс, систему акторів та запустити http сервер.

```
//основний клас запуску аплікації
object Boot extends App with DbSupport{
             implicit  val  system  =
ActorSystem(Config.actorSystemName)
   implicit val timeout = Timeout(5.seconds)
   startDb()
    Config.log.info("Database  is  up  and
running")
             val    application    =
system.actorOf(Props[ApplicationActor],
"website-service")
    val port = Properties.envOrElse("PORT",
Config.port.toString).toInt
    IO(Http) ? Http.Bind(
     listener = application,
     interface = Config.interface,
     port = port
   )}
```

5 РЕЗУЛЬТАТИ

Кінцевим варіантом експерименту є побудова SparkConf об'єкту, який містить інформацію про аплікацію. Він поданий на рис. 8, де наводиться інтерфейс роботи аплікації – зліва та справа – вміст бази даних, після завантаження csv документу з даними на сервер.

Apache Spark – це фреймворк, за допомогою якого можна створити додатки для розподілених інформаційних систем. Для роботи Spark надає програмне API для неперервної роботи з інформаційними процесами. Модуль розкидає код по вузлах кластера, при цьому розбиває їх на задачі, створює план виконання та слідкує за його застосуванням.

Результатом роботи метода Spark є можливість запускати код через різні розподілені інформаційні системи. В режимі Stand-alone mode, метод управляє усіма ресурсами кластера. Yarn дозволяє додаткам запускатись на Hadoop кластері. Mesos та Local mode працюють в локальному режимі та являються альтернативними системами для управління інформаційними процесами кластера.

Метод Spark порівняно з Hadoop MapReduce є гнучкіший. У нього новий архітектурний підхід, який дозволяє серилізувати та зберігати в оперативній пам'яті проміжні дані. Також в процесі його роботи, обмін інформаційними процесами між вузлами проходить напряму, що пришвидшує ініціалізацію та запуск задач метода Spark. Він оперує RDD абстракціями, які більш універсальні, а ніж MapReduce. Також Spark дозволяє розробляти додатки: для задач пакетної обробки даних (batch processing); для роботи з потоками даних(stream processing).

Рисунок 8 – Результат роботи аплікації

На основі проведеного аналізу поданих моделей Spark та Hadoop MapReduce встановлено, що для створення розподіленої інформаційної системи не існує єдиних методів та технологій, які б поєднували всі етапи побудови кластерних систем. Тому робота з детальним описом технологій вертикального масштабування інформації для вирішення такого роду проблем є актуальною.

6 ОБГОВОРЕННЯ

Сьогодні в усьому світі ведеться активна робота над розробленням новітніх інформаційних технологій, які допомагають у надшвидкий час обробляти великі обсяги неструктурованих даних [8–11]. Адже проблемою є широке використання класичних інформаційних систем, які здатні обробляти горизонтальні масиви даних. Проблематика розпаралелення інформаційних потоків спричинена неякісним способом представлення інформації на сайтах, аудіо та відео-метеріалах малою адаптованістю сучасних інформаційних систем.

Дана проблематика немає широкого застосування, адже повільними темпами розробляються інформаційні системи, які би забезпечували повний комплекс технологічного процесу адаптації інформаційних потоків у кластери. Існують поодинокі системи, які виконують лише окремі функції. Це спричинено складністю адаптації та об'єднання різних програмних та апаратних засобів у одну систему. У результаті немає повноцінних розроблених програмно-алгоритмічних засобів, які би забезпечували вирішення сформульованих задач.

ВИСНОВКИ

Завдяки прогресу і багатьом змінам в IT, машинний аналіз та обробка великих обсягів даних сприяють відкриттю нових горизонтів у області наукових досліджень. Розвиток явища, що називається «великими даними» створює можливість трансляції подій у реальному часі, призводить до розробки дедуктивного програмного забезпечення. Такі перспективні напрямки роботи аналізують структуру даних з метою прогнозу результатів та створюють алгоритми передових кореляцій, що сприяють новому розумінню діяльності розподілених інформаційних систем.

ПОДЯКИ

Дослідження, що становлять матеріал статті, безпосередньо пов'язані з науково-дослідним напрямом кафедри інформаційних систем та мереж Національного університету «Львівська політехніка». Результати, викладені у роботі, виконано відповідно до державної науково-дослідної роботи за темою Міністерства освіти і науки України ДБ «ЖЕСТ» у 2011–2012 рр. (номер державної реєстрації 0111U001222).

СПИСОК ЛІТЕРАТУРИ

1. What is Big Data [Electronic resource] – Access mode: http://datascience.berkeley.edu/what-is-big-data/
2. Shaw J. Why "Big Data" Is a Big Deal [Electronic resource] / J. Shaw. – Режим доступу: http://harvardmag.com/pdf/2014/03-pdfs/0314-HarvardMag.pdf
3. Schutt P. What is Big Data? [Electronic resource] / P. Schutt. – Режим доступу: https://blogs.oracle.com/bigdata/big-data-and-analytic-top-10-trends-for-2014
4. Boyko N. Basic concepts of dynamic recurrent neural networks development / N. Boyko, P. Pobereyko // ECONTECHMOD : an international quarterly journal on economics of technology and modelling processes. – Lublin : Polish Academy of Sciences, 2016. – Vol. 5, № 2. – P. 63–68.
5. Leskovec J. Mining of massive datasets / J. Leskovec, A. Rajaraman, J. D. Ullman. – Massachusetts : Cambridge University Press, 2014. – 470 p.
6. Mayer-Schoenberger V. A revolution that will transform how we live, work, and think / V. Mayer-Schoenberger, K. Cukier. – Boston New York, 2013. – 230 p.
7. Boyko N. A look trough methods of intellectual data analysis and their applying in informational systems / N. Boyko // Komp"yuterni nauky ta informatsiyni tekhnolohiyi CSIT 2016 : Materialy XI Mizhnarodnoyi naukovo-tekhnichnoyi konferentsiyi CSIT 2016 : proceedings. – L'viv : Vydavnytstvo L'vivs'koyi politekhniky, 2016. – P. 183–185.
8. Benderskaia E. N. Ostsilliatornye neironnye seti s khaoticheskoi dinamikoi v zadachakh klasternogo analiza / E. N. Benderskaia, S. V. Zhukova // Neirokomp'iutery: razrabotka, primenenie; Radiotekhnika : proceedings. – Moscow : Radyotekhnyka, 2011. – № 7. – P. 74–86.
9. Benderskaia E. N. Modelirovanie neironnoi aktivnosti mozga i bionspirirovannye vychisleniia / E. N. Benderskaia, K. V. Nikitin // Nauchno-tekhnicheskie vedomosti SPbGPU. Informatika. Telecommunicatcii. Upravlenie : proceedings. – St.-Petersburg : Izd-vo Politehn. un-ta. – 2011. – № 6–2(138). – P. 34–40.
10. Benderskaia E. N. Vozmozhnosti primeneniia nekotorykh kharakteristik sinkhronizatsii dlia vyiavleniia samoorganizuiushchikhsia klasterov v ostsilliatornoi neironnoi seti s khaoticheskoi dinamikoi / E.N. Benderskaia // Neirokomp'iutery: razrabotka, primenenie: nauchno-tekhnicheskii zhurnal : proceedings. – Moscow : Nauchnyi tsentr neirokomp'iuterov, 2012. – № 11. – P. 69–73.
11. Feng J. Fixed-point attractor analysis for a class of neurodynamics / J. Feng, D. Brown // Neural Computation : proceedings. – Massachusetts : MIT Press Cambridge, 1998. – Vol. 10. – P. 189–213.
12. Kaneko K. Life: an introduction to complex systems biology / K. Kaneko. – Berlin : Springer-Verlag, 2006. – 369 p.
13. Maass W. Real-time computing without stable states: a new framework for neural computations based on perturbations / W. Maass, T. Natschger, H. Markram // Neural Computation : proceedings. – Switzerland: Institute for Theoretical Computer Science, 2002. – Vol. 11. – P. 2531–2560.
14. Schrauwen B. An overview of reservoir computing theory, applications and implementations / B. Schrauwen, D. Verstraeten, J. V. Campenhout // Proc. of the 15th European Symp. on Artificial Neural Networks : proceedings. – Belgium : Bruges, 2007. – P. 471–482.
15. Coombes S. Waves, bumps, and patterns in neural field theories / S. Coombes // Biological Cybernetics : proceedings. – Nottingham : University of Nottingham, 2005. – Vol. 93, № 2. – P. 91–108.

Стаття надійшла до редакції 02.03.2017.
Після доробки 17.04.2017.

Бойко Н. И.
Канд. экон. наук, доцент, доцент кафедры информационных систем и сетей, Национальный университет «Львовская политехника», Львов, Украина

ПЕРСПЕКТИВНЫЕ ТЕХНОЛОГИИ ИССЛЕДОВАНИЯ БОЛЬШИХ ДАННЫХ В РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Актуальность. Рассмотрены вопросы корректной интерпретации информационных потоков в распределенных информационных системах. Объектом исследования являются методы исследования продвижение «больших данных» по кластерам системы.

Цель работы является исследование перспективных направлений и технологий для анализа структур данных в распределенных информационных системах.

Метод. Рассмотрены технологии обработки больших данных. Проведен анализ каждой из них. Приведен пример применения парадигмы MapReduce, загрузка больших объемов данных на сервер, обработка и анализ неструктурированной информации и распре-

деление ее в кластеризованного базу данных. В статье обобщены понятие «большие данные». Приводятся примеры методов по работе с массивами неструктурированных данных. Выделенные научные направления для анализа больших данных. Сформулированы принципы работы неструктурированных данных в распределенной информационной системы. Приводится работа платформ Hadoop MapReduce и Apache Spark. Анализируются свойства и приводятся различия. Приводится сравнительный анализ производительности обеих платформ в отношении – время выполнения количеству итераций. Рассматриваются способы создания RDD: распараллеливания переданной коллекции в программе и ссылки на внешнюю файловую систему в Hadoop. Также приводится пример распараллеленных системы RDD. Предлагается работа класса одиночка для основных операций с базой данных: подключение к базе данных, создание таблицы, уничтожение таблицы, получение строки по id, возвращение всех элементов базы данных, обновления, удаления и создания строки.

Результаты. Проведенный анализ моделей Spark и Hadoop MapReduce для поеапнои построения распределенной информационной системы. Построен SparkConf объект, который содержит информацию о аппликацию и является конечным вариантом эксперимента.

Выводы. Проведенные эксперименты подтвердили работоспособность предложенных методов, которые способны обрабатывать горизонтальные массивы данных, распараллеленных из-за некачественного способ представления информации. Такие перспективные направления работы анализируют структуру данных для прогноза результатов и создают алгоритмы передовых корреляций, способствующих новому пониманию деятельности распределенных информационных систем. Дальнейшие исследования могут заключаться в широком применении информационных систем, которые бы обеспечивали полный комплекс технологического процесса адаптации информационных потоков в кластеры.

Ключевые слова: система, технология, большие данные, информация, методика, база данных, веб-аппликация, моделирование, обработка, анализ.

Boyko N.

PhD., Associate Professor, Associate Professor of Department of Information Systems and Networks, Lviv Polytechnic National University, Lviv, Ukraine

ADVANCED TECHNOLOGIES OF BIG DATA RESEARCH IN DISTRIBUTED INFORMATION SYSTEMS

Context. Considered question correct interpretation information flow in distributed information systems. The object of study methods are promotion “big data” on cluster system.

Objective. Is the study promising areas and technology for the analysis of structures data in distributed information systems.

Method. The big data tendency prospects as well as timeliness of the problem are studied in this paper. The principles of work with them are addressed. Big data processing technologies are provided. The analysis of each one is performed. An example of “MapReduce” paradigm application, uploading of big volumes of data, processing and analyzing of unstructured information and its distribution into the clustered database is provided. The article summarizes the concept of “big data”. Examples of methods for working with arrays of unstructured data. Dedicated scientific guidance for analyzing big data. The principles of unstructured data in distributed information systems. Driven work platform “Hadoop MapReduce” and “Apache Spark”. Analyzed their properties and given the differences. An analysis of comparative performance against both platforms – the performance of the number of iterations. Consider ways to create RDD: parallelization transmitted collection program and a link to an external file system in “Hadoop”. There is an example rozparalelenoyi system RDD. Proposed work lone class for basic database operations: database connection, create a table, a table, get in line id, returning all elements of the database, update, delete and create the line.

Results. The analysis Models Spark and Hadoop MapReduce for phased construction distributed information system. built up SparkConf object, containing information about applique and is the final version of the experiment.

Conclusions. Conducted experiment confirmed efficiency the proposed method, are capable process horizontal data arrays, that parallelization by defective presentation of information. These promising areas of analyze structure data for the purpose of forecast results and create algorithms advanced correlation, contributing new understanding activity distributed information systems further research can consist in wide use information systems, that would provide a full range technological process adaptation information flows in clusters.

Keywords: system, technology, big data, information, technique, database, Web application, modeling, processing, analytics.

REFERENCES

1. What is Big Data [Electronic resource]. Access mode: http://datascience.berkeley.edu/what-is-big-data/
2. Shaw J. Why “Big Data” Is a Big Deal [Electronic resource]. Rezhym dostupu: http://harvardmag.com/pdf/2014/03-pdfs/0314-HarvardMag.pdf
3. Schutt P. What is Big Data? [Electronic resource]. Rezhym dostupu: https://blogs.oracle.com/bigdata/big-data-and-analytic-top-10-trends-for-2014
4. Boyko N., Pobereyko P. Basic concepts of dynamic recurrent neural networks development, *ECONTECHMOD : an international quarterly journal on economics of technology and modelling processes*. Lublin, Polish Academy of Sciences, 2016, Vol. 5, No. 2, pp. 63–68.
5. Leskovec J., Rajaraman A., Ullman J. D. Mining of massive datasets. Massachusetts, Cambridge University Press, 2014, 470 p.
6. Mayer-Schoenberger V., Cukier K. A revolution that will transform how we live, work, and think. Boston New York, 2013, 230 p.
7. Boyko N. A look trough methods of intellectual data analysis and their applying in informational systems, *Komp”yuterni nauky ta informatsiyni tekhnolohiyi CSIT 2016 : Materialy XI Mizhnarodnoyi naukovo-tekhnichnoyi konferentsiyi CSIT 2016 : proceedings*. L’viv, Vydavnytstvo L’vivs’koyi politekhniky, 2016, pp. 183–185.
8. Benderskaia E. N., Zhukova S. V. Ostsilliatornye neironnye seti s khaoticheskoi dinamikoi v zadachakh klasternogo analiza, *Neirokomp”iutery: razrabotka, primenenie; Radiotekhnika : proceedings*. Moscow, Radyotekhnyka, 2011, No. 7, pp. 74–86.
9. Benderskaia E. N., Nikitin K. V. Modelirovanie neironnoi aktivnosti mozga i bionspirirovannye vychisleniia, *Nauchno-tekhnicheskie vedomosti SPbGPU. Informatika. Telecommunicatcii. Upravlenie : proceedings*. St.-Petersburg, Izd-vo Politehn. un-ta, 2011, No. 6–2(138), pp. 34–40.
10. Benderskaia E. N. Vozmozhnosti primeneniia nekotorykh kharakteristik sinkhronizatsii dlia vyiavleniia samoorganizuiushchikhsia klasterov v ostsilliatornoi neironnoi seti s khaoticheskoi dinamikoi, *Neirokomp”iutery: razrabotka, primenenie: nauchno-tekhnicheskii zhurnal : proceedings*. Moscow, Nauchnyi tsentr neirokomp”iuterov, 2012, No. 11, pp. 69–73.
11. Feng J., Brown D. Fixed-point attractor analysis for a class of neurodynamics, *Neural Computation : proceedings*. Massachusetts, MIT Press Cambridge, 1998, Vol. 10, pp. 189–213.
12. Kaneko K. Life: an introduction to complex systems biology. Berlin, Springer-Verlag, 2006, 369 p.
13. Maass W., Natschger T., Markram H. Real-time computing without stable states: a new framework for neural computations based on perturbations, *Neural Computation : proceedings*. Switzerland, Institute for Theoretical Computer Science, 2002, Vol. 11, pp. 2531–2560.
14. Schrauwen B., Verstraeten D., Campenhout J. V. An overview of reservoir computing theory, applications and implementations, *Proc. of the 15th European Symp. on Artificial Neural Networks : proceedings*. Belgium, Bruges, 2007, pp. 471–482.
15. Coombes S. Waves, bumps, and patterns in neural field theories, *Biological Cybernetics : proceedings*. Nottingham, University of Nottingham, 2005, Vol. 93, No. 2, pp. 91–108.