ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНИВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ЕКОНОМІЧНОЇ ІНФОРМАТИКИ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Пояснювальна записка

до дипломної роботи

МАГІСТРА

на тему: «Аналіз продуктивності кластерів з оптимізованою пам’яттю на платформі Azure для розв’язання задач машинного навчання»

Виконав**:** студент 2 року навчання,
за освітнім ступенем «магістр»
зі спеціальності 122 «Комп’ютерні науки»
Самотой В. Є.

Керівник: д.т.н., проф. Мінухін С. В.

Харків – 2019 рік

РЕФЕРАТ

Пояснювальна записка до магістерської дипломної роботи містить: 115 с., 60 рис., 30 табл., 61 джерело, 4 додатки.

Метою роботи є дослідження продуктивності роботи кластеру Apache Spark на платформі Azure HDInsight для різних конфігурацій віртуальних машин з оптимізованою пам’яттю для методів машинного навчання шляхом проведення експериментального дослідження щодо виявлення найбільш оптимальної конфігурації.

Об’єкт дослідження – конфігурації віртуальних машин для побудови кластера Apache Spark, конфігурації компонентів Apache Spark.

Предмет дослідження – показники продуктивності кластерів при проходженні тестування за допомогою бенчмарку Spark-Perf.

Результатами дослідження є методи оцінки продуктивності кластеру та програмне забезпечення для статистичного аналізу отриманих результатів тестування бенчмарком Spark-Perf на основі лог-файлів.

Отримані результати дослідження можуть бути використані для оптимізації конфігурації кластеру та Apache Spark при розв’язанні задач машинного навчання.

ОЦІНКА, ПРОДУКТИВНОСТЬ, КЛАСТЕР, РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ, МАШИННЕ НАВЧАННЯ, КЛАСТЕРИЗАЦІЯ, AZURE HDINSIGHT, APACHE SPARK, SPARK MLLIB, SPARK MEASURE, APACHE HADOOP, YARN, GRAFANA, GRAPHITE, APACHE AMBARI

ABSTRACT

Explanatory note to the master's thesis contains: 115 p., 60 fig., 30 tab., 61 applications sources, 4 annexes.

The aim of the work is to study the performance of the Apache Spark cluster on the Azure HDInsight platform for various virtual machine configurations for machine learning methods by conducting an experimental research to identify the most optimal configuration.

The object of research is the configuration of virtual machines for building an Apache Spark cluster, the configuration of Apache Spark Components.

The subject of the research is the cluster performance indicators obtained during testing using the Spark-Perf benchmark suite.

The results of the research are methods for evaluating cluster performance, as well as software for statistical analysis of the results of testing machine learning methods with Spark-Perf benchmark suite based on log files.

The results of the research can be used to optimize the cluster configuration for solving problems of machine learning.

CLUSTER PRODUCTIVITY ASSESSMENT, DISTRIBUTED COMPUTING, MACHINE LEARNING, CLUSTERING, AZURE HDINSIGHT, APACHE SPARK, SPARK MLLIB, SPARK MEASURE, APACHE HADOOP, YARN, GRAFANA, GRAPHITE, APACHE AMBARI

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВМ – віртуальна машина;

API (application programming interface) – програмний інтерфейс;

ML (machine learning) – машинне навчання;

MLaaS (machine learning as a service) – ряд платформ, які надають послугимашинного навчання.

SDK (software development kit) – набір із засобів розробки, утиліт і документації;

CLI (command-line interface) – інтерфейс командного рядка;

ЦП – центральний процесор;

ОЗП – оперативний запам'ятовуючий пристрій, тобто оперативна пам’ять;

DTR (decision tree) – дерево рішень.

СКВ – середньоквадратичне відхилення.

ВСТУП

Останнім часом така галузь як хмарні обчислення набирає все більшої популярності, що сприяє її активному розвитку та поширенню. Основними перевагами хмарних сервісів є те, що вони надають швидкий доступ до гнучких та недорогих обчислювальних ресурсів. Актуальність роботи пов’язана зі зростанням використання машинного навчання у різних галузях роботи з даними та необхідність масштабувати та розподіляти обчислення для підвищення ефективності. При опрацюванні даних, що надходять у великих об’ємах, для вирішення задач використовують розподілені та масштабовані системи.

Однією із сфер застосування хмарних обчислень є розв’язання задач машинного навчання. Здебільшого для їх розв’язку створюється кластер з декількох обчислювальних машин, спеціально оптимізованих під конкретний тип задачі. На побудування локального кластеру потрібно багато ресурсів, часу та простору, тому набагато ефективніше для його створення використовувати платформу хмарних обчислень.

Метою роботи є дослідження продуктивності роботи з кластером Apache Spark на платформі Azure для різних конфігурацій віртуальних машин для методів машинного навчання шляхом проведення експериментального дослідження щодо виявлення найбільш оптимальної конфігурації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати методи розгортання кластеру для вирішення задач машинного навчання на хмарних платформах;
- проаналізувати методи підвищення продуктивності кластеру Azure Spark на хмарних платформах;
- розробити інформаційне забезпечення для моніторингу кластеру та порівняння різних показників кластеру;
- провести ряд тестувань для отримання статистики на різних конфігураціях кластерів.

Об’єкт дослідження – конфігурації віртуальних машин для побудови кластера Apache Spark, конфігурації компонентів Apache Spark.

Предмет дослідження – показники продуктивності кластерів при проходженні тестування в тестах машинного навчання Spark-Perf.

При виконанні роботи використовуються такі емпіричні методи дослідження: обчислювальні експерименти та статистичний аналіз отриманих результатів тестування методів машинного навчання в тестах Spark-Perf на основі лог-файлів результатів.

У результаті дослідження було протестовано різні конфігурації віртуальних машин для ПЗ Apache Spark у Azure HDInsight, різні конфігурації компонентів Apache Spark, обрані і підключені БД для тестування бенчмарку Spark-Perf, проведені обчислювальні експерименти та статистичний аналіз отриманих результатів тестування методів машинного навчання в тестах Spark-Perf на основі лог-файлів.

Результати дослідження можуть бути використані для оптимізації конфігурації кластеру при розв’язанні конкретних задач машинного навчання.

РОЗДІЛ 1. АКТУАЛЬНІСТЬ ВИКОРИСТАННЯ ХМАРНИХ ПЛАТФОРМ ДЛЯ ЗАДАЧ МАШИННОГО НАВЧАННЯ. ОГЛЯД МОЖЛИВОСТЕЙ ДОСТУПНИХ ІНСТРУМЕНТІВ

1.1. Організаційно-правова форма організації (підприємства)

Харківський національний економічний університет імені Семена Кузнеця є державним вищим навчальним закладом вищого, IV рівня акредитації, що підпорядковується Міністерству освіти і науки України.

На сьогодні Харківський національний економічний університет імені Семена Кузнеця є провідним спеціалізованим вищим навчальним закладом Сходу України, який надає повний спектр навчальних послуг, здійснюючи багатоступеневу підготовку, перепідготовку та підвищення кваліфікації фахівців з 26 спеціальностей.

До складу університету входять 7 факультетів, 35 кафедр, відділ заочної, дистанційної та післядипломної освіти, аспірантура та докторантура, спецради із захисту дисертацій, науково-дослідні та навчальні лабораторії. Також університет має такі підрозділи: бібліотека, відділ молодіжної політики та соціального розвитку, відділ працевлаштування студентів та взаємодії з бізнес структурами, інформаційно-обчислювальний центр, відділ забезпечення якості освіти та інноваційного розвитку, видавництво ХНЕУ ім. С. Кузнеця, видавничий дім «ІНЖЕК», відділ маркетингу та корпоративних комунікацій, відділ електронних засобів навчання, методичний відділ, медпункт.

Університет здійснює активну міжнародну діяльність на основі більш ніж 80 підписаних угод про співробітництво з вищими навчальними закладами та організаціями Європи і світу.

Кафедра інформаційних систем є однією з шести кафедр факультету економічної інформатики. Факультет економічної інформатики розпочав свою діяльність під назвою “Механізація обліку й обчислювальних робіт” у 1961 р. За період існування на факультеті підготовлено більш ніж 6800 фахівців, з них більше 700 фахівців для інших країн. Зараз на факультеті навчається більше ніж

800 студентів, підготовку ведуть понад 130 викладачів, з яких 80% мають вчені ступені та звання.

1.1.1. Цілі та завдання організації

Кафедра орієнтована на підготовку спеціалістів до проектної, аналітичної, науково-дослідній та виробничо-технологічній роботі в організації різноманітних галузей, св'язаних з дослідженням і розробкою ефективних методів реалізаціх інформаційних процесів і побудові інформаційних систем у прикладних областях на основі використання сучасних інформаційно-комунікаційних технологій з використанням прикладної інформатики, математичних та інструментальних методів моделювання і прогнозування видобничих процесів, що володіють фундаментальними науковими знаннями у предметній області. Випускники кафедри працюють IT-спеціалістами, розробниками програмного забезпечення в банках та на підприємствах, адміністраторами баз даних і комп'ютерних мереж в комерційних структурах різного розміру.

1.1.2. Історія створення організації

Кафедра інформаційних систем (ІС) заснована в 1964 р. На кафедрі працює 40 викладачів, з них 24 мають вчене звання доцента та професора, вчений ступінь доктора та кандидата наук. Стратегія підготовки студентів кафедри ґрунтується на принципах фундаментальності та цілісності надання знань, їх практичної спрямованості, індивідуалізації навчання, наукового та системного підходів. Навчальна програма орієнтована на підготовку спеціалістів зі знаннями, достатніми для того, щоб стати інтелектуальною елітою сучасного світу високих технологій.

Кафедра бере участь у міжнародних програмах підготовки висококваліфікованих фахівців, а саме: спільній франко-українській програмі підготовки магістрів за фахом "Бізнес-інформатика" з університетом Ліон 2 (Ліон, Франція); спільній магістерській програмі подвійного диплому «Створення інноваційних підприємств» з університетом Монпельє (Монпельє, Франція); спільній словаць-

ко-українській програмі «Бізнес-аналітика» та інформаційні системи у підприємництві» (Братислава, Словаччина).

1.1.3. Організаційна структура управління організацією

На рис. 1 представлена схема організаційної структури факультету економічної інформатики Харківського національного економічного університету ім. Семена Кузнеця.

Рисунок 1 – Схема організаційної структури факультету економічної інформатики

Структура кафедри Інформаційних систем складається з:

- завідувач кафедри;
- науково-педагогічна частина;
- допоміжний персонал.

Науково-педагогічна частина складається з:

- професори;
- доценти;
- викладачі.

До допоміжного персоналу відносяться:

- інженери;
- лаборанти.

Структура кафедри Інформаційних систем зображена на Рис. 2.

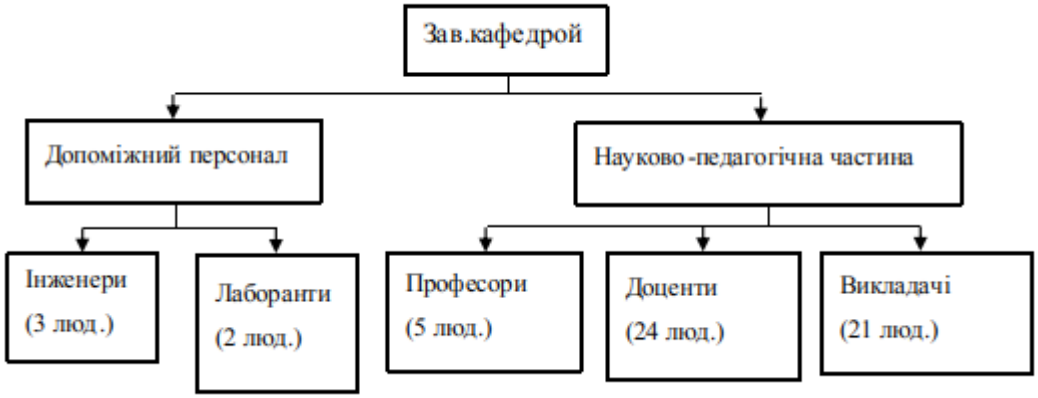

Рисунок 2 – Структура кафедри Інформаційних систем

1.1.4. Види діяльності організації

Основними напрямами наукових досліджень кафедри інформаційних систем є:

1. Геоінформаційні системи та технології.
2. Інтелектуальні методи обробки інформації.
3. Інформаційні технології та комп’ютерні засоби моделювання в економіці.
4. Машинне навчання.
5. Надпродуктивні обчислювальні системи обробки поточних даних.
6. Планування та управління ресурсами в GRID-системах.
7. Сучасні методи програмування в інформаційних системах.

Кафедрою інформаційних систем Харківського національного економічного університету ім. Семена Кузнеця кожного року проводяться наукові конференції.

1.2. Аналіз інформаційної системи на підприємстві

На базі кафедри побудована комп'ютерна система, структурна схема якої представлена на рис. 3.

П’ять персональних комп'ютерів (ПК) об'єднані в локальну мережу. На їх базі побудован кластер за домопогою Apache Spark. На одному з ПК встановлене программне забезпечення для управління завданнями та моніторингом ресурсів.

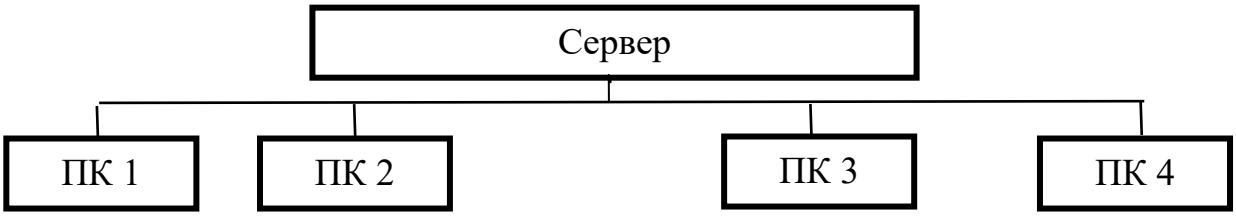

Рисунок 3 – Структура комп'ютерної системи факультету

1.3. Оцінювання проблем підприємства в сфері інформаційних систем і технологій

Однією з проблем у сфері ІС на підприємстві є те, що в недостатньому об’ємі використовуються паралельні обчислення на основі хмарних обчислень.

Хмарні обчислення – один з напрямів інформаційних технологій, що стрімко розвивається та все більше використовується. Хмарні обчислення – це надання обчислювальних потужностей, сховищ для баз даних, додатків та інших IT-ресурсів на платформах хмарних сервісів через Інтернет з оплатою за фактом використання.

Платформи хмарних сервісів надають швидкий доступ до гнучких і недорогих IT-ресурсів. Хмарні обчислення дозволяють позбутися великих попередніх витрат на обладнання і заощадити час, необхідний для управління ним. Замість цього можна розподілити обчислювальні ресурси таких типів і розмірів, які необхідні для реалізації ідей користувача або для управління
IT-відділом. Можна практично миттєво отримувати доступ до необхідної кількості ресурсів з оплатою за фактом використання. Хмарні обчислення забезпечують простий доступ до серверів, сховищ, баз даних і великого набору сервісів додатків в Інтернеті.

Хмарні обчислення мають велику кількість переваг:

1. Капітальні витрати перетворюються в змінні. Немає необхідності вкладати великі кошти в центри обробки даних (ЦОД) і сервери, не знаючи заздалегідь, які потужності згодом будуть потрібні. Можна платити тільки за фактичне використання обчислювальних ресурсів.

2. Значна економія при великих обсягах. При використанні хмарних обчислень можна досягти нижчої змінної вартості, ніж при створенні власної обчислювальної інфраструктури. Оскільки сотні тисяч клієнтів спільно використову-

ють хмарні ресурси, постачальники хмарних послуг можуть забезпечити значну економію за рахунок масштабу, що дозволяє знизити вартість фактичного використання ресурсів.

3. Немає необхідності прогнозувати, який обсяг ресурсів інфраструктури буде потрібним. Приймаючи рішення щодо обсягу ресурсів для розгортання програми, часто в кінцевому рахунку можна зіткнутися з простоєм дорогих ресурсів або нестачею потужностей. При використанні хмарних обчислень ці проблеми зникають. Можна використовувати тільки потрібні ресурси з можливістю масштабування в бік збільшення або зменшення всього за кілька хвилин.

4. Збільшення швидкості і гнучкості розробки. Доступ до нових IT-ресурсів для розробників в середовищі хмарних обчислень можна отримати за один клік мишею. Це дозволяє скоротити час, необхідний для їх впровадження, з декількох тижнів до хвилин. В результаті організація стає більш гнучкою, оскільки на експерименти і розробку витрачається набагато менше часу і коштів.

5. Відмова від витрат на запуск і підтримку ЦОД. Хмарні обчислення дозволяють сконцентруватися на потребах клієнтів, а не на складних завданнях по установці, комплектації і забезпеченні живлення серверів.

6. Вихід на світовий рівень за лічені хвилини. Просте розгортання додатків в безлічі регіонів по всьому світу виконується всього за кілька кліків мишкою. Таким чином можна легко та з мінімальними витратами забезпечити меншу затримку і поліпшити якість обслуговування клієнтів.

Світовий ринок хмарних рішень і послуг зростає настільки інтенсивно, що передбачити темп його збільшення виявляється на практиці досить важко, тому дані провідних аналітичних компаній інколи сильно відрізняються. Проте, всі вони фіксують одні й ті ж тенденції: швидкий темп зростання витрат на хмарні обчислення, а також супутнього ринку сервісів, центрів обробки даних і трафіку даних в таких системах.

Тому доцільно підвищити об’єм використання хмарних обчислень для різних задач підприємства для підвищення ефективності використання ресурсів підприємства.

1.4. Відомості про Apache Spark

Apache Spark програмним забезпеченням з відкритим вихідним кодом для вирішення задач Bigdata та має різні переваги перед іншими ПЗ, зокрема воно є динамічним за природою, підтримує обчислення в пам'яті RDD та надає можливість повторного використання, толерантності до помилок, обробки потоку в реальному часі тощо. Фреймворк Spark - це не що інше, як загальноприйнята платформа обчислювальних кластерів. Він включає API, яке допомагає працівникам, що працюють з даними виконувати потокове, машинне навчання або робочі високонавантажені запити SQL, які вимагають повторного доступу до наборів даних. Spark може виконувати одночасно пакетну обробку та обробку потоків. Пакетна обробка відноситься до обробки раніше зібраної роботи в єдиному пакеті, в той час як потокова обробка означає роботу з потоковими даними Spark. Він може отримати доступ до будь-якого джерела даних Hadoop, а також може працювати на кластерах Hadoop. Крім того, Apache Spark поширює можливості Hadoop MapReduce, що включає в себе ітераційні запити та обробку потоків.

Apache Spark пропонує високорівневі API для розробників на таких мовах програмування, як Java, Scala, Python та R. У порівнянні Spark з Hadoop, він у 100 разів швидше, ніж Big Data Hadoop, і має у 10 разів швидший доступ до даних з диска.

1.4.1. Spark In-memory Computing.

При обчисленні в пам'яті дані зберігаються в оперативній пам'яті (ОП) замість деяких повільних дисків і обробляються паралельно. Користуючись цим, можна виділити шаблон, при аналізі великих даних, що знижує вартість використання пам'яті. Таким чином, обробка в пам'яті мінімізується для додатків.

Основою Apache Spark є Resilient Distributed Dataset (RDD).

RDD є стійкою до відмов колекцією елементів, які можуть використовуватись для паралельної обробки (з існуючої колекції можна у паралельний спосіб створити RDD у програмі-драйвері). Також, її можна посилатися на набір даних у зовнішній системі зберігання, такий як загальна файлова система, HDFS, HBase або будь-яке джерело даних, що пропонує Hadoop InputFormat.

За замовчуванням кожен RDD може бути перераховано кожен раз при виконанні дії на ньому. Тим не менш, ви можете зберігати RDD в пам'яті, використовуючи метод persist (або кеш), в цьому випадку Spark зберігатиме елементи навколо кластера для набагато швидшого доступу наступного разу, коли ви його запитаєте. Існує також підтримка постійних RDDs на диску, або реплікації на декількох вузлах.

Основні компоненти, з яких складається Apache Spark, наведені на рис. 4.

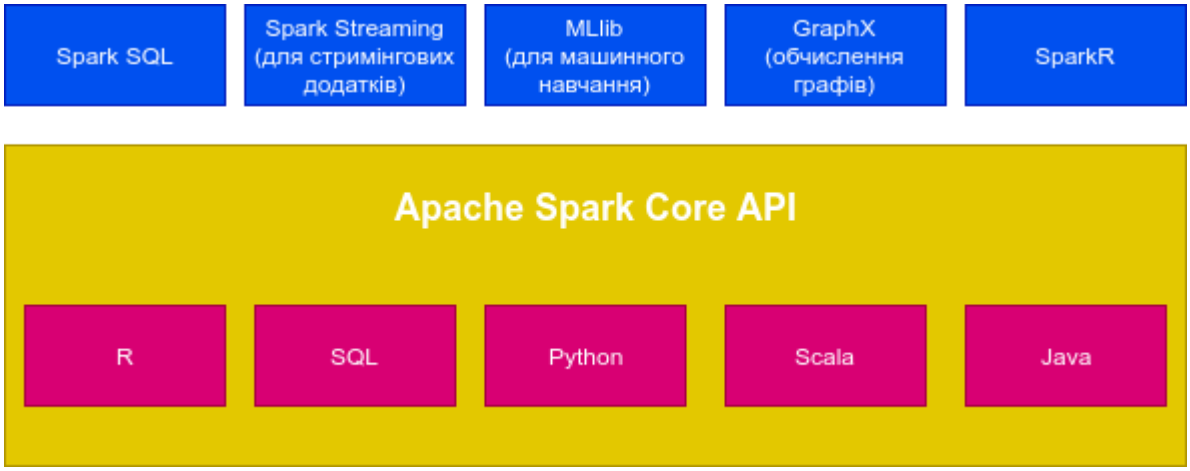

Рисунок 4 – Склад компонент Spark Ecosystem

Гнучкість Spark полягає ще і в тому, що програми можуть бути запущені у таких режимах за методом керування:

1) Standalone. У цьому режимі можна самостійно розгорнути інфраструктуру Spark, він сам буде керувати всіма ресурсами кластера і виконувати програми.

2) Yarn. Цей механізм входить в екосистему Hadoop.

3) Mesos. Альтернативний механізм управління ресурсами кластера.

4) Local mode. Локальний режим, створений для розробки і налагодження програм.

У всіх цих систем є свої переваги, які є актуальними для різного роду завдань і вимог.

Spark використовує архітектуру майстер-робітник, що відображена на рис. 5. Існує драйвер, який спілкується з одним координатором по імені master, який керує worker’ами, в яких працюють виконавці.

Драйвер і виконавці працюють в своїх власних процесах Java. Можна запустити їх на одній машині (вертикальний кластер) або на окремих машинах (горизонтальний кластер) або в змішаній конфігурації комп'ютерів.

SparkContext є входом для функцій Apache Spark. Найважливішим кроком будь-якого додатка драйвера Spark є створення SparkContext. Вона дозволяє вашому Spark Application отримувати доступ до кластеру Spark за допомогою менеджера ресурсів (YARN / Mesos).

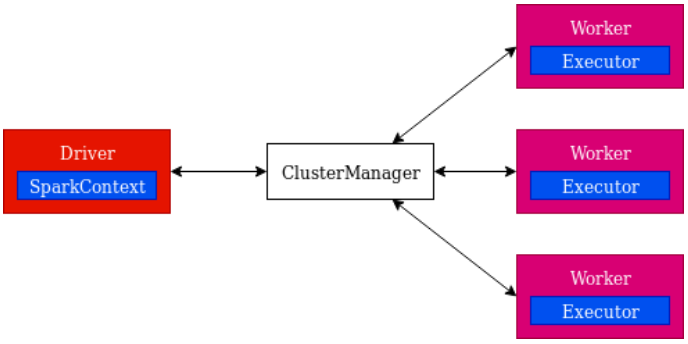

Рисунок 5 – Схема взаємодії менеджера та робочих вузлів (виконавців) Spark

1.4.2. DataSets

Завданням Spark DataSets є забезпечення програмного інтерфейсу, що дозволяє користувачам легко виражати перетворення об'єктів, одночасно забезпечуючи продуктивність і переваги надійного механізму виконання Spark SQL.

Подібно RDD, DataFrame є незмінним розподіленим збором даних. На відміну від RDD, дані організовані в імена стовпців, як таблиця в реляційній базі даних. Ця структура була розроблена для того, щоб зробити обробку великих наборів даних ще простішою, вона дозволяє розробникам нав'язувати структуру розподіленому збору даних, дозволяючи абстракцію більш високого рівня; він надає мовний API для конкретних доменів, щоб маніпулювати розподіленими даними; і робить Spark доступним для широкої аудиторії, за винятком спеціалізованих інженерів даних.

1.4.3. Spark Core

Spark Core є центральною точкою Spark. В основному, вона забезпечує платформу виконання для всіх додатків Spark.

Рисунок 6 – Функції SparkContext в Apache Spark

1.4.4. Spark MLlib

Spark MLlib – це бібліотека машинного навчання, яка забезпечує як ефективність, так і високоякісні алгоритми, а її здатність до обробки даних в пам'яті значно покращує продуктивність ітеративного алгоритму.

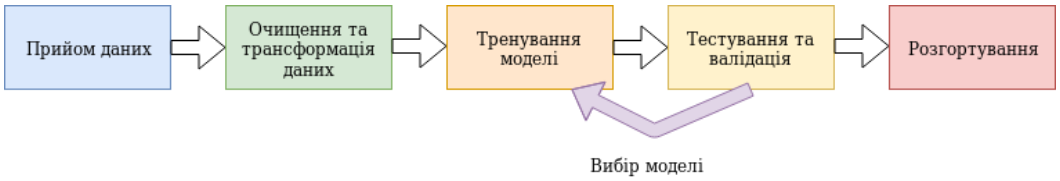

Рисунок 7 – Типова схема Spark ML конвеєру

В Spark 2.0 API RDD на базі пакету spark.mllib увійшов до режиму технічного обслуговування. У цьому випуску API на основі DataFrame є основним API для навчання Spark. Отже, MLlib не додає нових функцій до API, що базується на RDD.

Причина переходу MLlib на API на основі DataFrame полягає в тому, що він є більш зручним для користувача, ніж RDD. Однією з переваг використання

DataFrames є джерела даних Spark Data, SQL DataFrame. MLlib також використовує пакет лінійної алгебри Breeze, що є колекцією бібліотек для чисельних обчислень і машинного навчання.

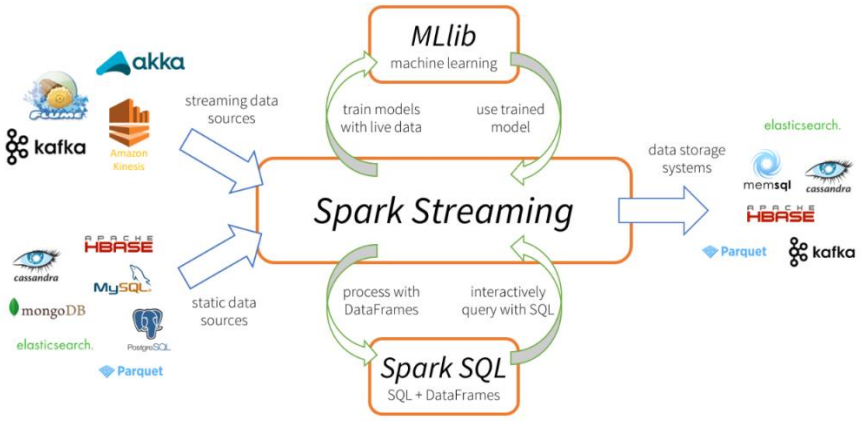

Рисунок 8 – Схема взаємодії компонент Spark з пакетом MLlib

1.5. Режими запуска Apache Spark

1.5.1. Автономний режим

Автономний режим – легкий менеджер кластеру, включений до Spark. Це робить його простим для налаштування кластеру і він може виконуватися на Windows, Linux або Mac OSX. Для протоколів зв'язку Spark використовує шифрування SSL, але для блокової передачі даних треба використовувати SASL-шифрування.

Режими налаштування роботи Apache Spark наведені на рис. 9, 10.

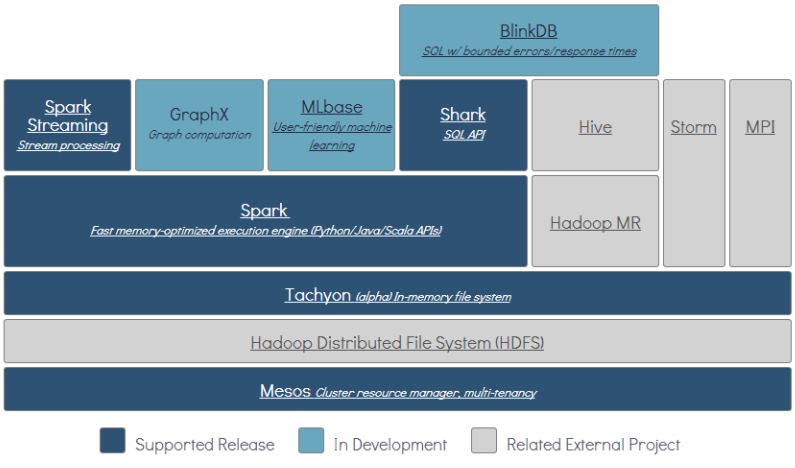

Рисунок 9 – Налаштування Apache Spark

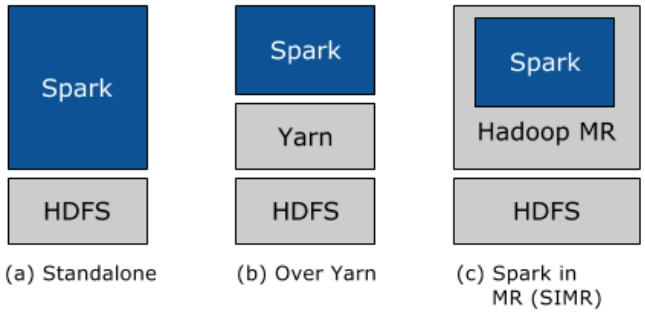

Рисунок 10 – Режими налаштування роботи Apache Spark

1.5.2. Spark з Mesos.

При використанні Mesos він використовується як менеджер кластера. Тепер, коли драйвер створює завдання і запускається, видаючи чергу завдань для планування, Mesos визначає, які машини будуть оброблювати ці завдання. Кілька фреймворків можуть співіснувати на одному кластері, не вдаючись до статичного розбиття ресурсів (рис. 11).

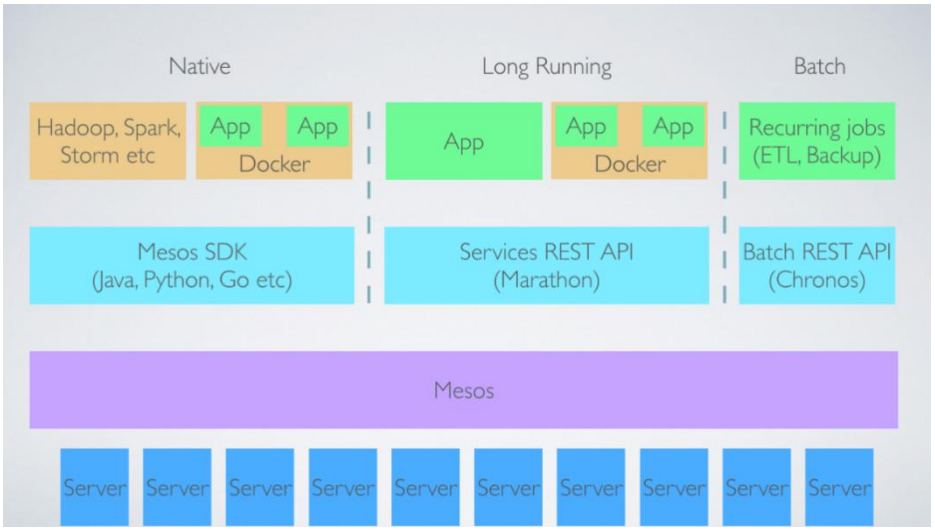

Рисунок 11 – Схема запуску завдань на кластері Spark з Mesos

1.6. Опис платформи Azure HDInsight

Дослідження продуктивності роботи кластеру Apache Spark проводиться на платформі Azure HDInsight для віртуальних машин з оптимізованою пам’яттю, а

саме на версії 3.6. Детальна інформація про версії супутніх компонентів Azure HDInsight представлену у таблиці 1.

Таблиця 1 – Версії компонент HDInsight 3.6

Компонент ПЗ	Версія ПЗ
Apache Hadoop и YARN	2.7.3
Apache Tez	0.7.0
Apache Pig	0.16.0
Apache Hive	2.1.0, 1.2.1
Apache Tez Hive2	0.8.4
Apache Ranger	0.7.0
Apache HBase	1.1.2
Apache Sqoop	1.4.6
Apache Oozie	4.2.0
Apache Zookeeper	3.4.6
Apache Storm	1.1.0
Apache Mahout	0.9.0+
Apache Phoenix	4.7.0
Apache Spark	2.3.0, 2.2.0, 2.1.0
Apache Livy	0,4, 0,4, 0,3
Apache Kafka	1.1, 1.0
Apache Ambari	2.6.0
Apache Zeppelin	0.7.0
Mono	4.2.1

Одними з найважливіших компонент Azure HDInsight є Apache Hadoop та YARN, оскільки покращення продуктивності роботи напряму залежить від їх конфігурації. Apache Hadoop складається з двох основних компонентів - розподіленої файлової системи (HDFS) Apache Hadoop, яка забезпечує зберігання, і модуля управління ресурсами Apache Hadoop YARN, що забезпечує обробку. Завдяки можливостям зберігання і обробки на кластері можна запускати програми MapReduce, щоб виконувати необхідну обробку даних.

Платформа YARN управляє обробкою даних в Hadoop. Вона складається з двох основних служб, що виконуються в якості процесів на вузлах кластера:

- диспетчер ресурсів (ResourceManager);
- диспетчер вузлів (NodeManager).

Диспетчер ресурсів (ResourceManager) надає обчислювальні ресурси кластера процесам. Ці ресурси надаються в якості контейнерів, кожен з яких складається з виділених ядер ЦП і обсягу оперативної пам'яті.

Диспетчер вузлів (NodeManager) запускає завдання, з яких складаються програми, а потім повідомляє про хід виконання та стан програми до ApplicationMaster. Він, в свою чергу, відправляє звіт про стан програм в ResourceManager, який повертає результати клієнту.

YARN розгортається у всіх типах кластерів HDInsight. Для забезпечення високого рівня доступності ResourceManager розгортається в двох екземплярах (первинний і вторинний), що виконуються на першому і другому головних вузлах в кластері відповідно. Одночасно може бути активним тільки один екземпляр ResourceManager. Примірники NodeManager виконуються на робочих вузлах, доступних в кластері. Базова архітектура кластеру HDInsight з YARN зображена на рис. 12.

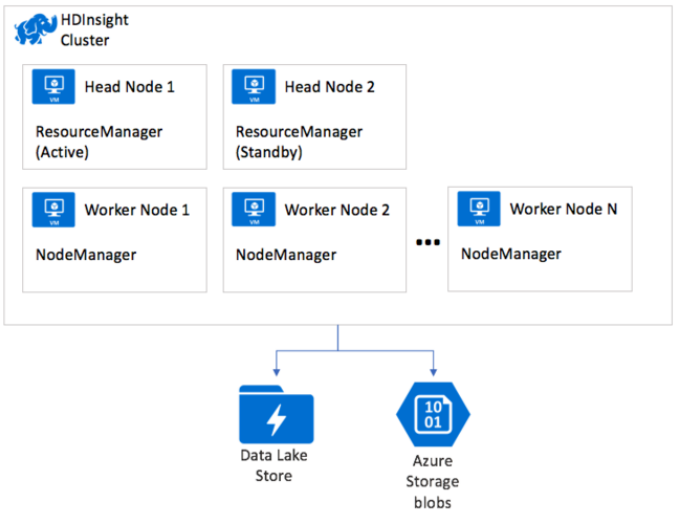

Рисунок 12 – Схематична архітектура кластеру HDInsight

1.7. Опис віртуальних машин з оптимізованою пам’ятю для HDInsight

Віртуальні машини з оптимізованою пам’ятю для HDInsight забезпечують високе співвідношення оперативної пам’яті до продуктивності процесора, що чудово підходить для серверів реляційних баз даних, кешів середнього та великого рівнів та аналізу в пам'яті.

Віртуальні машини серії Dv2 працюють на процесорах Intel® Xeon® 8171M 2.1 ГГц (Skylake) або Intel® Xeon® E5-2673 v4 2.3 ГГц (Broadwell) або

процесорах Intel® Xeon® E5-2673 v3 2,4 ГГц (Haswell) та підтримують Intel Turbo Boost 2.0.

Azure Compute пропонує розміри віртуальної машини, які виділяються під конкретний тип обладнання та присвячені одному клієнту. Ці розміри віртуальних машин найкраще підходять для робочих навантажень, які вимагають високого рівня ізоляції від інших клієнтів для робочих навантажень, що включають такі елементи, як відповідність та нормативні вимоги. Клієнти також можуть додатково розподілити ресурси цих ізольованих віртуальних машин, використовуючи підтримку Azure для вкладених віртуальних машин. Будь ласка, дивіться таблиці сімейств віртуальних машин нижче для ваших ізольованих параметрів VM.

У таблиці 2 наведені характеристики віртуальних машин серії Dv2.

Таблиця 2 – Характеристики віртуальних машин серії Dv2

Віртуальна машина	Віртуальних ядер CPU, шт.	Оперативна пам'ять, ГБ	Тимчасове сходище, ГБ	Максимальна пропускна здатність сховища: IOPS / Read MBps / Write MBps	Максимальна кількість дисків даних / пропускна здатність: IOPS	Максимальна NICs / Очікувана пропускна здатність мережі (Mbps)	Ціна операційної системи, $/годину	Ціна за HDInsight, $/годину	Загальна ціна, $/годину
Standard_D11_v2	2	14	100	6000 / 93 / 46	8 / 8x500	2 / 1500	0,149	-	0,149
Standard_D12_v2	4	28	200	12000 / 187 / 93	16 / 16x500	4 / 3000	0,299	0,075	0,374
Standard_D13_v2	8	56	400	24000 / 375 / 187	32 / 32x500	8 / 6000	0,598	0,15	0,748
Standard_D14_v	16	112	800	48000 / 750 / 375	64 / 64x500	8 / 12000	1,196	0,3	1,496

Віртуальна машина	Віртуальних ядер CPU, шт.	Оперативна пам'ять, ГБ	Тимчасове сходище, ГБ	Максимальна пропускна здатність сховища: IOPS / Read MBps / Write MBps	Максимальна кількість дисків даних / пропускна здатність: IOPS	Максимальна NICs / Очікувана пропускна здатність мережі (Mbps)	Ціна операційної системи, $/годину	Ціна за HDInsight, $/годину	Загальна ціна, $/годину
2									
Standard_D15_v2 1	20	140	1000	60000 / 937 / 468	64 / 64x500	8 / 25000	1,495	-	1,495

1.8. Характеристика пакету Machine Learning Library Apache Spark

Машинне навчання являє собою наукову галузь, що займається вивченням алгоритмів та статистичних моделей, які комп’ютерні системи використовують для вирішення конкретного завдання без використання чітких інструкцій, а спирається на закономірність та відомі висновки до вихідних даних. Алгоритми машинного навчання будують математичну модель на основі вибіркових даних, що відомі як «дані для тренування» для того, щоб приймати не чітко запрограмовані рішення чи давати прогнози для вирішення завдання.

Розрізняють два типи машинного навчання: індуктивне навчання та дедуктивне. Дедуктивне навчання відноситься до області експертних систем, в той час як індуктивне навчання можна вважати синонімом до класичного поняття машинного навчання. Спираючись на варіант підходу, виділяють два типи індуктивного машинного навчання: контрольоване, що також називається навчанням з учителем), неконтрольоване (навчання без учителя) та навчання з підкріпленням.

Контрольоване навчання працює з набором даних, який подається у вигляді пар вхідних-вихідних даних, наприклад, набір даних, що містить різні значення ознак обраного товару, а також, відповідну до характеристик, його ціну. Контрольований алгоритм навчання аналізує дані для навчання та виробляє

певну функцію, яку можна використовувати для відтворення нових прикладів. Алгоритми, що відносяться до контрольованого навчання поділяються на дві підкатегорії:

- Алгоритми класифікації – в результаті роботи повертається однозначний висновок щодо належності до певної категорії;
- Алгоритми регресії – замість чіткої відповіді повертають діапазон можливих значень.

Неконтрольоване навчання - тип самоорганізованого навчання, яке базується на біхевіористській психології, який займається вивченням питання, які дії буде виконувати вихідний об’єкт у певній ситуації, що називається середовищем. Головним завданням цього типу навчання є знаходження зв’язків між окремими даними, виявлення закономірностей та підбір шаблонів для класифікації даних. Воно набуло свого поширення в системах рекомендацій, наприклад коли в магазині обираються товари, що можуть зацікавити покупця з більшою ймовірністю, базуючись на його попередніх переглядах чи покупках.

Навчання з підкріпленням - тип навчання, при якому програмний алгоритм взаємодіє з динамічним середовищем, в якому він повинен виконувати певну дію (наприклад, керування транспортним засобом або грати в гру проти супротивника). На вхід надається зворотний зв'язок з точки зору винагород та покарань, оскільки вона орієнтується в середовищі, що відповідає предметрій області.

Найбільш поширеними варіантами використання машинного навчання є:

- моніторинг систем безпеки та виявлення шахрайства;
- рекомендації продуктів та персоналізовані рекомендації;
- показ таргетованої реклами;
- розпізнавання рукописного тексту та природнього мовлення.

Для коректної роботи машинного навчання потрібні великі набори даних, щоб виявити закономірності та їх основі приймати рішення чи робити прогнози.

Spark MLlib - це компонент машинного навчання Apache Spark. Однією з головних особливостей Spark є можливість масового масштабування обчислень, а саме це і потрібно для алгоритмів машинного навчання. Але майже всі

алгоритми машинного навчання не можуть бути ефективно паралелізовані. Кожен алгоритм має свої завдання для паралелізації, будь то паралелізм задач чи паралелізм даних.

Бібліотека MLlib складається з двох програмних пакетів:

- spark.mllib - містить оригінальний API, побудований на RDD;
- spark.ml – реалізація API більш високого рівня, побудований з використанням DataFrames для конструювання конвеєрів машинного навчання, також називаємих ML Pipelines.

У Spark MLlib реалізована підтримка кількох типів даних для представлення вхідних даних і вихідних результатів. Типи даних Spark MLlib поділяються на наступні категорії:

- локальний вектор (local vector);
- локальна матриця (local matrix);
- розмічена точка (labeled point);
- розподілена матриця (distributed matrix).

Задачі машинного навчання, які можна вирішити вбудованими в Spark MLlib алгоритмами машинного навчання:

- регресія:
    - лінійна регресія (Linear regression);
    - узагальнена лінійна регресія (Generalized linear regression);
    - регресія дерева рішень (Decision tree regression);
    - регресія випадкового лісу (Random forest regression);
    - регресія дерева градієнтного бустингу (Gradient-boosted tree regression);
    - регресія виживання (Survival regression);
    - ізотонічна регресія (Isotonic regression);
- базова статистика:
    - Підсумкова статистика (максимум, мінімум, середнє значення, дисперсія, кількість заповнених значень, а також загальна кількість);
    - Кореляції між декількома рядами за допомогою двохметодів – співвідношення Пірсона та Спірмена;

- Розшарована вибірка, включаюча sampleBykey та sample-ByKeyExact;
- Перевірка гіпотез за допомогою критерію Пірсона та хі-квадрату на те, що результат є статистично значущим чи випадковим;
- Генерація випадкових даних за допомогою методів RandomRDD, Normal та Poisson;

- класифікація:
  - Логістична регресія (Logistic regression);
  - Біноміальна логістична регресія (Binomial logistic regression);
  - Поліноміальна логістична регресія (Multinomial logistic regression);
  - Класифікатор дерева рішень (Decision tree classifier);
  - Класифікатор випадкового лісу (Random forest classifier);
  - Класифікатор дерева з градієнтним бустингом (Gradient-boosted tree classifier);
  - Багатошаровий персептрон класифікатор (Multilayer perceptron classifier);
  - Лінійна опорна векторна машина (Linear Support Vector Machine);
  - Класифікатор «Один проти усіх» (One-vs-Rest classifier);
  - Байєсова класифікація (Naive Bayes);
- кластеризація:
  - K-середніх (K-means);
  - Латентне розміщення Діріхле (Latent Dirichlet allocation, LDA);
  - Бісектриса k-середніх (Bisecting k-means);
  - Модель гаусової суміші (Gaussian Mixture Model, GMM).
- зменшення розмірності:

  а) Алгоритми пошуку ознак:
  - TF-IDF;
  - Word2Vec;
  - CountVectorizer;
  - FeatureHasher.

б) Алгоритми вибору ознак:

- VectorSlicer;
- RFormula;
- ChiSqSelector.

- оптимізація:
  - Градієнтний спуск (Gradient descent);
  - Стохастичний градієнтний спуск (Stochastic gradient descent, SGD);
  - Алгоритм BFGS з обмеженою пам'яттю (Limited-memory BFGS, L-BFGS).

Регресія – це алгоритм машинного навчання, який можна навчити прогнозувати реальні нумеровані виходи; наприклад температура, ціна акцій тощо. Регресія базується на гіпотезі, яка може бути лінійною, квадратичною, поліноміальною, нелінійною. Гіпотеза – це функція, що ґрунтується на деяких прихованих параметрах та вхідних значеннях.

Класифікація – це процес прогнозування класу даних точок даних. Класи іноді називають мітками або категоріями. Класифікаційне прогностичне моделювання – завдання наближення функції відображення (f) від вхідних змінних (X) до дискретних вихідних змінних (y).

Кластеризація – це задача групування набору об'єктів таким чином, щоб об'єкти в одній і тій же групі мали більше схожих характеристик між собою, ніж об'єкти в інших групах, що є основним завдання інтелектуального аналізу даних і загальною методикою статистичного аналізу даних, яка використовується в багатьох областях, включаючи машинне навчання, розпізнавання образів, аналіз зображень, пошук інформації тощо.

Зменшення розмірності – процес зменшення числа розглянутих випадкових величин шляхом отримання набору головних змінних. Процес пошуку ознак починається з початкового набору даних вимірювань і виявлення похідних ознак, призначених для полегшення наступних етапів навчання і узагальнення.

Оптимізація – це задача вибору кращого елемента з набору доступних альтернатив, враховуючи деякий критерій. Оптимізація включає в себе пошук

«найкращих доступних» значень деякої цільової функції для заданої області (або вхідних даних), включаючи різні типи цільових функцій і різні типи доменів.

РОЗДІЛ 2. МОДЕЛІ МАШИННОГО НАВЧАННЯ НА AZURE HDINSIGHT

Швидка еволюція Spark і швидке зростання попиту на цей фреймворк перевищують можливості розробників та фахівців з розгортання систем на здійснення обгрунтованих компромісів між різними конструкціями системи, складами робочого навантаження, оптимізаціями конфігурацій, версіями програмного забезпечення тощо. Проектувальники його основних та багаторівневих можливостей не можуть легко оцінити, наскільки масштабними можуть бути потенційні наслідки змін параметрів при плануванні та визначенні пріоритетів при розробці програмного забезпечення. Тому, для полегшення цієї задачі були розроблені бенчмарки, за допомогою яких можно оцінити продуктивність системи для конкретних задач.

2.1. Аналіз бенчмарків для Apache Spark

Набори тестів на ефективність Spark розроблені для того, щоб бути частиною тестування технічних рішень. Вони спрямовані на створення специфічного, всебічного та репрезентативного набору робочих навантажень, що охоплюють широкий спектр типів додатків, що успішно реалізуються в екосистемі Spark. Хоча наборів тестів велика кількість, більшість з них охоплюють невелику кількість можливостей Spark, вони далеко не включають всебічне охоплення повного набору типів додатків, що підтримуються в Spark.

2.1.1. SparkBench

SpackBench - це різноманітний набір додатків для тестування різних ресурсів кластера та показників продуктивності, що дозволяють користувачам кількісно порівнювати між різними реалізаціями Spark, різними середовищами або конфігураціями кластера. Він всебічно охоплює репрезентативні навантаження, які зараз підтримує Spark. В таблиці 3 показано класифікацію навантаження на чотири категорії, включаючи додатки машинного навчання, що підтримуються MLlib of Spark, програми для обчислення графіків, які працюють на GraphX of Spark, програми SQL, що обслуговується за допомогою Hive on top of Spark, а

також початкова служба SQL Spark та потокові додатки на платформі DStream of Spark.

Таблиця 3 – SparkBench навантаження

Тип додатку	Навантаження при роботі
Машинне навчання	Логістична регресія Метод опорних векторів Матрична факторизація
Обчислення для графах	Ранг сторінки (PageRank) Сингулярне розкладання величини (SVD++) Трикутний граф (TriangleCount)
SQL запити	Hive RDDRelation
Стрімінгові додатки	Twitter Перегляд Сторінки (PageView)

2.1.1.1. Машинне навчання

Набір для машинного навчання включає 3 основні групи тестів: логістичну регресію, метод опорних векторів (SVM) та матричну факторизацію (MF). Вони широко використовують регресію, алгоритми класифікації та рекомендацій для вирішення задач машинного навчання. Логістична регресія, як класифікатор машинного навчання, може використовуватися для прогнозування безперервних або категоричних даних. Наприклад, його використовують для прогнозування наявності у пацієнта раку на основі вимірюваних характеристик, таких як різні аналізи крові, історія сімейних захворювань, вік і стать. Алгоритм використовує стохастичний градієнтний спуск для підготовки класифікаційної моделі. Набір вхідних даних зберігається в пам'яті за допомогою абстракцій RDD, а вектор параметрів обчислюється, оновлюється та транслюється в кожній ітерації. Методи опорних векторів (МОВ) навчають моделі, будуючи набір гіперпланів у високому або навіть нескінченному розмірному просторі для класифікації. У порівнянні з лінійною та логістичною класифікацією, МОВ можуть неявно відображати входи у простір з високими розмірами та ефективно вести нелінійні класифікації.

Матричну факторизацію, яку зазвичай використовують у рекомендаційних системах, - це метод спільної фільтрації, який заповнює пропущені записи матриці асоціації елементів користувача. MF in Spark на даний момент підтримує спільну фільтрацію на основі моделей і може бути налаштована на використання явних або неявних зворотних зв’язків від користувачів.

2.1.1.2. Обчислення на графах

Ранг сторінки (PageRank), Сингулярне розкладання величини (SVD++) та Трикутний граф (TriangleCount) є репрезентативними та популярними алгоритмами обчислення графів. Інтенсивне споживання ресурсів цих трьох алгоритмів також може допомогти вивчити вузькі місця в продуктивності Spark. PageRank - це перший алгоритм, який використовується веб-пошуковою системою Google для ранжирування сторінок шляхом вимірювання важливості сторінок веб-сайту на основі кількості та якості посилань на сторінку. SVD++ - це модель спільної фільтрації, яка враховує як явні, так і неявні відгуки користувачів та покращує якість рекомендацій. Алгоритм оновлює терміни зміщення та вектори ваги кожного краю під час кожної ітерації. TriangleCount - основний алгоритм аналітики графів, який підраховує кількість трикутників у графі. Його зазвичай використовують у складних графічних програмах реального світу, зокрема виявлення спаму та розкриття прихованих тематичних структур у графах веб-сторінок та посилань. Завдяки інтенсивному навантаженню обчислень, його можна використовувати для виявлення межі продуктивності системи. Як алгоритми машинного навчання, так і алгоритми обчислення графіків складаються з різної кількості етапів залежно від кількості ітерацій, які виконує кожен алгоритм. Spark кешує дані в RDD, щоб уникнути операцій вводу-виводу для диска. Однак налаштування потрібного розміру пам'яті для RDD не є легким завданням, і оптимальні значення залежать від конкретних додатків.

2.1.1.3. SQL Engine

SQL продовжує залишатися стійкою мовою запитів завдяки своєму поширенню, широкій екосистемі інструментів, які її підтримують, а також здатності розвиватися та підтримувати нову базову інфраструктуру та нові вимоги, такі як

вдосконалена аналітика та вибірки даних. Однією з областей, де можна гарантувати інший підхід, є побудова запитів. Історично різні постачальники пропонували запити, що поєднували в собі різні конструкції SQL-обробки, такі як еталони TPC-D/H/DS. Був хороший аналіз набору запитів TPC-H [8]. В запропонованих тестах вводиться набір елементарних або атомних запитів, які оцінюють основні властивості сканування, агрегації та об'єднання, а потім набір проміжних запитів, що підвищують навантаження як на оптимізатора запитів, так і на двигун виконання, і, нарешті, деяких складних і дуже складних запитів, що представляють концепції ROLAP та розширену аналітичну обробку.

2.1.1.4. Потокові програми

Потокове передавання є головною перевагою Spark перед Hadoop. В тесті обирається дві потокові програми, одна, що обробляє популярні тег у соціальній мережі Twitter, а інший - реалізує PageView. Популярний тег Twitter отримує дані з веб-сайту Twitter через бібліотеку Java Twitter, Twitter4j [26], і обчислює найбільш популярні теги кожні 60 секунд. PageView - це програма для потокової передавачі, яка отримує синтетично створені користувацькі кліки та підраховує різні статистичні дані, такі як кількість активних користувачів та кількість переглядів сторінки за 60 секунд.

Окрім ретельно підібраних десяти робочих навантажень, SparkBench визначає ряд показників, що полегшують користувачам порівняння між різними варіантами оптимізації, конфігураціями та налаштуваннями кластерів. SparkBench в даний час повідомляє про час виконання завдання (секунди), вимірюючи час виконання завдання кожного робочого навантаження, швидкість обробки даних (МБ/сек), що визначається як розмір вхідних даних, поділений на загальний час виконання завдання. Час виконання є найважливішим показником ефективності системи Spark. Будь-яка оптимізація Spark повинна знизити середній час виконання роботи на Spark, щоб вимагати підвищення продуктивності кластеру. Швидкість обробки даних відображає можливості обробки даних системи Spark. В майбутньому SparkBench планує збирати і додаткові показники,

такі як розмір даних, що передаються, розмір вхідних/вихідних даних, середнє споживання ресурсів, що дозволяє користувачам глибше розуміти навантаження.

2.1.2. BigBench

BigBench – бенчмарк, що пропонує комплексний набір аналітичних систем для тестування на великому об’ємі даних. Щоб відповідати потребам орієнтиру Big Data і дозволити ефективно порівнювати продуктивність різних систем для Big Data, BigBench зосереджується на трьох основних характеристиках: об’єм, різноманітність та швидкість. Крім того, генератор даних цього бенчмарку може бути використаний для генерації великого об’єму даних для заданої моделі.

Оскільки специфікація BigBench є загальною та техногенною, вона повинна бути реалізована спеціально для кожної системи Big Data. Початкова реалізація BigBench була зроблена для платформи Teradata Aster. Це було зроблено в синтаксисі Ster-MR Aster, що подається у додатковому описі англійською мовою – як початкова специфікація навантажень BigBench. В загальній реалізації, BigBench працює за допомогою Hadoop, використовує MapReduce та інші компоненти на зразок Hive, Mahout та OpenNLPз екосистеми Hadoop. Підсумовуючи це, BigBench складається з моделі даних, що зображена на рис. 13, генератору даних та специфікацію навантажень.

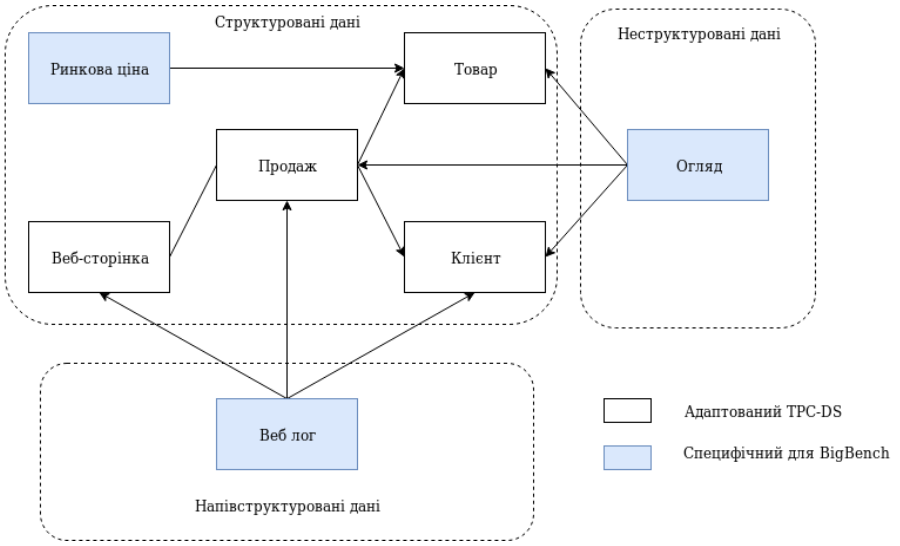

Рисунок 13 – Схема роботи BigBench

Рисунок 13 показує, як BigBench реалізує властивість різноманітності Big Data. Це досягається шляхом класифікації моделі даних на три групи: структуровані, напівструктуровані та неструктуровані дані. В якості основної бізнес-моделі використовується вигаданий роздрібний продавець товарів. Ділова модель та значна частина структурованої частини моделі даних походить від еталону TPC-DS .

Генератор даних базується на розширенні PDGF і дозволяє генерувати дані відповідно до моделі даних BigBench, включаючи структуровані, напівструктуровані та неструктуровані частини. Генератор даних може масштабувати кількість даних на основі коефіцієнту масштабування. Завдяки паралельному генеруванню даних, створення набору даних відбувається ефективно для великої кількості факторів моделі. Таким чином, в BigBench реалізується об’ємна властивість Big Data. Крім того, велика швидкість обробки Big Data реалізується періодичною схемою оновлення, яка постійно додає нові дані до різних таблиць моделі даних.

Крім того, BigBench може працювати і на Spark, оскільки Spark SQL підтримує HiveQL, запити типу “Pure HiveQL” можна успішно перенести та виконати на Spark.

2.1.3. Spark-perf

Spark-perf – це фреймворк для тестування продуктивності для Apache Spark 1.0+. Він написаний на мовах Python та Scala. Spark-perf розроблений компанією DataBricks для перевірки продуктивності MLlib, бібліотеки, що реалізує алгоритми машинного навчання, з можливістю також тестувати потокові додатки, SQL-запити та ядро Spark (Spark Core тести), що наразі розробляється.

Головними перевагами Spark-perf перед іншими фреймворками є:

- можливість тестування на ефективність для Spark, PySpark, Spark Streaming та MLlib;
- параметризовані конфігурації тесту;
- легкість у конфігуруванні (файл конфігурації написаний мові на Python);

- має набори параметрів, щоб перевірити одразу на декількох версіях Spark і конфігурацій тестів;
- автоматичне завантаження та створення Spark, що забезпечення кешування успішних збірок для швидкого тестування на кількох версіях Spark.

Spark-perf має реалізацію таких алгоритмів MLlib, як:

- glm-regression: Узагальнена модель лінійної регресії;
- glm-classification: Узагальнена модель лінійної класифікації;
- naive-bayes: Наївний баєсів класифікатор;
- naive-bayes-bernoulli: Модифікований Бернулі наївний баєсів класифікатор;
- decision-tree: Дерево рішень;
- als: метод найменших квадратів, що чергуються;
- kmeans: K-Means кластерізація;
- gmm: Модель змішування Гаусса;
- svd: Сингулярне розкладання величини;
- pca: Аналіз основних компонентів;
- summary-statistics: Зведена статистика (min, max, ...);
- block-matrix-mult: Матричне множення;
- pearson: Кореляція Пірсона;
- spearman: Кореляція Спірмена;
- chi-sq-feature/gof/mat: Хі-квадрат тести;
- word2vec: Word2Vec розподілене подання слів;
- fp-growth: Частотні набори елементів FP-зростання;
- python-glm-classification: Узагальнена модель лінійної класифікації на Python;
- python-glm-regression: Узагальнена модель лінійної регресії на Python;
- python-naive-bayes: Наївний баєсів класифікатор на Python;
- python-als: метод найменших квадратів, що чергуються на Python;
- python-kmeans: K-Means кластерізація на Python;
- python-pearson: Кореляція Пірсона на Python;

- python-spearman: Кореляція Спірмена на Python;

2.2. Моделі машинного навчання у бенчмарку Spark-perf

В таблиці 4 наведено інформацію про характеристики тестів машинного навчання у пакеті Spark-Perf.

Таблиця 4 – Характеристики тестів машинного навчання бенчмарку Spark-Perf

Кодова назва	Назва тесту	Кількість випробувань	Кількість тестових примірників
Алгоритми регресії та класифікації			
glm-regression	Generalized Linear Regression Model	10	5000
glm-classification	Generalized Linear Classification Model	10	5000
naive-bayes	Naive Bayes	10	5000
Алгоритми з деревом			
decision-tree	Decision Tree	16 x 10	5000
Алгоритми рекомендацій			
als	Alternating Least Squares	10	500
Алгоритми кластеризаії			
kmeans	K-Means clustering	10	5000
lda	Latent Dirichlet allocation	10	
pic	Power Iteration Clustering	10	
gmm	Gaussian Mixture Model	10	5000
Статистичні алгоритми			
summary-statistics	Summary Statistics	10	
pearson	Pearson's Correlation	10	50000
spearman	Spearman's Correlation	10	50000
chi-sq-feature/gof/mat	Chi-square Tests	10	100000
Алгоритми лінійної алгебри			
svd	Singular Value Decomposition	10	
pca	Principal Component Analysis	10	
block-matrix-mult	Matrix Multiplication	10	
Алгоритми пошуку асоціативних правил			
fp-growth	FP-growth frequent item sets	10	
prefix-span	PrefixSpan	10	
Алгоритми пошуку ознак			
word2vec	Word2Vec distributed presentation of words	10	

2.2.1. ALS (метод найменших квадратів, що чергуються)

Алгоритм найменших квадратів, що чергуються (ALS) розподіляє задану матрицю R на два коефіцієнти U і V, такі що R≈UTV. Невідомий розмір рядка задається як параметр алгоритму і називається прихованим коефіцієнтом. Оскі-

льки матрична факторизація може використовуватися в контексті рекомендацій, матриці U і V можна назвати відповідно користувачем і матрицею елементів. I-й стовпчик матриці користувача позначається Ui, а i-й стовпець матриці елемента - Vi. Матрицю R можна назвати матрицею оцінок з Ri,j=ri,j.

Для того, щоб знайти матрицю користувача та елемента, вирішується наступна проблема:

$$arg\ min(U,V) \sum_{\{i,j|r_{i,j}\neq 0\}} (r_{i,j} - u_i^T v_j)^2 + \lambda(\sum_i n_{u_i}||u_i||^2 + \sum_j n_{v_j}||v_j||^2)$$

де λ - коефіцієнт регуляризації, nui - кількість елементів, які оцінив i-ий користувач, а nvj - кількість разів, яку оцінювали j-ий елемент. Ця схема регуляризації, щоб уникнути перевитрати, називається зваженою λ-регуляризацією.

Закріплюючи одну з матриць U або V, ми отримуємо квадратичну форму, яку можна вирішити безпосередньо. Рішення модифікованої проблеми гарантується монотонним зниженням загальної функції витрат. Застосовуючи цей крок поперемінно до матриць U та V, ми можемо ітеративно вдосконалити матричну факторизацію.

Матриця R задається в її розрідженому поданні як кортеж (i, j, r), де i позначає індекс рядка, j індекс стовпця і r - значення матриці в положенні (i, j).

2.3. Розробка інформаційного забезпечення

Для аналізу результатів, які отримуються у результаті роботи бенчмарку Spark-perf, потрібно розробити програмне забезпечення. Крім того, як відомо, для нормальної роботи кластеру потрібно моніторити його ресурси. Для цього був розроблений комплекс програмного забезпечення.

2.3.1. Структура вихідної папки з результатами Spark-Perf

Структура папки з лог-файлами з результатами роботи Spark-Perf має вигляд, представлений на рис. 14. Папка включає директорію та один файл для кожного набору тестів, в нашому випадку - тільки для набору MLlib.

Рисунок 14 – Структура папки з лог-файлами

В папці знаходяться для кожного тесту 2 файли, один з розширенням out, інший - err. У файлі з розширенням out, який представлений на рис. 35, є наступна інформація:

- Java options у форматі (параметр=значення) - налаштування віртуальної машини Java;
- Options у форматі (параметр=значення) - інші налаштування (Spark);
- Інформація про виконання тесту (напраклад, кількість прикладів при навчанні) у форматі (назва: 1000). Також тут може бути відладочна інформація;
- Scheduling mode = FIFO
- Spark Context default degree of parallelism = 32
- Інформація зі SparkMeasure у форматі (параметр => значення)
- Об'єкт результатів у форматі JSON

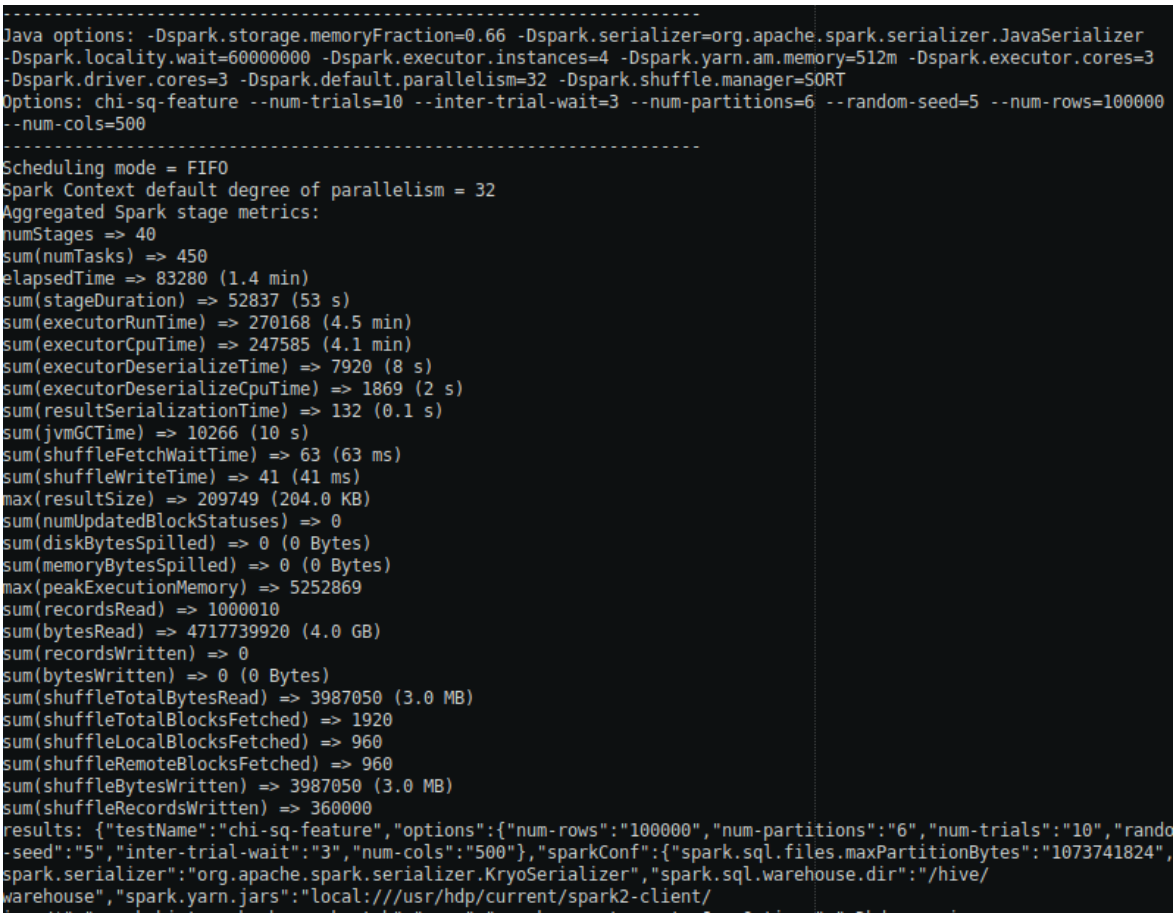

Рисунок 15 – Приклад лог-файлу з розширенням out

У об’єкті, який представлений на рис. 15, результату міститься наступна інформація:

- options: структура (може бути порожнім):
  - closure-size: рядок (може бути порожнім);
  - inter-trial-wait: рядок (може бути порожнім);
  - key-length: рядок (може бути порожнім);
  - num-jobs: рядок (може бути порожнім);
  - num-partitions: рядок (може бути порожнім);
  - num-records: рядок (може бути порожнім);

  [...]
- results: масив (може бути порожнім):
  - element: структура (не може бути порожнім):
    - time: число з рухомою комою (може бути порожнім);
    - trainingMetric: число з рухомою комою (може бути порожнім);
    - testTime: число з рухомою комою (може бути порожнім);

- recommendUsersForProductsTime: число з рухомою комою (може бути порожнім);
- recommendProductsForUsersTime: число з рухомою комою (може бути порожнім);
- testMetric: число з рухомою комою (може бути порожнім);
- trainingTime: число з рухомою комою (може бути порожнім);

- sparkConf: структура (може бути порожнім):
  - spark_app_id: рядок (може бути порожнім);
  - spark_app_name: рядок (може бути порожнім);
  - spark_driver_host: рядок (може бути порожнім);
  - spark_driver_memory: рядок (може бути порожнім);
  - spark_driver_port: рядок (може бути порожнім);

  [...]
- sparkVersion: рядок (може бути порожнім);
- systemProperties: структура (може бути порожнім):
  - SPARK_SUBMIT: рядок (може бути порожнім);
  - awt_toolkit: рядок (може бути порожнім);
  - file_encoding: рядок (може бути порожнім);

  [...]
- testName: рядок (може бути порожнім).

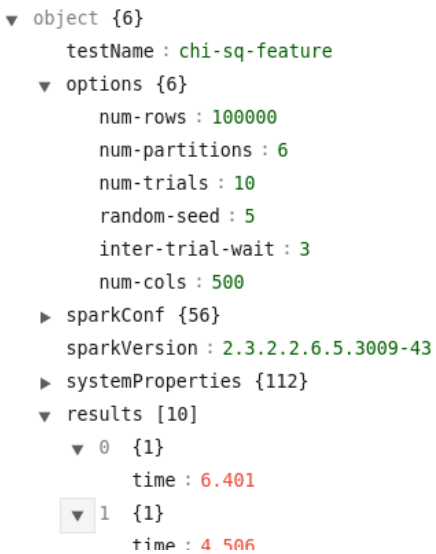

Рисунок 16 – Приклад об’єкту результату

2.3.2. Звіт за результатами Spark-Perf

На рис. 15 представлена діаграма послідовності, яка описує процес отримання результату з результатів тестування та навчання Spark-Perf (модуль SparkPerfReport).

Для збору та обробки результатів тестування, що записані у об’єкті результату, був розроблений на мові Python3.7 програмний модуль Spark-PerfReport. Вхідними даними для програми є шлях до папки з результатами. Далі модуль ітеративно йде по внутрішнім папкам та шукає файли з розширенням .out, розбирає файл, збирає відповідну інформацію так формує звіт у вигляді ex-cel-документу.

У модулі SparkPerfReport до звіту виводиться статистичний аналіз отриманих результатів для кожного тесту, а саме середнє арифметичне значення, медіана, та середньоквадратичне відхилення для тестування та навчання, а також відношення середнього часу тестування до середнього часу навчання. Код SparkPerfReport доступний у репозиторії на GitHub та в додатках А.1-А.3.

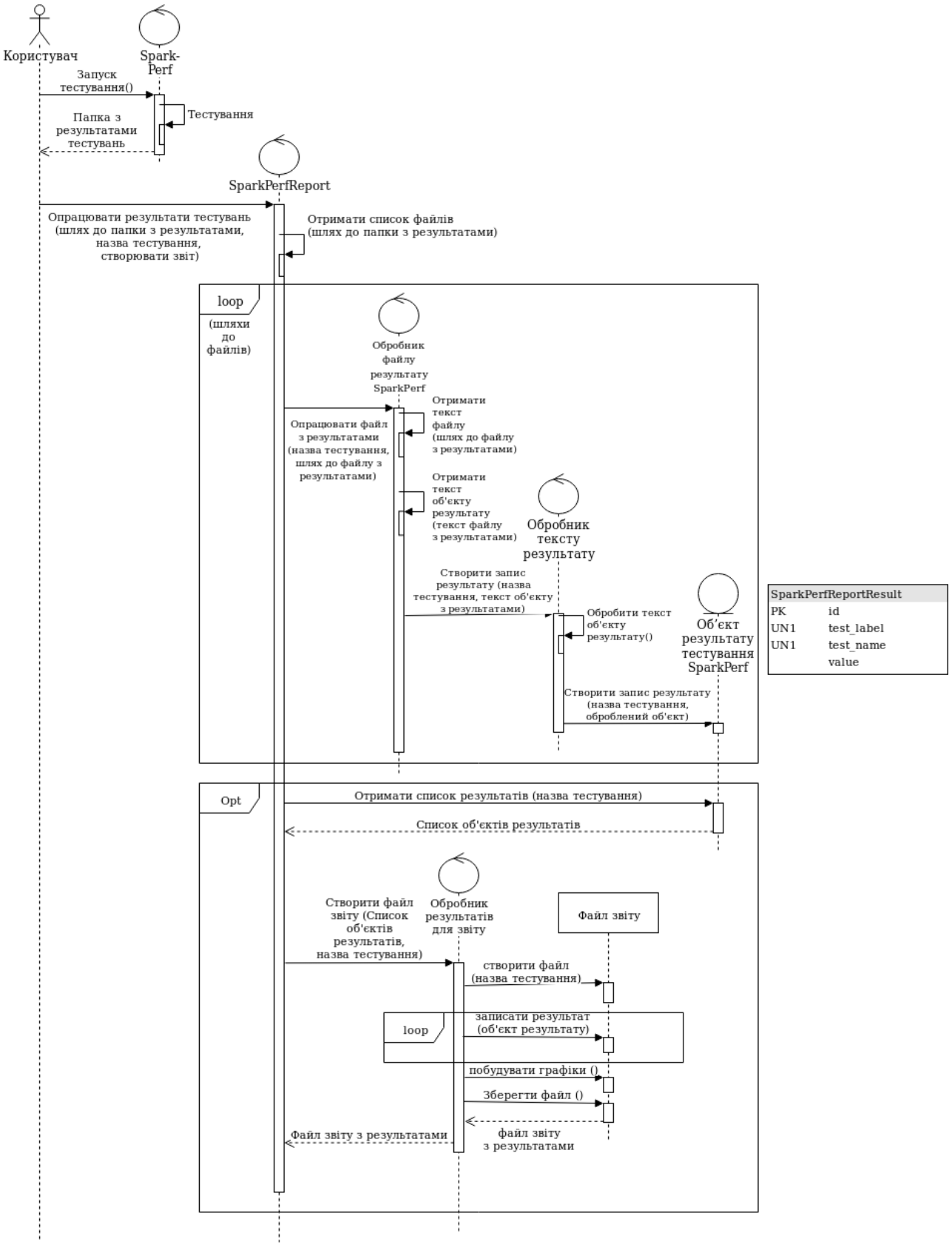

Рисунок 17 – діаграма послідовності для модуля SparkPerfReport

2.3.3. Звіт за результатами SparkMeasure

На рисунку 18 представлена діаграма послідовності, яка описує процес отримання результату з модуля SparkMeasure (модуль SparkMeasureReport).

Вхідні параметри у модулі такі ж, як і у модуля SparkPerfReport, але принцип роботи інший. Поля метрик, які буде опрацьовувати програма, визначені в окремому файлі, а структура таблиці для бази даних, у яку зберігаються наші дані, має інший вигляд. Так як тепер для тесту є декілька метрик - поле з назвою метрики винесений у окреме поле. У модулі SparkMeasureReport до звіту виводиться зведена таблиця з метриками для всіх тестів, а також графіки, побудовані на її основі. Код SparkPerfReport доступний у репозиторії на GitHub та в додатках В.1-В.2. Приклад вихідної таблиці представлений в додатку Г1.

2.3.4. Моніторинг метрик Ambari

Для збору та аналізу метрик для моніторингу ресурсів кластеру використовується комплекс ПЗ, який складається з:

- Ambari;
- Grafana;
- Graphite;
- Ambari2Grafana.

Apache Ambari - це інструмент адміністрування з відкритим вихідним кодом, що розгортається на кластерах Hadoop, і використовується для відстеження запущених програм та їх стану. Apache Ambari можна назвати веб-додатком, який управляє, контролює та забезпечує здоров'я кластерів Hadoop. Він забезпечує високоінтерактивну інформаційну панель, яка дозволяє адміністраторам візуалізувати хід та стан кожної програми, що працює над кластером Hadoop. Його гнучкий і масштабований користувальницький інтерфейс дозволяє встановлювати на кластер цілу низку інструментів, таких як Pig, MapReduce, Hive тощо.

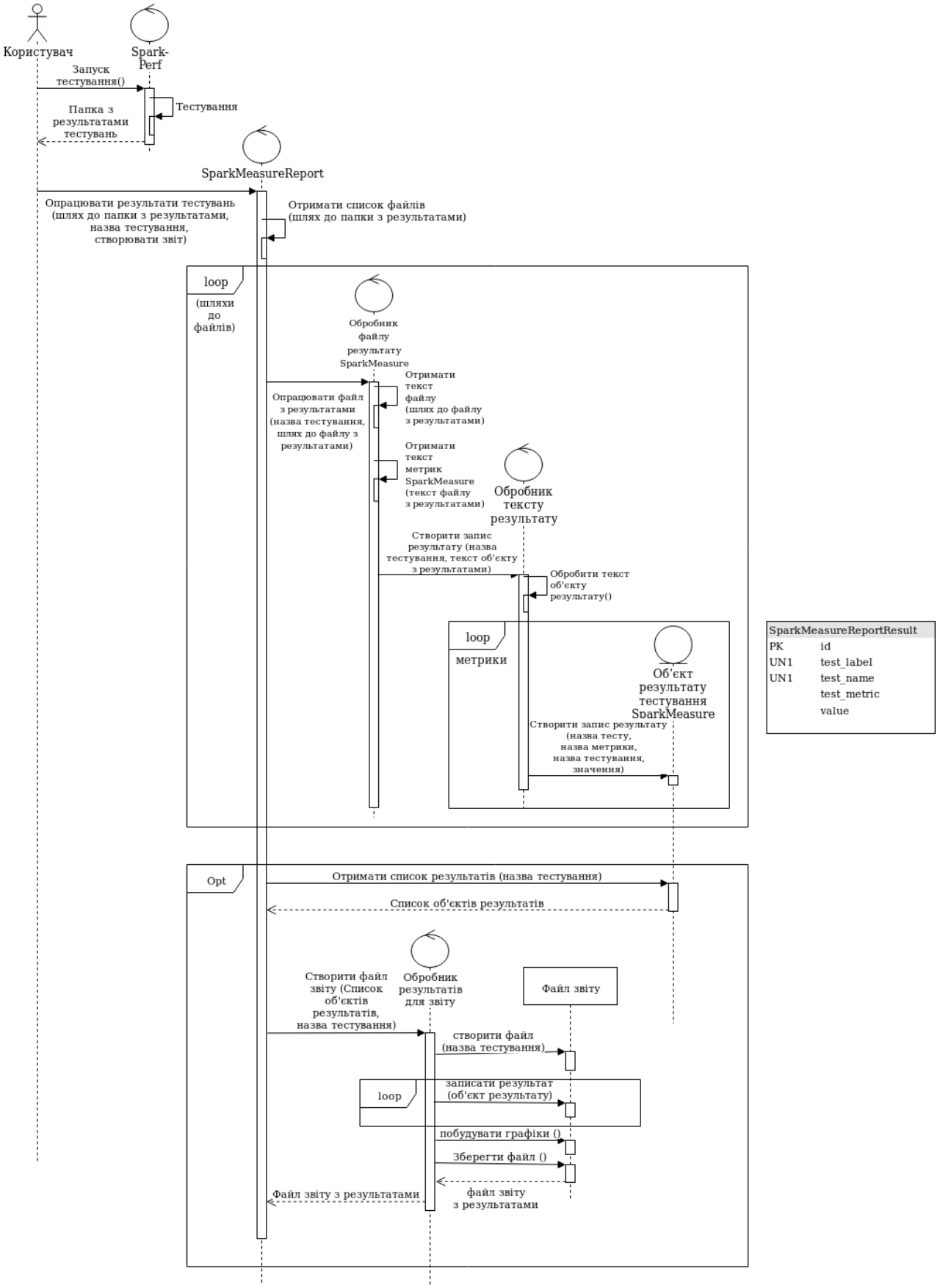

Рисунок 18 – діаграма послідовності для модуля SparkMeasureReport

Основними особливостями Apache Ambari є:

- Можливість моніторингу стану кластеру Hadoop за допомогою попередньо налаштованих оперативних показників;
- Зручна конфігурація та просте покрокове керівництво з конфігурації;
- Моніторинг залежностей та результатів роботи шляхом візуалізації та аналізу завдань;
- Аутентифікація, авторизація та аудит шляхом встановлення кластерів Hadoop на базі Kerberos;
- Програмний інтерфейс (API) для зручної співпраці з зовнішніми програмними продуктами;
- Гнучка та адаптивна технологія, яка ідеально вписується в корпоративне середовище.

Ambari пропонує інтуїтивні та REST програмний інтерфейс, які автоматизують операції в кластері Hadoop. Його стійкий і безпечний інтерфейс дозволяє йому бути досить ефективним в оперативному контролі. На рис. 37 представлена схема роботи Ambari.

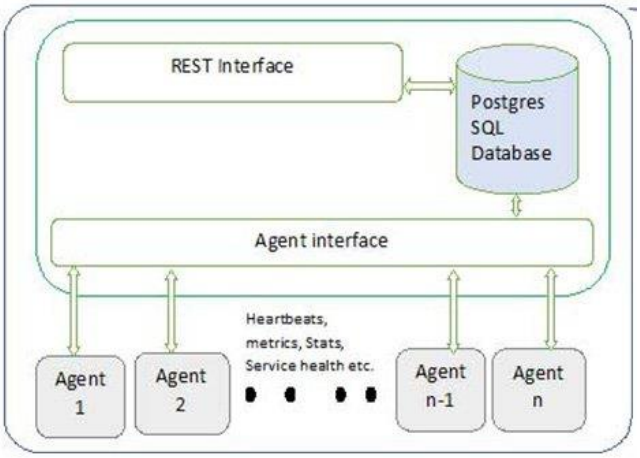

Рисунок 19 – Схема роботи Ambari

Більш детальна архітектура Apache Ambari представлена на рис. 20.

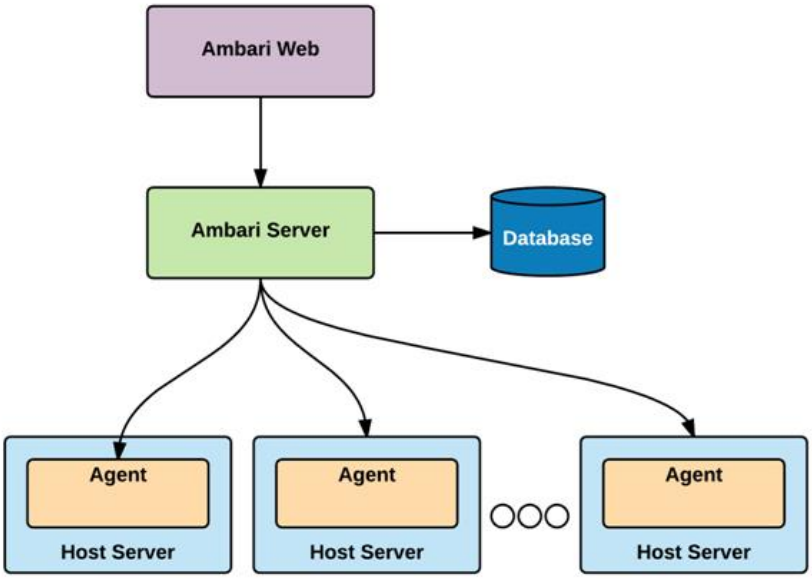

Рисунок 20 – архітектура Ambari

Apache Ambari дотримується архітектури master-slave, де головний вузол дає інструкції веденим вузлам виконувати певні дії та повідомляти про стан кожної дії. Головний вузол відповідає за збір та зберігання стану інфраструктури. Для цього головний вузол використовує сервер бази даних, який можна налаштувати під час налаштування.

Graphite - це безкоштовний інструмент з відкритим кодом, який здійснює моніторинг та візуалізацію числових даних часових рядів, таких як продуктивність комп'ютерних систем. Графіт був розроблений Orbitz Worldwide, Inc та випущений у 2008 році.

Графіт дозволяє збирать, зберігати та відображати дані часових рядів у режимі реального часу.

Інструмент має три основні компоненти:

- Carbon - фонова програма (демон), який слухає дані часових рядів;
- Whisper - проста бібліотека баз даних для зберігання даних часових рядів (за конструкцією схожа на RRD);
- Graphite webapp - веб-додаток на мові Django, який надає графіки на вимогу за допомогою бібліотеки в Cairo. На рис. 38 зображений приклад головної сторінки Graphite webapp.

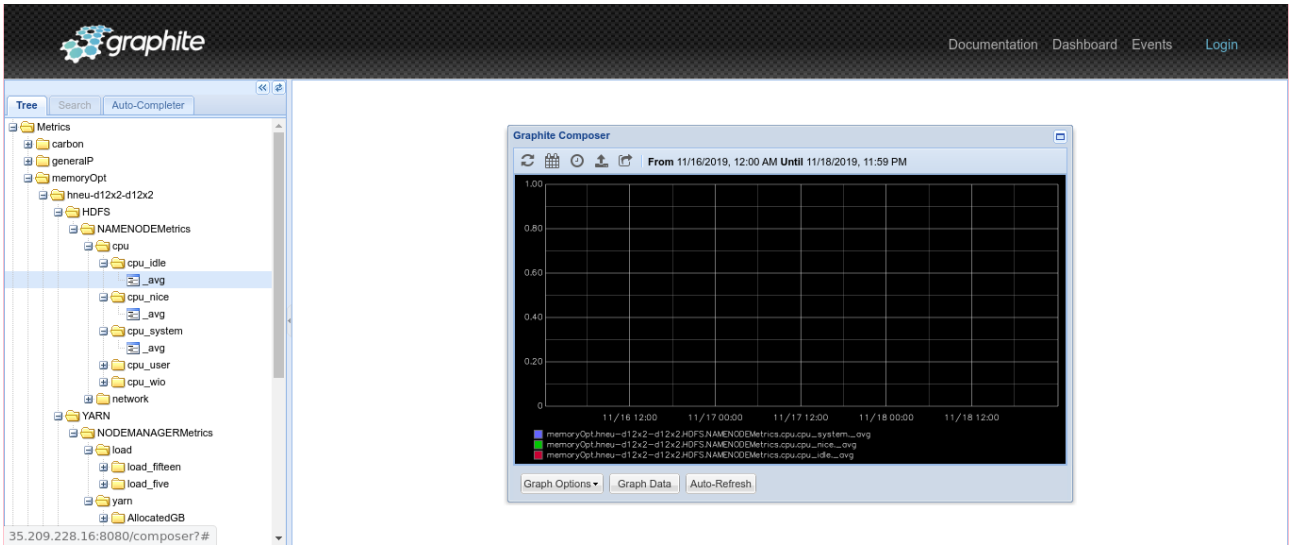

Рисунок 21 – Головна сторінка Graphite webapp

Завантажувати дані у Graphite досить легко, як правило, більша частина зусиль припадає на збір даних для початку. Коли ви надсилаєте точки даних на Carbon, вони стають доступними для графічного відображення у веб-переглядачі, який пропонує декілька способів створення та відображення графіків, включаючи простий API URL для візуалізації, що дозволяє легко вставляти графіки на інші веб-сторінки..

Grafana – це безкоштовне програмне забезпечення, що дозволяє візуалізувати та форматувати метричні дані, яке розповсюджується за ліцензією Apache 2.0.2. Це дозволяє створювати інформаційні панелі та графіки з різних джерел, включаючи бази даних часових рядів, такі як Graphite, InfluxDB та OpenTSDB. Grafana є крос-платформенним додатком і її також можна розгорнути за допомогою Docker. Він написаний на Go і має повний HTTP API.

Окрім керування класичними інформаційними панелями (додавання, видалення, вибірка), Grafana пропонує поділитися поточною інформаційною панеллю, створивши посилання або статичний знімок її. Всі панелі управління та джерела даних прив’язуються до організацією, а користувачі програм пов’язані з організаціями через ролі. Grafana не дозволяє користувачам випадково перезаписати панель управління. Аналогічний захист існує при створенні нової панелі управління, назва якої вже існує.

Схема роботи з цим модулем показана на рис. 22.

Для збору інформації з Ambari у базу даних Graphite використовується програма Ambari2Graphite, яка була розроблена на мові Python3.7. Вхідними даними для програми є файл налаштування, який містить доступи до Ambari та Graphite, дата початку моніторингу та дата початку тестування. Модуль використовує API Ambari для завантаження значень значень за заданим набором метрик. В результаті, створюються файли з метриками (як резервна копія) та дані завантажуються у Graphite за допомогою Carbon. Код Ambari2Graphite доступний у репозиторії на GitHub та в додатку Б.1.

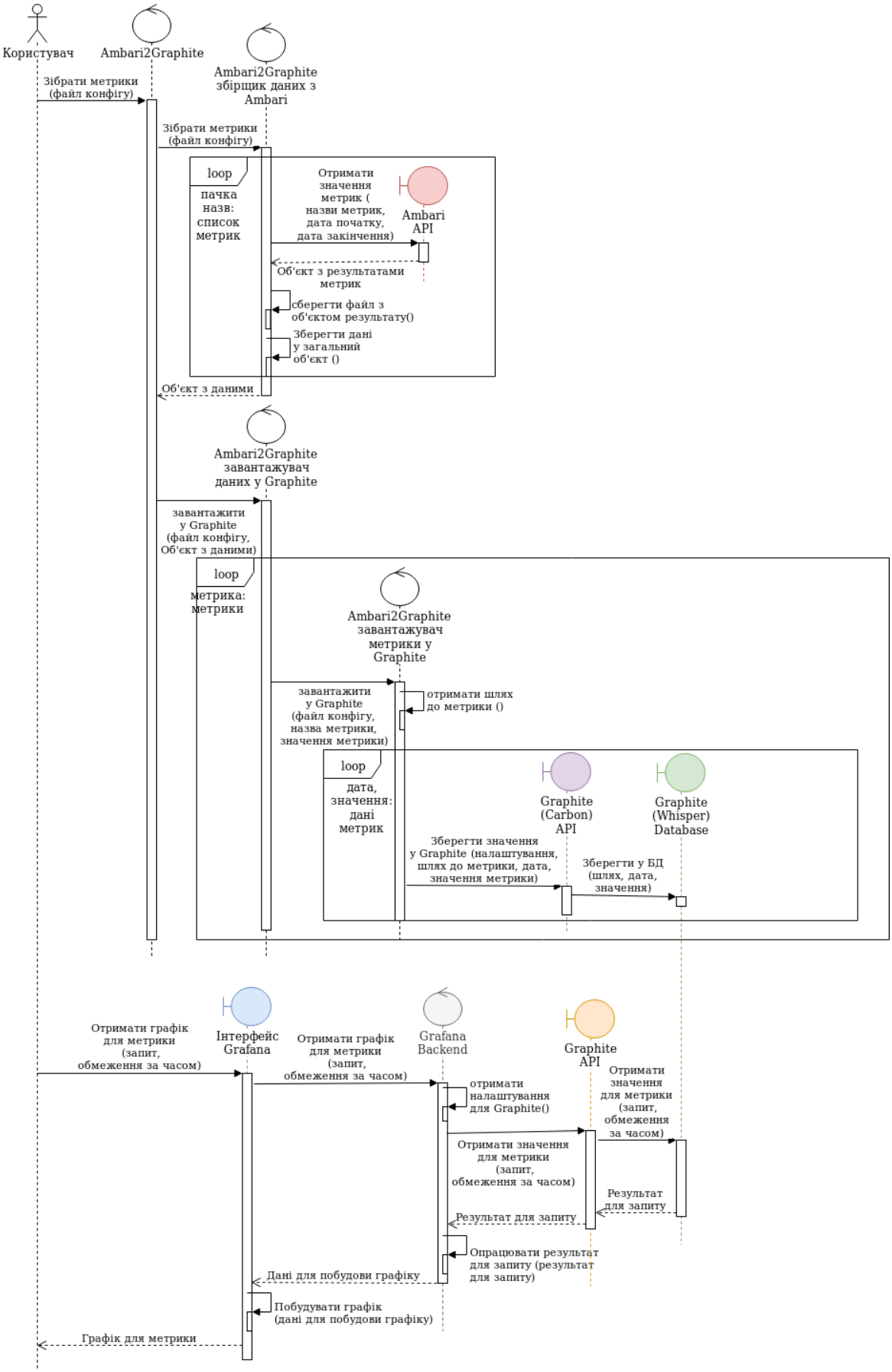

Рисунок 22 – діаграма послідовності модуля моніторингу

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ КЛАСТЕРІВ З ОПТИМІЗОВАНОЮ ПАМ'ЯТТЮ НА ХМАРНІЙ ПЛАТФОРМІ AZURE HDINSIGHT

3.1. Методичне забезпечення

3.1.1. Створення кластеру у Microsoft Azure

На головній сторінці Microsoft Azure Portal (https://portal.azure.com/#home) обираємо пункт «Create a resource» у боковому меню (рис. 23).

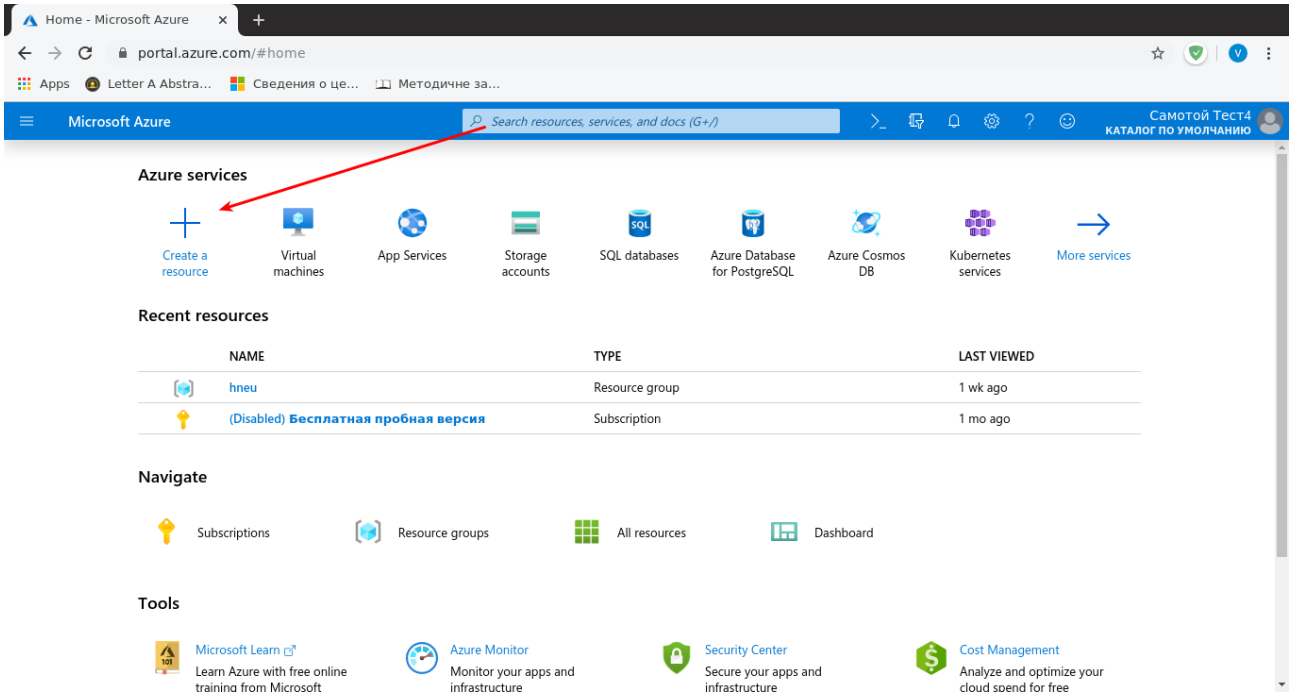

Рисунок 23 – Головна сторінка Microsoft Azure Portal

У полі пошуку ресурсів водимо запит "Template" та обираємо пункт "Template deployment" із запропонованих варіантів (рис. 24).

Шаблони диспетчера Azure Resource Manager дозволяють розгортати ресурси і управляти ними як групою за допомогою їх опису у форматі JSON і їх параметрів розгортання.

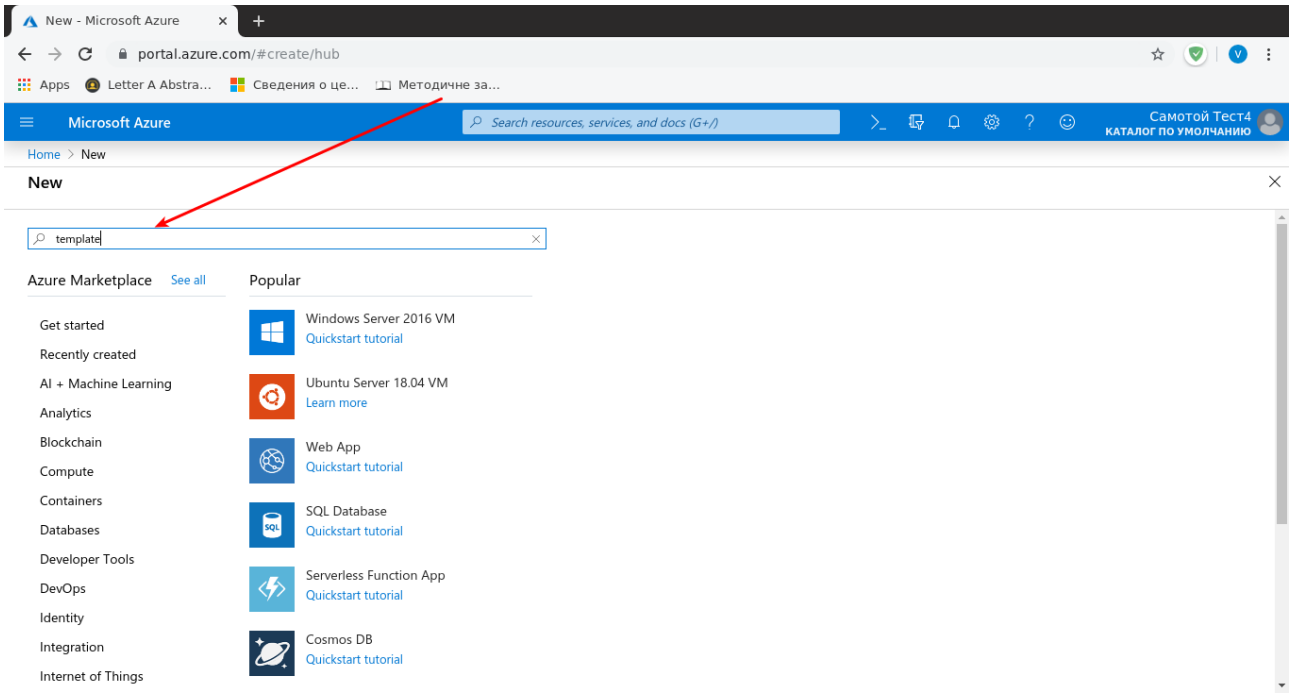

Рисунок 24 – Пошук потрібного ресурсу

Щоб створити новий кластер з шаблону, у відкритій сторінці натаскаємо кнопку “Create” (рис. 25).

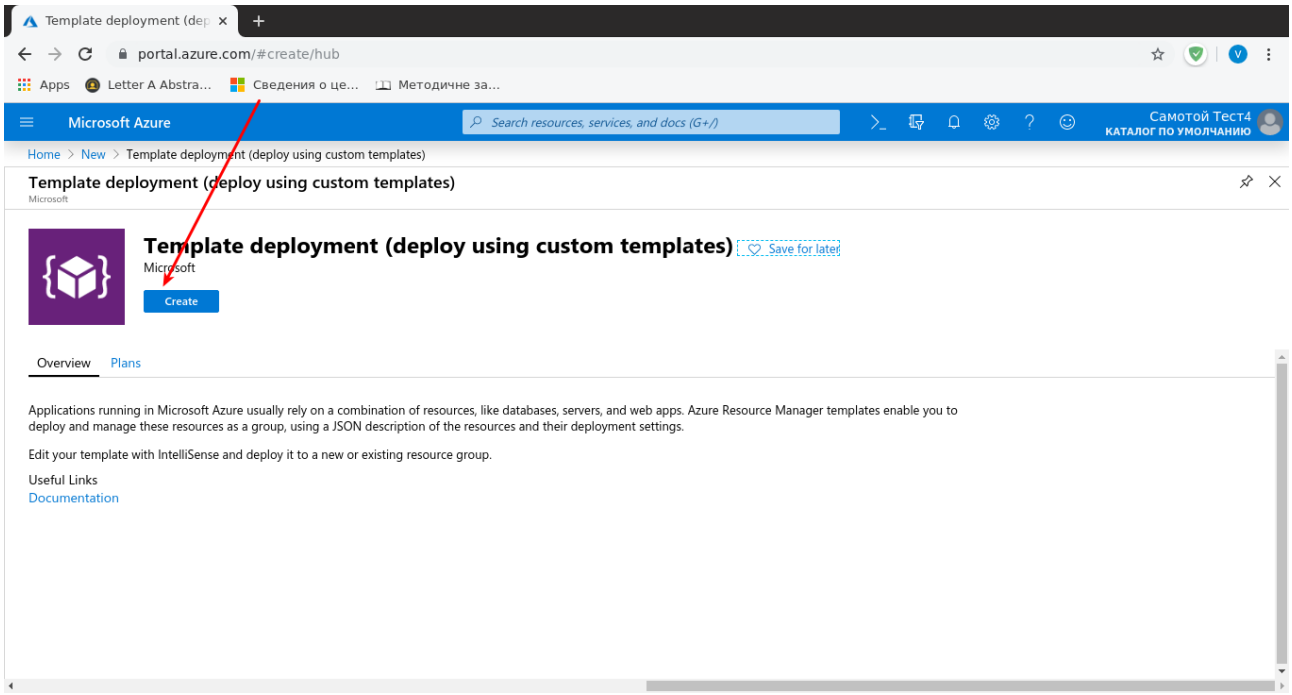

Рисунок 25 – Сторінка з описом опції створення нового ресурсу на основі шаблону

Для того, щоб відредагувати шаблон для розвертання кластеру, натискаємо на кнопку “Edit template” (рис. 26).

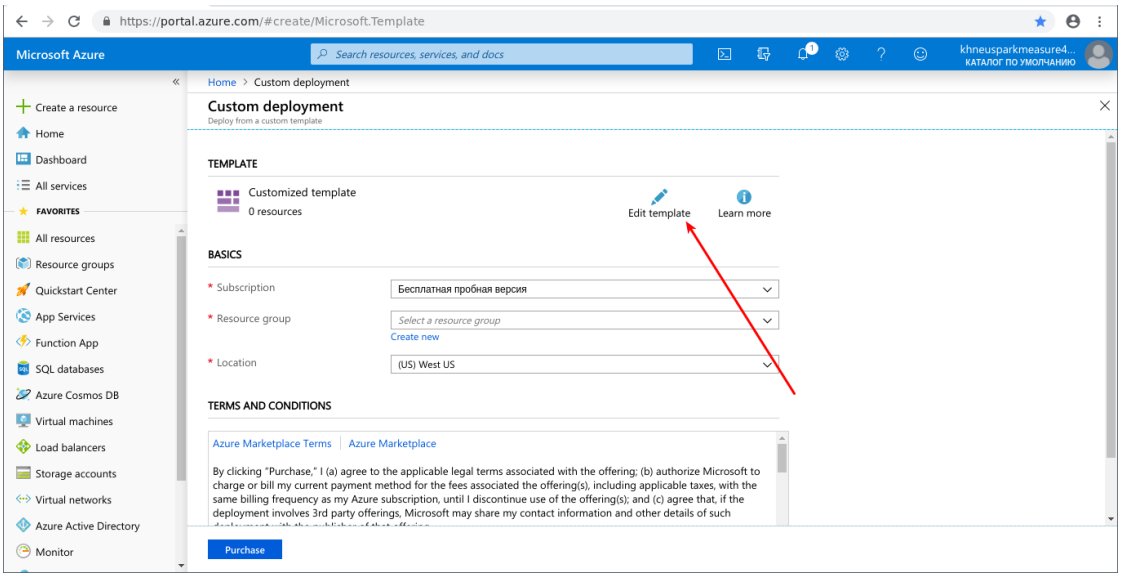

Рисунок 26 – Сторінка створення ресурсу на основі шаблону

Шаблон розгортання можна створити власноруч, завантажити у вигляді файлу у Azure, або обрати один з вже доступних шаблонів. Існує декілька шаблонів для розгортання кластеру Spark у Azure. Для того, щоб побачити всі з них потрібно ввести у поле пошуку шаблонів слово “Spark”.

Нам потрібно обрати шаблон “101-hdinsight-spark-linux” оскільки він має найбільш підходящу конфігурацію (рис. 27). Для того, щоб підтвердити наш вибір натискаємо на кнопку «ОК».

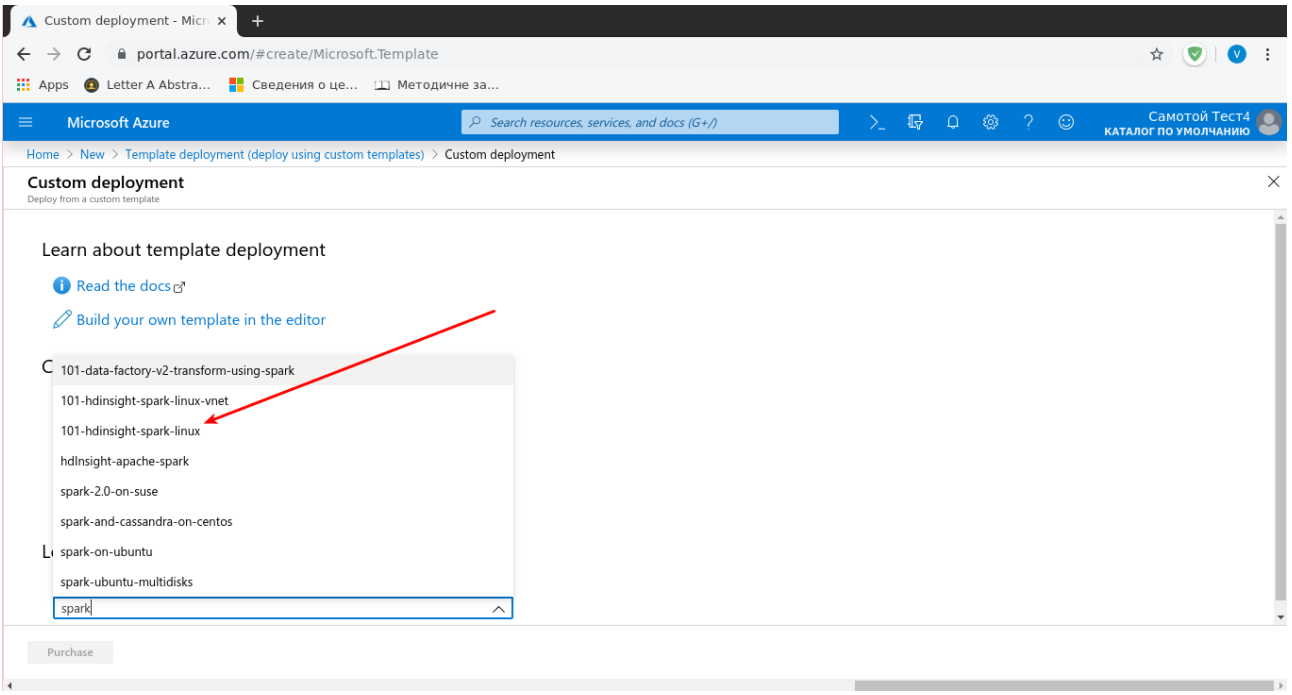

Рисунок 27 – Пошук потрібного шаблону для розгортання кластеру

У обраному шаблоні є багато параметрів, але нам треба звернути увагу на наступні:

- headNode.targetInstanceCount - кількість master-вузлів;
- headNode.hardwareProfile.vmSize - обрана конфігурація розміру віртуальних машин для master-вузлів;
- workerNode.targetInstanceCount - кількість worker-вузлів ;
- workerNode.hardwareProfile.vmSize - обрана конфігурація розміру віртуальних машин для worker-вузлів.

Спочатку налаштовуємо master-вузли (рис. 28). Рекомендована кількість master-вузлів – 2 штуки, конфігурація розміру «Standard_D12_v2» взята для прикладу.

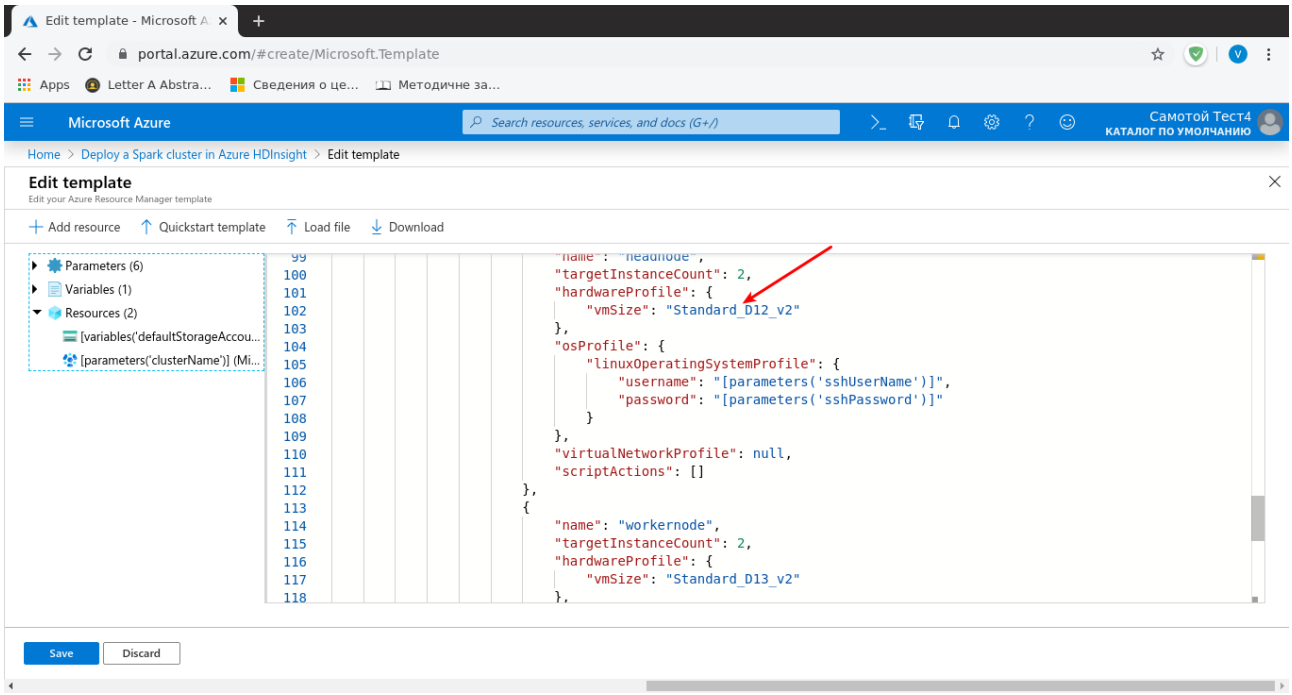

Рисунок 28 – Налаштування master-вузлів

Далі налаштовуємо worker-вузли (рис. 29). Кількість worker-вузлів а також їх конфігурація розміру може варіюватися в залежності від складності задач, що будуть виконуватися. В даному прикладі ми маємо 2 вузли з конфігурацією розміру «Standard_D13_v2».

Після внесення змін до шаблону, натискаємо кнопку “Save”, для того щоб зберегти наші зміни.

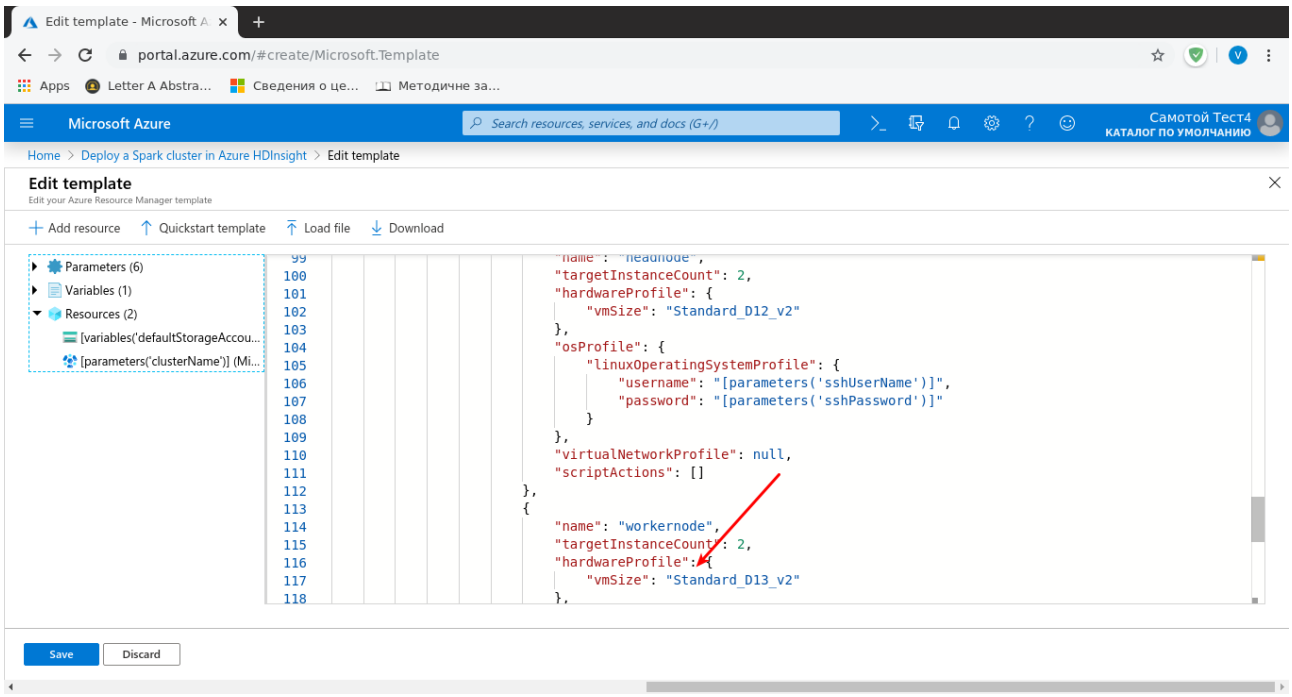

Рисунок 29 – Налаштування worker-вузлів

Далі обираємо регіон для розгортання кластеру, заповнюємо поля для імені кластеру та паролів, підтверджуємо що ми згодні з правилами та натискаємо на кнопку “Purchase” (рис. 30).

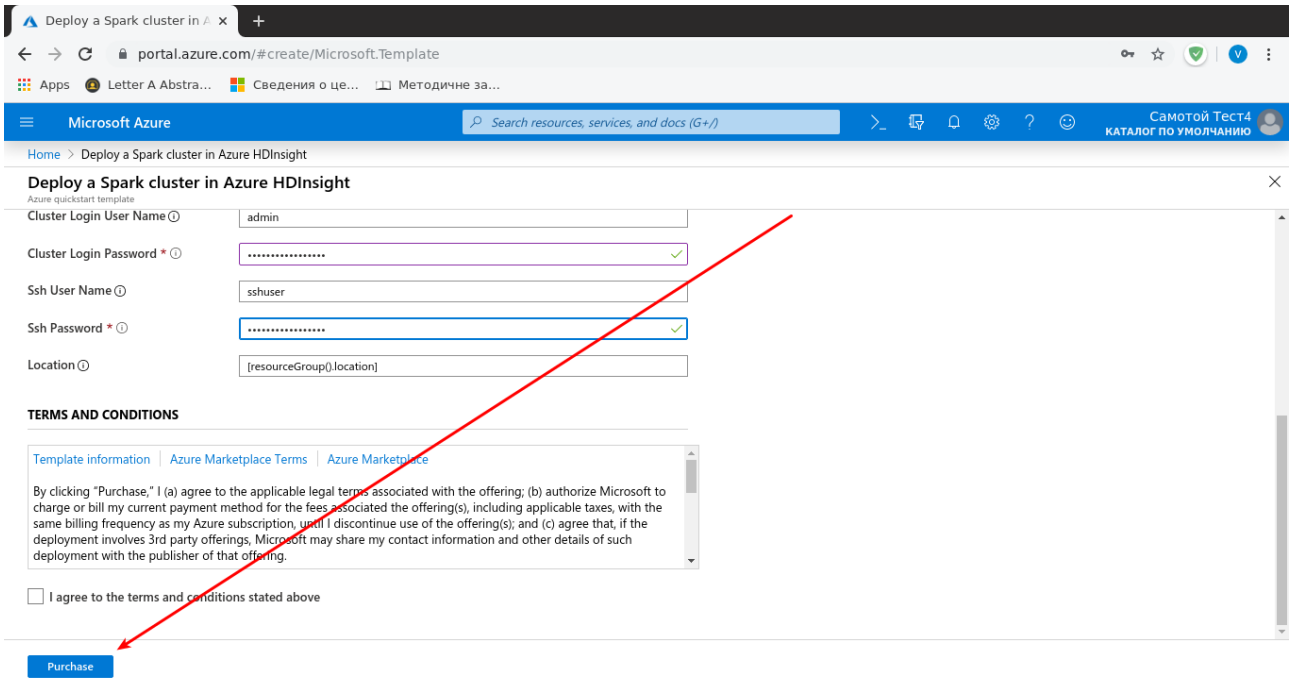

Рисунок 30 – Заповнення необхідних атрибутів та підтвердження придбання ресурсу

Далі потрібно зачекати доки кластер розгорнеться. Зазвичай це займає близько 20 хвилин. Після успішного розгортання біля назви кластеру з’явиться зелена іконка, що інформує про успішне розгортання. Для того, щоб перейти на панелі управління ресурсом, потрібно натиснути на ім’я кластеру (рис. 31).

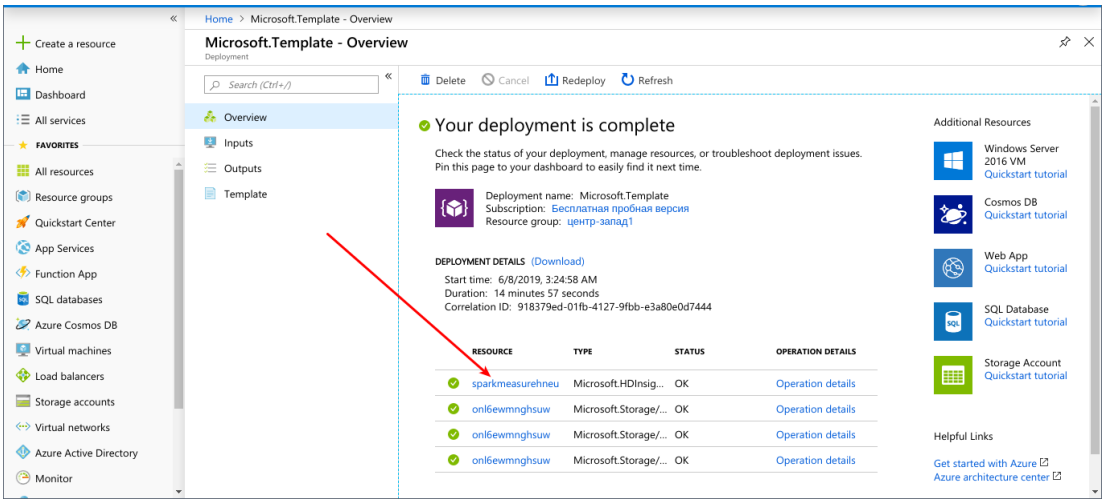

Рисунок 31 – Сторінка зі статусом розгортання ресурсу

На головній сторінці панелі управління ресурсом (рис. 32) доступна базова інформація про розгорнутий ресурс, в нашому випадку кластер HDInsight, а також посилання на пов’язані з кластером ресурси.

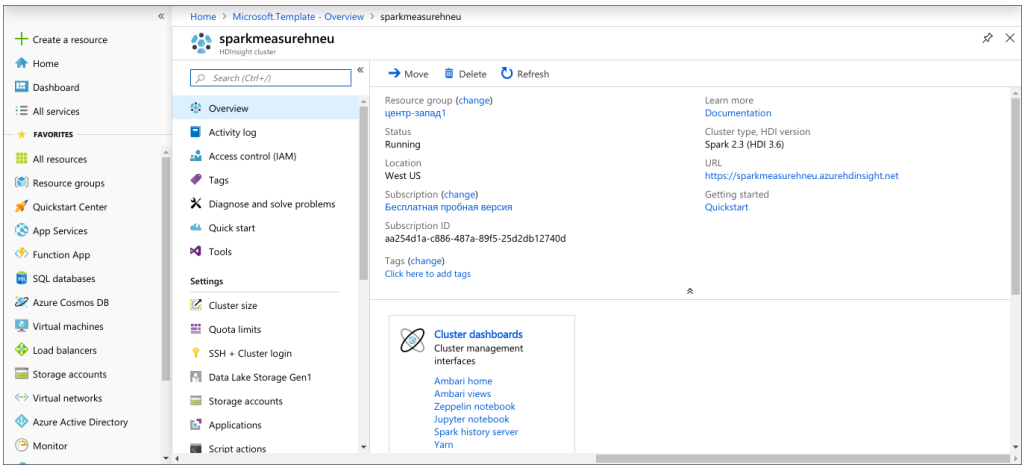

Рисунок 32 – Головна сторінка панелі управління ресурсом

3.1.2. Налаштування кластеру

Для налаштування кластеру в Azure HDInsight необхідно зайти на сторінку Ambari створеного кластеру. Для цього на головній сторінці панелі управління ресурсом в розділі Overview натиснути на посилання в секції URL, або на посилання “Ambari home” (рис. 33).

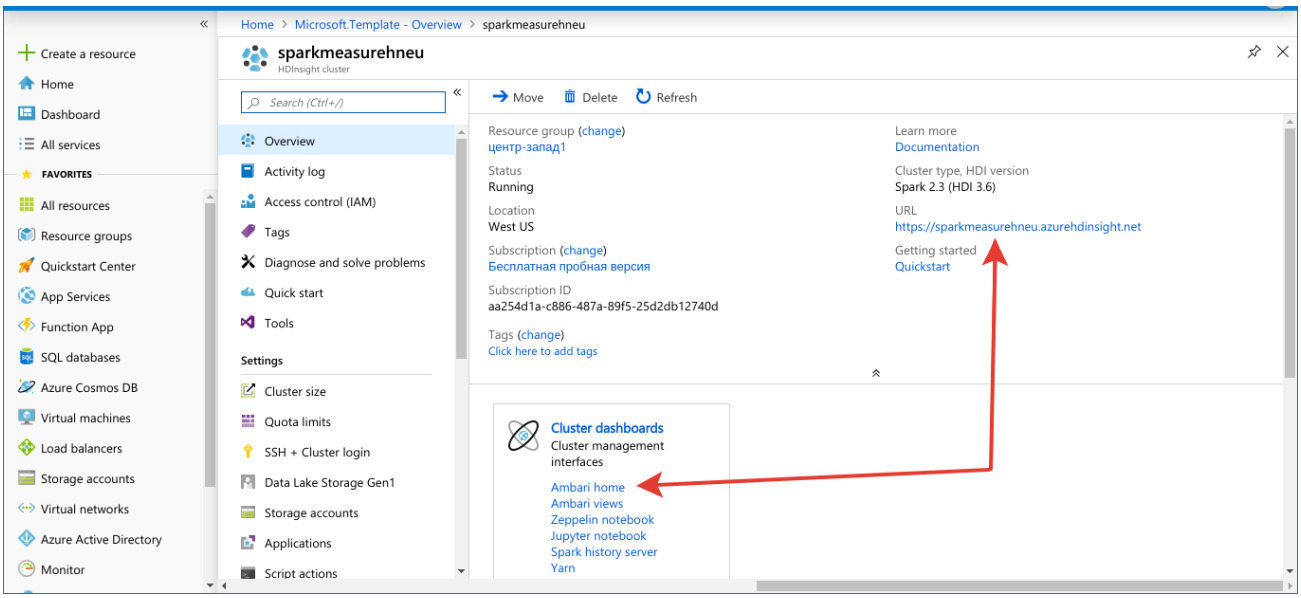

Рисунок 33 – Посилання на сторінку Ambari

Для входу необхідно ввести логін та пароль від master-вузла, що були вказані при його створенні (рис. 34).

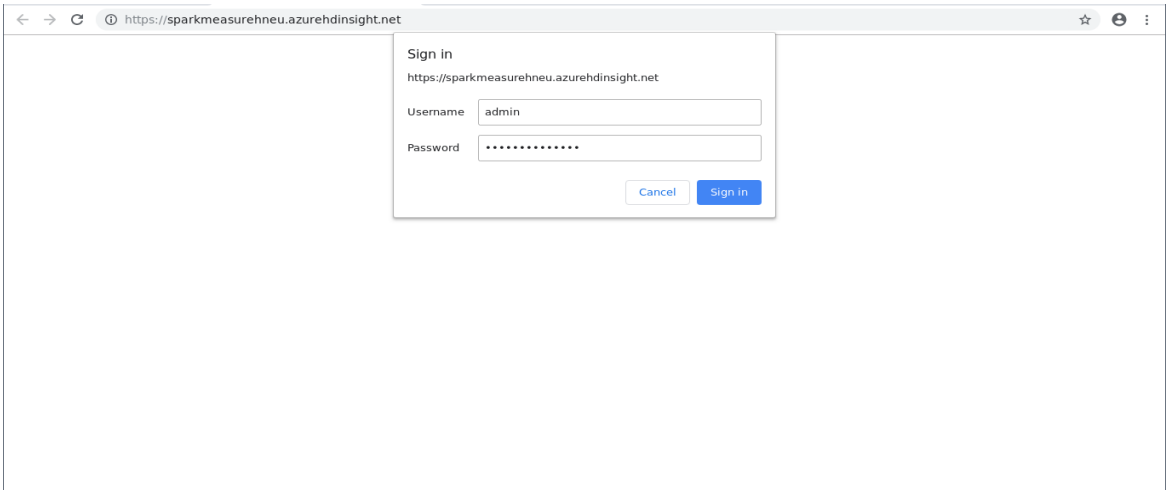

Рисунок 34 – Авторизація у Ambari

Далі необхідно натиснути на пункт YARN в лівому меню (рис. 35).

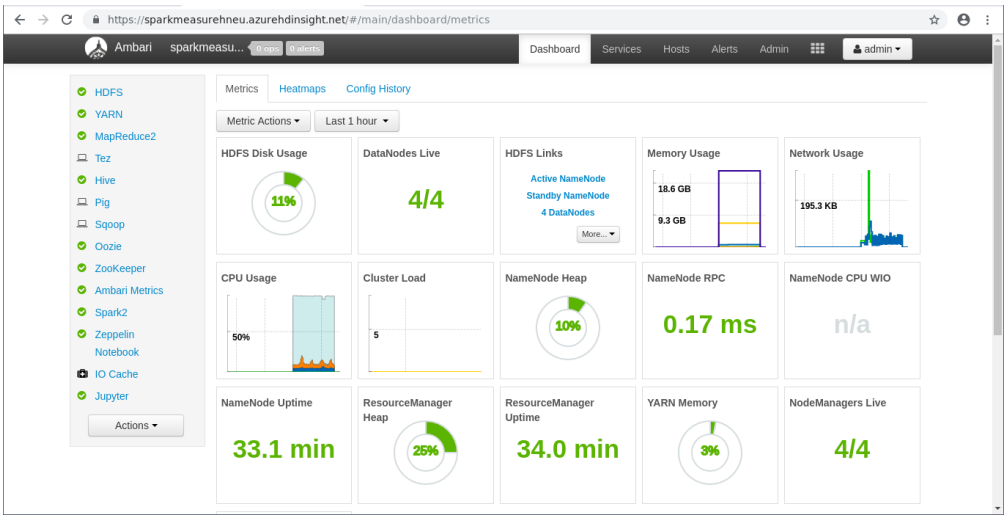

Рисунок 35 – Головна сторінка Ambari

Переходимо на вкладку Config та встановлюємо в поля, що підсвічені жовтим знаком попередження значення з файлу конфігурації: yarn.scheduler.maximum.allocation.mb та yarn.scheduler.maximum.allocation.vcores (рис.36-37).

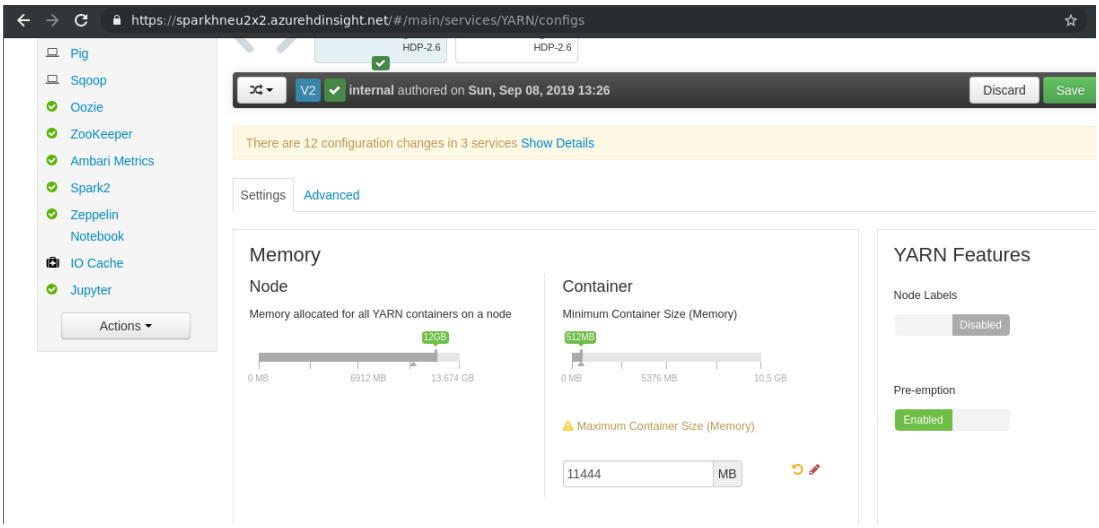

Рисунок 36 – Настройка максимального розміру контейнера

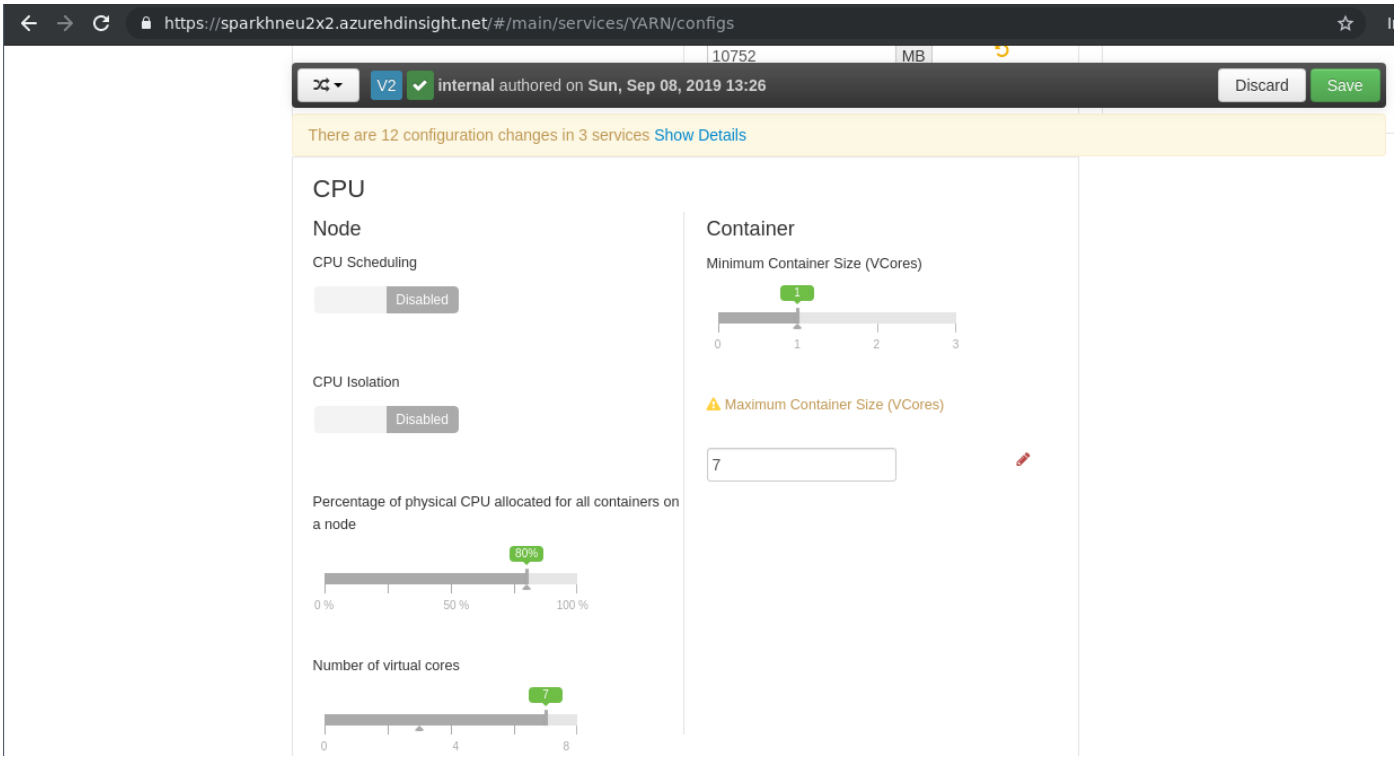

Рисунок 37 – Настройка максимальної кількості віртуальних ядер для контейнера

Зберігаємо зміни, після чого з’явиться модальне вікно з пропозицією обновити пов'язані налаштування (рис. 38), де необхідно підтвердити зміни.

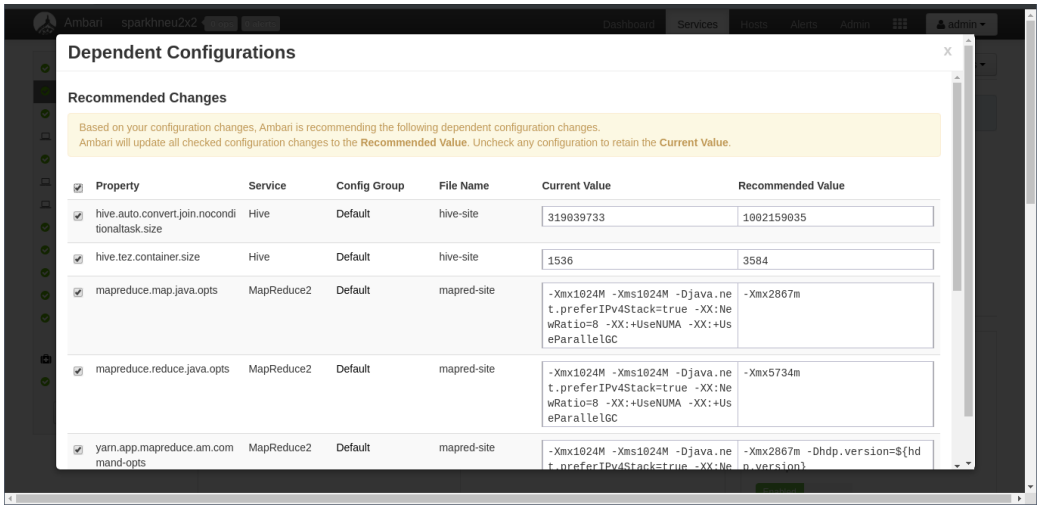

Рисунок 38 – Модальне вікно з пов'язаними налаштування

Далі нам випадає модальне вікно, що зображено на рис. 39, де показано, які конфігурації треба ще змінити до оптимальних. Йдемо за порядком та змінюємо конфігурації до рекомендованих.

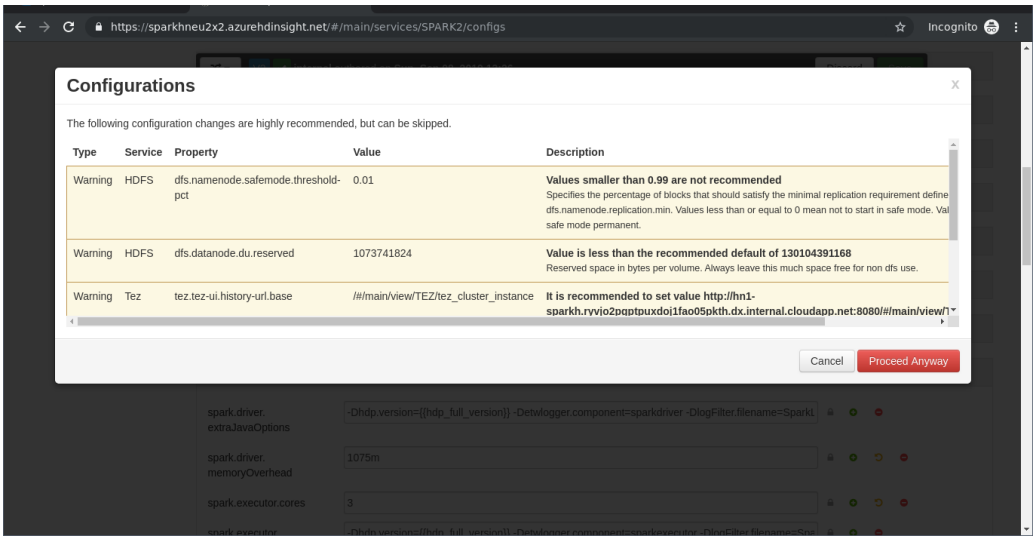

Рисунок 39 – Модальне вікно з рекомендованими налаштування

Далі переходимо на вкладнику Spark, обираємо Advanced, після чого попадаємо на сторінку з розширеними настройками. В блок під назвою «Custom spark2-defaults» (рис. 40) вносимо значення з конфігураційного файлу, а саме:

- spark.driver.memoryOverhead;
- spark.executor.cores;
- spark.executor.instances;
- spark.executor.memory;
- spark.executor.memoryOverhead.

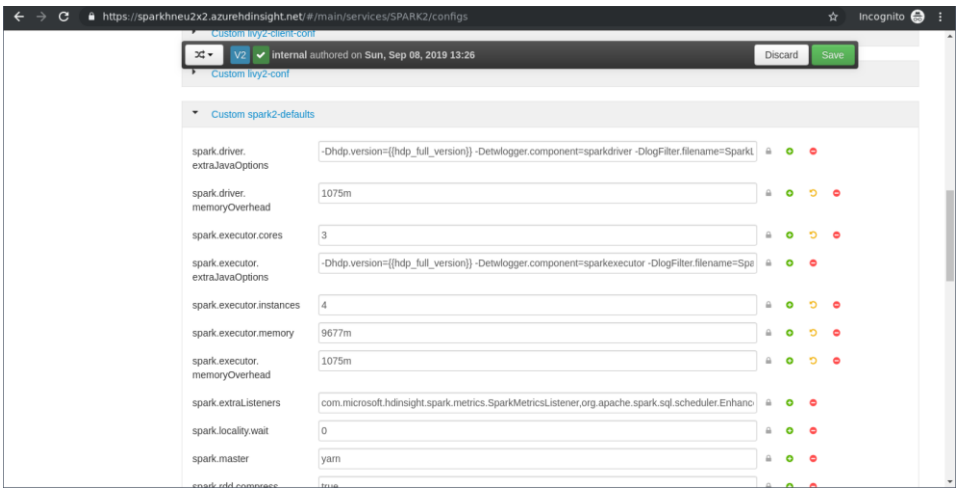

Рисунок 40 – Блок розширених настройок «Custom spark2-defaults»

Після цього зберігаємо настройки та перезапускаємо усі компоненти, в яких є значок оновлення.

3.1.3. Підготовка бенчмарку для роботи на кластері

Щоб підключитися до кластеру, потрібно дізнатися його SSH адресу. Для цього переходимо на розділ “SSH + Cluster login” у боковому меню та вибираємо відповідний хост у випадаючому меню Hostname (рис. 41).

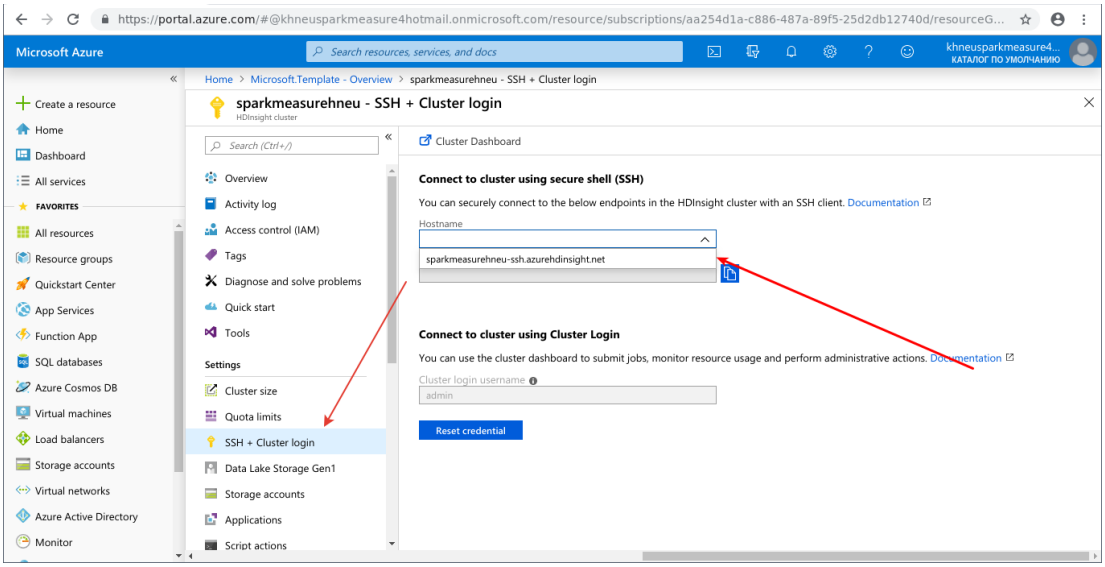

Рисунок 41 – Розділ “SSH + Cluster login” панелі управління кластером

У полі під випадаючим меню з'явиться команда для підключення до кластеру по SSH, яку потрібно скопіювати (рис. 42).

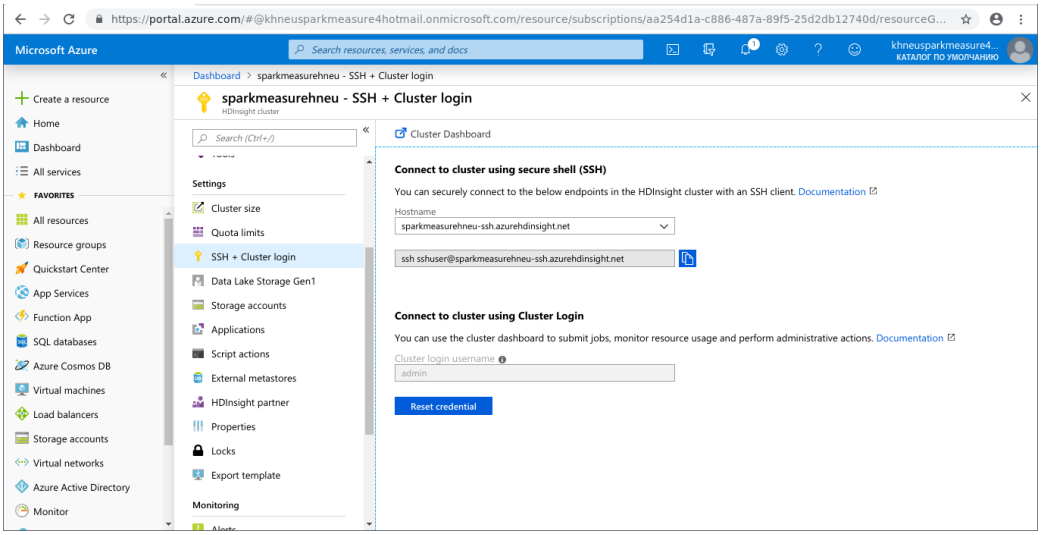

Рисунок 42 – Команда для підключення до кластеру по SSH

3.1.4. Запуск тестування на кластері

Скопійовану команду з попереднього розділу потрібно вставити у вікно терміналу на клієнтській машині. Після виконання команди відбудеться стандартний процес підключення по SSH. В разі успіху у вікні терміналу з'явиться привітальне повідомлення (рис. 43).

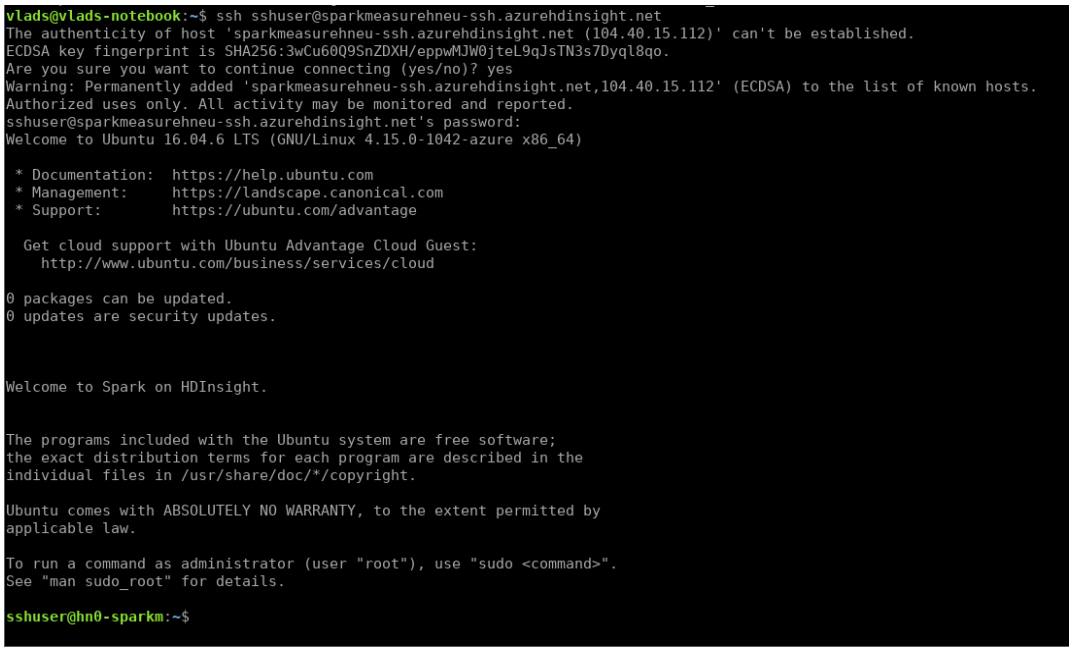

Рисунок 43 – Підключення до кластеру за допомогою SSH

Для того, щоб виконати тестування Spark кластеру, будемо використовувати репозиторій spark-perf, який покриває тестами ядро Spark та бібліотеку Machine Learning. Спочатку потрібно клонувати репозиторій з сирцевим кодом на клієнтську машину (рис. 44), виконавши наступну команду:

git clone https://github.com/databricks/spark-perf.git

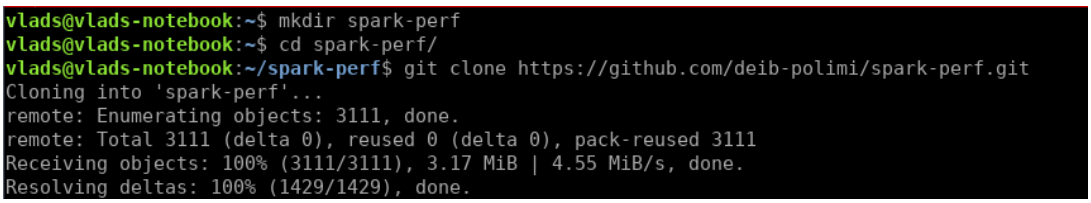

Рисунок 44 – Клонування репозиторію spark-perf з тестами

Далі потрібно створити конфігурацію для виконуваних тестів. Для цього копіюємо файл “config.py.template” з назвою “config.py” (рис. 45) за допомогою наступних команд:

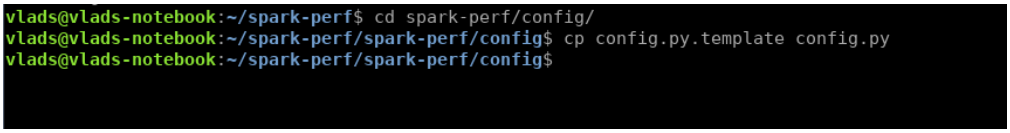

Рисунок 45 – Копіювання файлу з налаштуваннями

У файлі “config.py” можна налаштувати конфігурацію для тестування. Потрібно звернути увагу на наступні опції:

1. SPARK_CLUSTER_URL – URL кластеру Spark, який буде використаний при запуску задач. Має бути задано значення “yarn” (рис. 49), оскільки за замовчуванням кластери Spark на Azure HDInsight використовують саме планувальник Yarn.

2. Секції “RUN” та “PREP”, в яких можна вказати які тести треба запустити і які треба підготовляти. Процес підготовки складається з завантаження third-party пакетів і компіляції коду (тільки для Scala).

Доступні наступні набори тестів:

- SPARK_TESTS – набір тестів Spark Core на Scala;
- PYSPARK_TESTS – набір тестів Spark Core на Python;
- STREAMING_TESTS – набір тестів Spark Streaming на Scala;
- MLLIB_TESTS – набір тестів Spark MlLib на Scala;
- PYTHON_MLLIB_TESTS – набір тестів Spark MlLib на Python.

В нашому випадку вмикаємо лише тести SPARK_TESTS та MLLIB_TESTS (рис. 46).

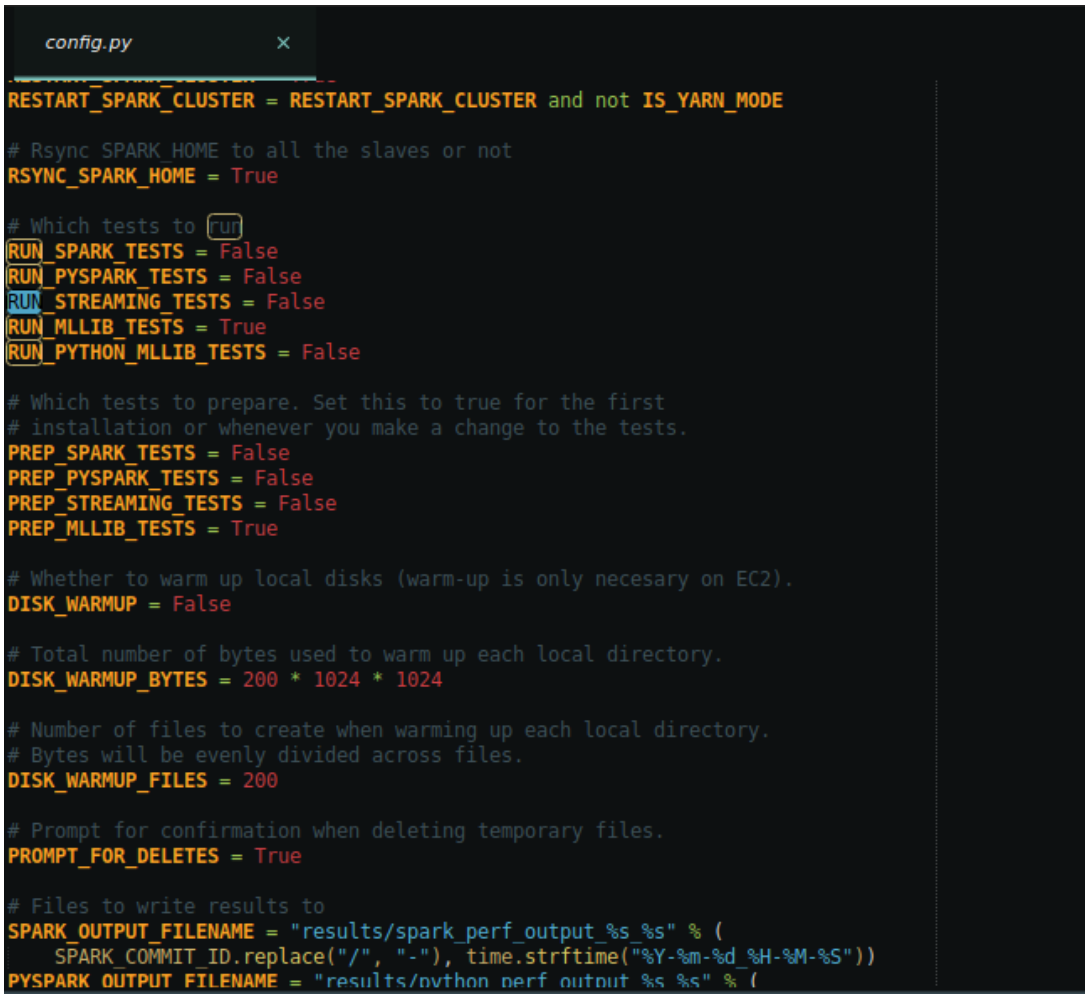

Рисунок 46 – Редагування файлу config.py. “RUN” та “PREP” секції

Далі необхідно додати сервіс моніторингу і метрики SparkMeasure, для чого потрібно модифікувати скрипт запуску тестів.

Для того, щоб налаштувати SparkMeasure вручну необхідно виконати наступні кроки:

1. Додати до файлів spark-perf/mllib-tests/project/MLlibTestsBuild.scala та spark-perf/spark-tests/project/SparkTestsBuild.scala 2 рядка як показано на рис. 47.

"org.apache.spark" %% "spark-sql" % "2.0.0" % "provided",

"ch.cern.sparkmeasure" %% "spark-measure" % "0.13"

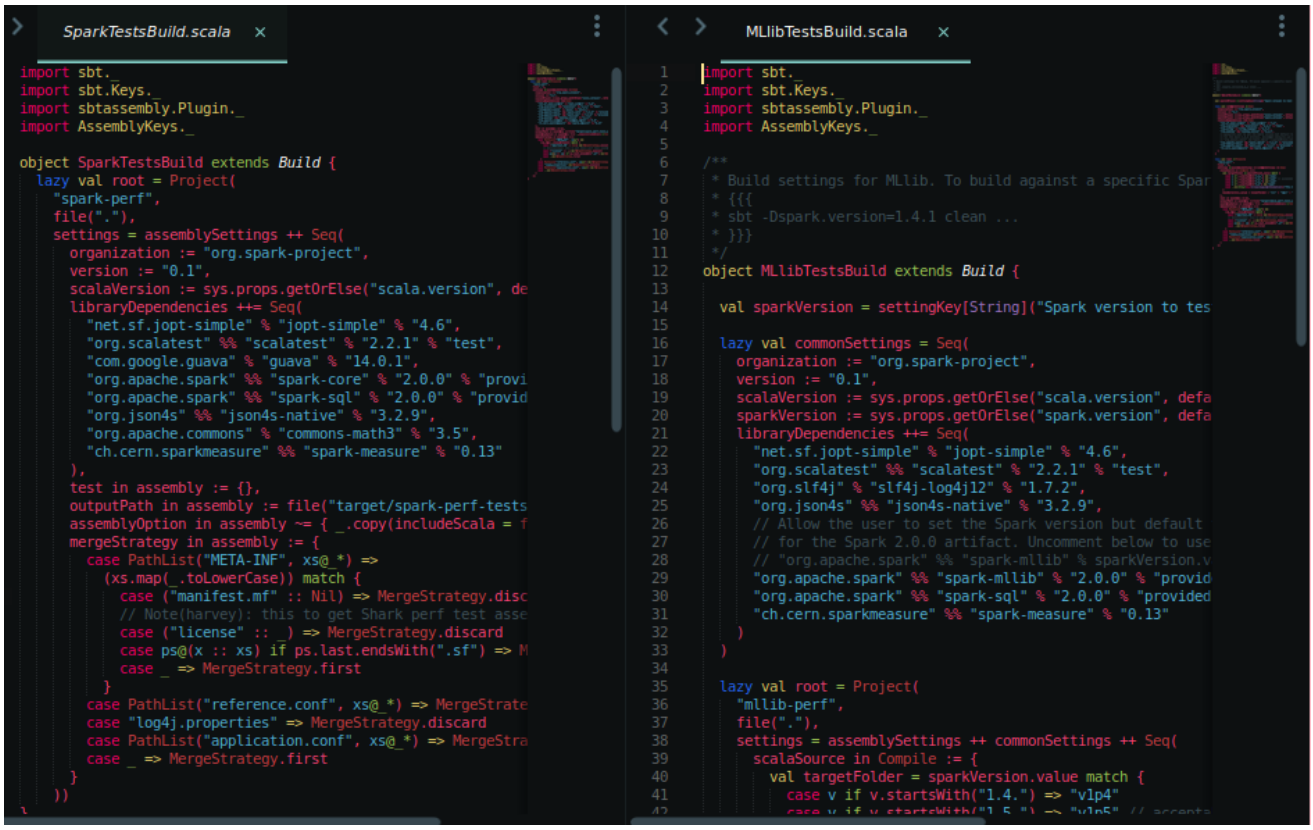

Рисунок 47 – Підключення необхідних бібліотек до файлів MLlibTestsBuild.scala та SparkTestsBuild.scala

2. Модифікувати код наступних файлів:

- mllib-tests/v2p0/src/main/scala/mllib/perf/TestRunner.scala
- spark-tests/src/main/scala/spark/perf/TestRunner.scala

import org.apache.spark.sql.SparkSession

…

val sparkSession = SparkSession

.builder()

.appName("TestRunner: " + testName)

.getOrCreate();

val sc = sparkSession.sparkContext

…

val stageMetrics = ch.cern.sparkmeasure.StageMetrics(sparkSession)

stageMetrics.begin()

...

stageMetrics.end()

stageMetrics.printReport()

Модифікація фалу spark-tests/src/main/scala/spark/perf/TestRunner.scala робиться повністю аналогічно тому, як показано на рис. 48.

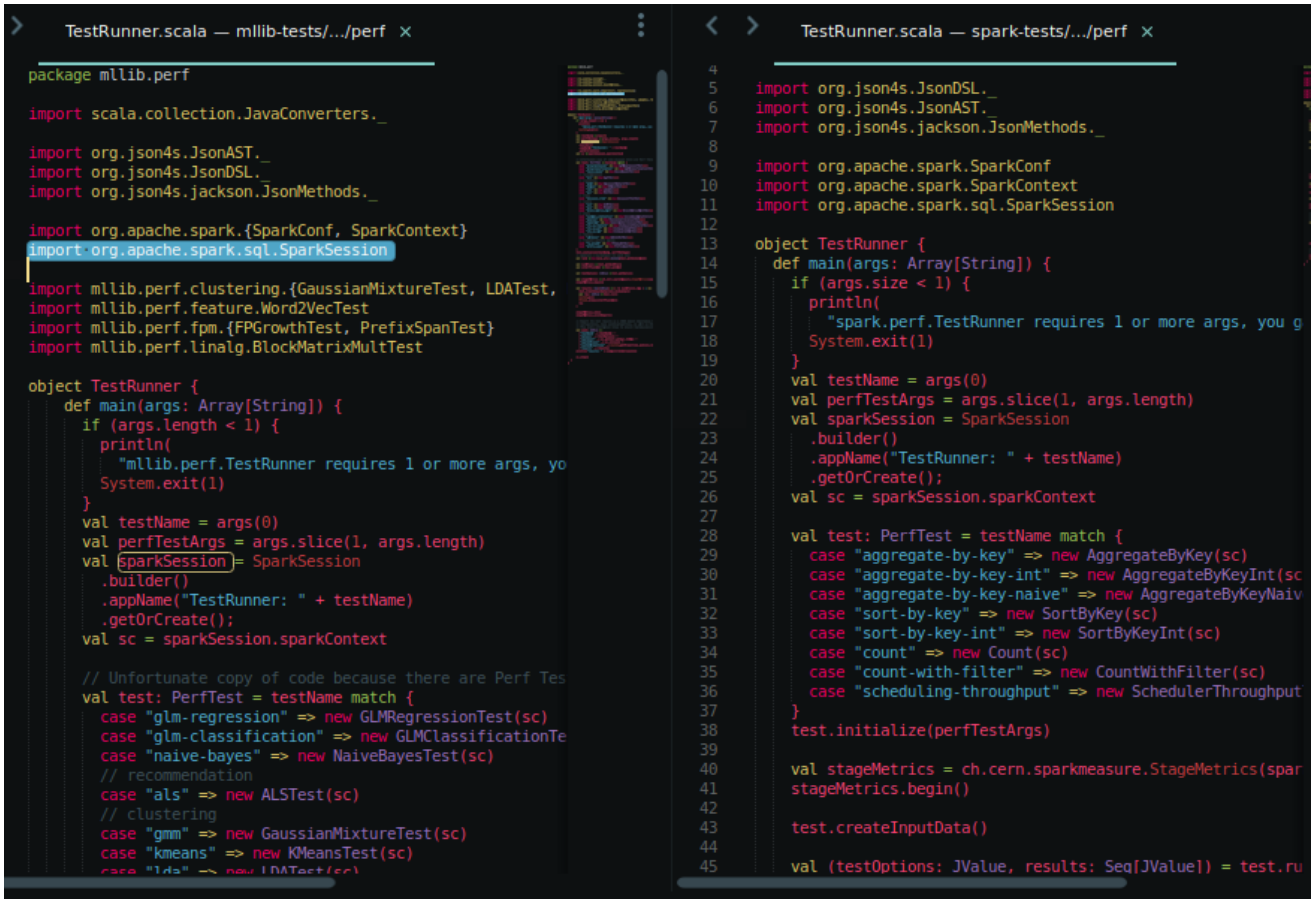

Рисунок 48 – Додавання Spark Measure до скрипту запуску тестів

Після того, як робота над конфігурацією завершена, можна завантажити сирцевий код spark-perf на master-вузол кластеру. Для цього потрібно виконати наступні кроки:

Заархівувати каталог, який був створений при клонуванні репозиторію з GitHub з новою конфігурацією (рис. 49). Щоб зменшити розмір архіву можна видалити папку “.git” перед архівацією.

Завантажити створений архів на master-вузол. Це можна виконати за допомогою команди scp:

scp spark-perf.zip sshuser@sparkmeasurehneu-ssh.azurehdinsight.net

```
vlads@vlads-notebook:~/spark-perf$ zip spark-perf.zip spark-perf
updating: spark-perf/ (stored 0%)
vlads@vlads-notebook:~/spark-perf$
```

Рисунок 49 – Архівація каталогу spark-perf

Перед тим як запускати тестування, необхідно створити SSH ключі, для комунікації master-вузла з worker-вузлами без паролю. Це можна зробити за допомогою команд вказаних нижче (рис. 50). При виконанні команди ssh-keygen пароль потрібно залишити порожнім.

sudo rm -rf ~/.ssh

ssh-keygen

sudo cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

chmod -R 750 ~/.ssh/authorized_keys

Перевірити можливість підключення до localhost без паролю можна за допомогою наступної команди (рис. 51):

ssh localhost

```
sshuser@hn0-sparkm:~$ sudo rm -rf ~/.ssh
sshuser@hn0-sparkm:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sshuser/.ssh/id_rsa):
Created directory '/home/sshuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sshuser/.ssh/id_rsa.
Your public key has been saved in /home/sshuser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:PUbe3qPJynVxa7mH64qn4FJm4+BTmd84fYuq85bGA5I sshuser@hn0-sparkm
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|          .      |
|         + .     |
|       .So= . . .|
|      E X. o . oo|
|     . O.= =o += |
|      +.ooX++=+.o|
|       oo*BB*+++.|
+----[SHA256]-----+
sshuser@hn0-sparkm:~$ sudo cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
sshuser@hn0-sparkm:~$ chmod -R 750 ~/.ssh/authorized_keys
sshuser@hn0-sparkm:~$
```

Рисунок 50 – Створення SSH-ключа

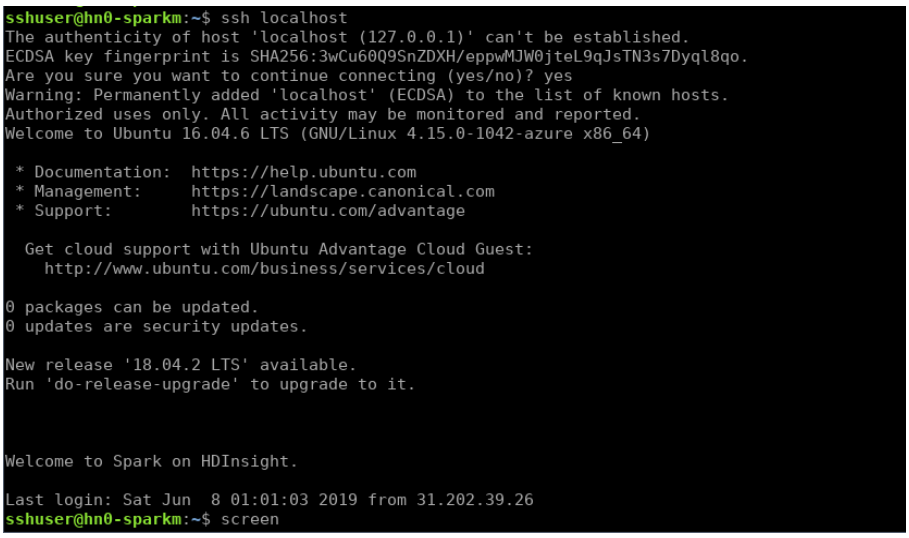

Рисунок 51 – Перевірка можливості підключення до localhost без паролю

Далі разархівуємо архів за допомогою команди *unzip* (рис. 52):

unzip spark-perf.zip

Рисунок 52 – Розархівація архіву з тестами

Наступним кроком необхідно надати права на виконання для build файлів, для цього потрібно виконати команду *chmod* з параметром *+x* для папки з тестами (рис. 53). Права на виконання доречно надавати лише для тих наборів тестів, для яких в файлі “config.py” параметр PREP_<назва_тесту> вказаний True, в нашому випадку потрібно виконати наступні команди:

chmod +x ~/spark-perf/spark-tests/sbt/sbt

chmod +x ~/spark-perf/mllib-tests/sbt/sbt

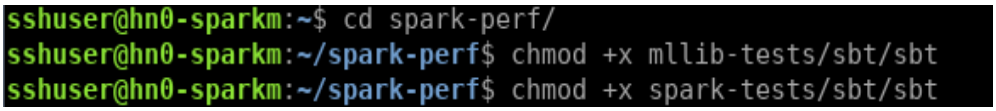

Рисунок 53 – Надання прав на виконання build файлам

Для запуску процесу тестування потрібно надати права на виконання файлу /bin/run та виконати його за допомогою команд, що наведені нижче (рис. 54):

chmod +x ~/spark-perf/bin/run

cd bin/

./run

```
sshuser@hn0-sparkm:~/spark-perf$ chmod +x bin/run
sshuser@hn0-sparkm:~/spark-perf$ cd bin/
sshuser@hn0-sparkm:~/spark-perf/bin$ ./run
```

Рисунок 54 – Запуск тестування

3.2. Вибір оптимальної конфігурації компонентів Spark

Для того, щоб обчислювати продуктивність кластеру, треба знайти оптимальну конфігурацію його параметрів.

Протестуємо конфігурації на однаковому гомогенному кластері, опис установки якого наведений у таблиці 5.

Таблиця 5 – Опис установки

№ п/п	Назва характеристики	Значення
1	Конфігурація майстер-вузлів	2 x Standard_D12_v2
2	Конфігурація воркер-вузлів	2 x Standard_D12_v2
3	Планувальник ресурсів	Capacity Scheduler
4	Ступінь паралелізму	Опціональна
5	Кількість вузлів	4
6	Застосовано ядер	2x4 + 2x4 = 16
7	Застосовано оперативної пам’яті	2x28 + 2x28 = 122
8	Процесор на вузлах	2x4 = 8
9	Оперативна пам’ять на вузлах	2x28 = 56

У таблиці 6 наведено різні конфігурації для заданої установки кластеру.

Таблиця 6 – конфігурації установки

Параметр	Значення для конфігурацій		
	Стандартна	Рекомендована	Оптимізована
spark.driver.memory	1g (1 ГБ)	18,9	18,9

Параметр	Значення для конфігурацій		
	Стандартна	Рекомендована	Оптимізована
spark.driver.memoryOverhead	384 (384 МБ)	2,1	2,1
spark.driver.cores	1	3	3
spark.executor.cores	3	3	3
spark.executor.instances	2	3	4
spark.executor.memory	9216m	18,9	18,9
spark.executor.memoryOverhead	384 (384 МБ)	2,1	2,1
spark.default.parallelism	6	18	32

3.2.1. Стандартна конфігурація

Таблиця 7 – Результати роботи тестового набору «Дерево рішень» (Dtr) для стандартної кофнігурації

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
1	DTR1	0.1148	0.0855	0.069	1.749	1.3805	0.986	0.066
2	DTR2	0.1794	0.18	0.065	2.4783	2.0995	1.005	0.072
3	DTR3	0.1141	0.096	0.043	1.7879	1.3485	1.097	0.064
4	DTR4	0.0936	0.08	0.036	12.9707	12.5285	1.792	0.007
5	DTR5	0.1591	0.146	0.048	5.8457	5.3735	1.152	0.027
6	DTR6	0.4028	0.379	0.067	18.827	18.1975	1.691	0.021
7	DTR7	0.187	0.181	0.063	6.0536	5.614	1.435	0.031
8	DTR8	0.4186	0.3785	0.085	58.7034	58.43	3.764	0.007
9	DTR9	0.1213	0.112	0.04	1.7782	1.393	1.004	0.068
10	DTR10	0.1565	0.1335	0.059	2.3445	2.037	1.041	0.067
11	DTR11	0.118	0.1015	0.048	1.8128	1.436	1.114	0.065
12	DTR12	0.1104	0.0885	0.048	13.5283	12.73	2.347	0.008
13	DTR13	0.1335	0.1285	0.042	3.4776	2.9835	1.285	0.038
14	DTR14	0.2939	0.236	0.15	7.3836	6.9075	1.476	0.04
15	DTR15	0.1412	0.1185	0.066	3.5719	3.1715	1.235	0.04
16	DTR16	0.3266	0.3155	0.053	49.2332	48.1385	3.122	0.007

Таблиця 8 – Результати роботи тестового набору «MLlib» для стандартної кофнігурації

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
als	ML1	0,5056	0,428	0,213	3,5236	3,0785	1,527	0,143
lda-1	ML2	14,9137	14,033	3,839	-	-	-	-
lda-2	ML3	11,0317	10,1165	2,935	-	-	-	-
svd	ML4	152,0374	151,373	10,110	-	-	-	-
pca	ML5	17,7665	17,3095	1,771	-	-	-	-
summary-statistics	ML6	1,2467	1,0055	0,719	-	-	-	-
block-matrix-mult	ML7	1,5094	1,1125	1,038	-	-	-	-
pearson	ML8	10,0619	9,9	0,918	-	-	-	-
spearman	ML9	13,9851	13,4195	1,891	-	-	-	-
chi-sq-feature	ML10	9,9857	9,956	0,866	-	-	-	-
word2vec	ML11	19,3284	18,768	1,766	-	-	-	-
fp-growth	ML12	132,043	131,3375	3,020	-	-	-	-
prefix-span	ML13	1,4047	1,231	0,600	-	-	-	-
glm-regression	ML14	1,1058	1,1695	0,510	14,3570	13,9600	2,805	0,077
glm-classification-1	ML15	0,9748	1,085	0,587	13,2186	13,4850	2,279	0,074
glm-classification-2	ML16	0,8155	0,8445	0,346	24,1888	22,8905	4,411	0,034

3.2.2. Рекомендована конфігурація

Таблиця 9 – Результати роботи тестового набору «Дерево рішень» (Dtr) для рекомендованої конфігурації

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
1	DTR1	0,1068	0,097	0,0409	2,153	1,6835	1,3528	0,0496
2	DTR2	0,1308	0,1075	0,0649	2,7707	2,2305	1,5179	0,0472
3	DTR3	0,0996	0,0905	0,0373	2,0768	1,6515	1,192	0,048
4	DTR4	0,1191	0,103	0,0366	15,4205	14,5605	2,7114	0,0077
5	DTR5	0,1838	0,146	0,0855	6,5867	6,124	1,7387	0,0279
6	DTR6	0,4072	0,392	0,0579	24,4883	24,1445	2,2165	0,0166
7	DTR7	0,173	0,1475	0,079	6,5031	6,0115	1,7389	0,0266
8	DTR8	0,4106	0,3975	0,0411	62,2016	61,6795	2,4467	0,0066
9	DTR9	0,1078	0,091	0,0486	2,0751	1,604	1,2172	0,0519
10	DTR10	0,1384	0,114	0,0767	2,4393	1,9415	1,3654	0,0567
11	DTR11	0,1057	0,0955	0,0408	2,21	1,6425	1,5898	0,0478
12	DTR12	0,1158	0,1065	0,0246	15,8496	14,97	2,6195	0,0073
13	DTR13	0,1358	0,118	0,0471	4,0638	3,4325	1,7404	0,0334
14	DTR14	0,2711	0,233	0,09	7,1248	6,461	1,8322	0,0381
15	DTR15	0,1348	0,1185	0,0472	4,2705	3,662	1,6376	0,0316
16	DTR16	0,3567	0,3245	0,0806	53,8817	53,083	4,2503	0,0066

Таблиця 10 – Результати роботи тестового набору «MLlib» для рекомендованої конфігурації

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
als	ML1	0,4274	0,386	0,1919	2,4123	2,0675	1,1059	0,1772
lda-1	ML2	11,9619	10,6755	3,7995	-	-	-	
lda-2	ML3	9,4459	8,368	2,3635	-	-	-	
svd	ML4	81,3531	80,483	6,0233	-	-	-	
pca	ML5	12,0711	11,6095	1,4527	-	-	-	
summary-statistics	ML6	0,7248	0,5555	0,6514	-	-	-	
block-matrix-mult	ML7	1,3122	1,335	1,0082	-	-	-	
pearson	ML8	5,477	5,214	0,7379	-	-	-	
spearman	ML9	8,5268	7,698	2,0968	-	-	-	
chi-sq-feature	ML10	5,7515	5,649	0,5235	-	-	-	
word2vec	ML11	10,2544	9,9155	1,3219	-	-	-	
fp-growth	ML12	79,5234	78,314	3,0691	-	-	-	
prefix-span	ML13	1,0462	0,883	0,534	-	-	-	
glm-regression	ML14	0,1189	0,0885	0,0979	3,8912	2,8465	3,1655	0,0306
glm-classification-1	ML15	0,1001	0,097	0,0232	3,3272	2,966	1,2443	0,0301
glm-classification-2	ML16	0,0946	0,0825	0,0315	7,6107	6,5285	3,679	0,0124

3.2.3. Оптимізована конфігурація

Таблиця 11 – Результати роботи тестового набору «Дерево рішень» (Dtr) для оптимізованої конфігурації

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
1	DTR1	0,0971	0,0855	0,0362	2,1292	1,6805	1,2861	0,0456
2	DTR2	0,1239	0,099	0,0785	2,7242	2,236	1,4112	0,0455
3	DTR3	0,1125	0,0925	0,0603	2,2199	1,865	1,1948	0,0507

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
4	DTR4	0,1165	0,1	0,0588	15,8561	15,046	2,5527	0,0073
5	DTR5	0,1568	0,1475	0,0416	6,2	5,4965	1,7309	0,0253
6	DTR6	0,4309	0,4	0,0952	22,3699	21,914	1,8451	0,0193
7	DTR7	0,1464	0,1405	0,0306	6,2603	5,6925	1,5614	0,0234
8	DTR8	0,4459	0,4065	0,1125	60,5524	59,599	3,2356	0,0074
9	DTR9	0,1065	0,093	0,054	2,1411	1,683	1,2019	0,0497
10	DTR10	0,1531	0,127	0,0727	2,6058	2,116	1,2782	0,0588
11	DTR11	0,1037	0,081	0,0623	2,1943	1,777	1,2666	0,0473
12	DTR12	0,1125	0,088	0,0537	16,4109	15,585	2,457	0,0069
13	DTR13	0,1408	0,113	0,0627	4,4476	3,723	1,5414	0,0317
14	DTR14	0,2543	0,2345	0,0626	6,6381	6,0135	1,7165	0,0383
15	DTR15	0,1507	0,1355	0,0806	4,5846	4,0185	1,8031	0,0329
16	DTR16	0,3694	0,3345	0,0847	52,7224	52,0535	3,0916	0,007

Таблиця 12 – Результати роботи тестового набору «MLlib» для оптимізованої конфігурації

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
als	ML1	0,4386	0,4135	0,1173	2,5215	2,3425	0,9856	0,1739
lda-1	ML2	13,18	12,307	2,8959	-	-	-	
lda-2	ML3	9,2357	8,482	2,3535	-	-	-	
svd	ML4	65,6771	65,081	5,9376	-	-	-	

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
pca	ML5	10,9313	10,4935	1,3336	-	-	-	
summary-statistics	ML6	0,6955	0,5295	0,5878	-	-	-	
block-matrix-mult	ML7	1,6982	1,5715	0,7247	-	-	-	
pearson	ML8	5,1556	4,954	0,7141	-	-	-	
spearman	ML9	7,9012	7,192	2,0138	-	-	-	
chi-sq-feature	ML10	5,0123	5,0445	0,5812	-	-	-	
word2vec	ML11	9,6737	9,2245	1,2815	-	-	-	
fp-growth	ML12	67,2651	66,7105	1,6931	-	-	-	
prefix-span	ML13	1,1547	0,908	0,6306	-	-	-	
glm-regression	ML14	0,1056	0,0945	0,046	3,3832	2,584	2,0805	0,0312
glm-classification-1	ML15	0,0954	0,0875	0,0228	3,198	2,6625	1,5105	0,0298
glm-classification-2	ML16	0,0997	0,087	0,0417	8,3858	7,4125	3,0929	0,0119

3.2.4. Порівняння результатів для конфігурацій

Таблиця 13 – Порівняння результатів для середнього часу навчання

№	Ідентифікатор тесту	Серед. час навчання, с			Краща конфігурація
		Стандартна	Рекомендована	Оптимізована	
1	DTR1	2,9184	2,1530	2,1292	Оптимізована
2	DTR2	4,1255	2,7707	2,7242	Оптимізована
3	DTR3	3,1569	2,0768	2,2199	Рекомендована
4	DTR4	23,4146	15,4205	15,8561	Рекомендована
5	DTR5	10,9460	6,5867	6,2000	Оптимізована

№	Ідентифікатор тесту	Серед. час навчання, с			Краща конфігурація
		Стандартна	Рекомендована	Оптимізована	
6	DTR6	37,1587	24,4883	22,3699	Оптимізована
7	DTR7	11,6783	6,5031	6,2603	Оптимізована
8	DTR8	113,1543	62,2016	60,5524	Оптимізована
9	DTR9	2,9734	2,0751	2,1411	Рекомендована
10	DTR10	3,5518	2,4393	2,6058	Рекомендована
11	DTR11	3,0505	2,2100	2,1943	Оптимізована
12	DTR12	24,6764	15,8496	16,4109	Рекомендована
13	DTR13	6,2475	4,0638	4,4476	Рекомендована
14	DTR14	14,0640	7,1248	6,6381	Оптимізована
15	DTR15	6,6548	4,2705	4,5846	Рекомендована
16	DTR16	97,8011	53,8817	52,7224	Оптимізована
17	als	3,5236	2,4123	2,5215	Рекомендована
18	glm-regression	14,3570	3,8912	3,3832	Оптимізована
19	glm-classification-1	13,2186	3,3272	3,1980	Оптимізована
20	glm-classification-2	24,1888	7,6107	8,3858	Рекомендована

Рисунок 55 – Графік порівняння середнього часу навчання на різних конфігураціях кластерів

Таблиця 14 – Порівняння результатів для середнього часу тестування

№	Ідентифікатор тесту	Серед. час тестування, с			Краща конфігурація
		Стандартна	Рекомендована	Оптимізована	
1	DTR1	0,2088	0,1068	0,0971	Оптимізована
2	DTR2	0,2468	0,1308	0,1239	Оптимізована
3	DTR3	0,241	0,0996	0,1125	Рекомендована
4	DTR4	0,2327	0,1191	0,1165	Оптимізована
5	DTR5	0,3596	0,1838	0,1568	Оптимізована
6	DTR6	0,6871	0,4072	0,4309	Рекомендована
7	DTR7	0,3061	0,1730	0,1464	Оптимізована
8	DTR8	0,6727	0,4106	0,4459	Рекомендована
9	DTR9	0,2663	0,1078	0,1065	Оптимізована
10	DTR10	0,3086	0,1384	0,1531	Рекомендована

№	Ідентифікатор тесту	Серед. час тестування, с			Краща конфігурація
		Стандартна	Рекомендована	Оптимізована	
11	DTR11	0,2499	0,1057	0,1037	Оптимізована
12	DTR12	0,2312	0,1158	0,1125	Оптимізована
13	DTR13	0,2494	0,1358	0,1408	Рекомендована
14	DTR14	0,4594	0,2711	0,2543	Оптимізована
15	DTR15	0,2905	0,1348	0,1507	Рекомендована
16	DTR16	0,6955	0,3567	0,3694	Рекомендована
17	als	0,5056	0,4274	0,4386	Рекомендована
18	glm-regression	1,1058	0,1189	0,1056	Оптимізована
19	glm-classification-1	0,9748	0,1001	0,0954	Оптимізована
20	glm-classification-2	0,8155	0,0946	0,0997	Рекомендована

Рисунок 56 – Графік порівняння середнього часу тестування на різних конфігураціях кластерів

Виходячи з результатів для навчання та тестування можна зробити висновок, що в більшості тестів оптимальна конфігурація перемагає рекомендовану конфігурацію, тому ми будемо використовувати її для подальших обчислень

3.3. Аналіз отриманих результатів для різних конфігурацій віртуальних машин

В результаті проведених обчислювальних експериментів були отримані результати для наступних конфігурацій кластера Apache Spark (таблиця 15). Відповідні конфігурації параметрів кластерів представлені у таблиця 16.

Таблиця 15 – Конфігурації кластерів, що приймали участь у тестуванні

Тип кластеру	Master-вузли		Worker-вузли	
	Кількість, шт	Конфігурація розміру	Кількість, шт	Конфігурація розміру
Гомогенний	2	Standard_D12_v2	2	Standard_D12_v2
	2	Standard_D12_v2	3	Standard_D12_v2
Гетерогенний	2	Standard_D12_v2	2	Standard_D13_v2
	2	Standard_D12_v2	3	Standard_D13_v2

Таблиця 16 – Параметри для Spark, HDFS та YARN

Конфігурація ВМ	2xD12v2 + 2xD12v2	2xD12v2 + 3xD12v2	2xD12v2 + 2xD13v2	2xD12v2 + 3xD13v2
spark.executor.instances	4	5	4	5
spark.yarn.executor.memoryOverhead	2,1g	2,1g	4,2g	4,2g
spark.executor.memory	18,9g	18,9g	37,8g	37,8g
spark.yarn.driver.memoryOverhead	2,1g	2,1g	4,2g	4,2g
spark.driver.memory	9,45g	9,45g	9,45g	9,45g
spark.executor.cores	3	3	6	6
spark.driver.cores	3	3	3	3
spark.default.parallelism	32	40	48	64

3.3.1. Гомогенні кластери

3.3.1.1 2 master-вузли «Standard_D12_v2» та 2 worker-вузли «Standard_D12_v2»

Результати для цієї конфігурації представлені в таблицях 11-12.

3.3.1.2 2 master-вузли «Standard_D12_v2» та 3 worker-вузли «Standard_D12_v2»

Таблиця 17 – Опис установки

№ п/п	Назва характеристики	Значення
1	Конфігурація майстер-вузлів	2 x Standard_D12_v2
2	Конфігурація воркер-вузлів	3 x Standard_D12_v2
3	Планувальник ресурсів	Capacity Scheduler
4	Ступінь паралелізму	40
5	Кількість вузлів	5
6	Застосовано ядер	2x4 + 3x4 = 20
7	Застосовано оперативної пам’яті	2x28 + 3x28 = 140
8	Процесор на вузлах	3x4 = 12
9	Оперативна пам’ять на вузлах	3x28 = 84

Таблиця 18 – Результати роботи тестового набору «Дерево рішень» (Dtr)

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
1	DTR1	0,08	0,08	0,043	2,0680	1,5885	1,0779	0,048
2	DTR2	0,1106	0,0945	0,0471	2,5752	2,092	1,385	0,0429
3	DTR3	0,096	0,0795	0,0374	1,9800	1,6105	1,0612	0,0485
4	DTR4	0,0912	0,0825	0,0325	14,9217	13,629	4,1106	0,0061
5	DTR5	0,145	0,1335	0,0505	5,5558	4,766	1,8786	0,0261
6	DTR6	0,4084	0,39	0,0574	19,0045	18,114	2,1001	0,0215
7	DTR7	0,1205	0,1785	0,0438	5,6504	5,1135	1,5272	0,0311
8	DTR8	0,3909	0,382	0,0538	53,1322	52,407	3,6426	0,0074
9	DTR9	0,085	0,092	0,0505	1,9680	1,6905	1,1522	0,0548
10	DTR10	0,1246	0,115	0,0534	2,3276	1,912	1,254	0,0535
11	DTR11	0,09	0,09	0,0464	2,1006	1,695	1,1414	0,0509
12	DTR12	0,1025	0,0915	0,0309	15,4948	13,9155	3,6253	0,0066
13	DTR13	0,1356	0,116	0,044	3,8171	3,2755	1,4972	0,0355

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
14	DTR14	0,2388	0,2205	0,0601	5,8492	5,352	1,5988	0,0408
15	DTR15	0,1335	0,1235	0,0429	3,9426	3,4285	1,5484	0,0339
16	DTR16	0,33	0,2955	0,0749	47,1632	45,7705	3,878	0,007

Таблиця 19 – Результати роботи тестового набору «MLlib»

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
als	ML1	0,4594	0,427	0,1566	2,5845	1,987	1,7238	0,1778
lda-1	ML2	12,5016	11,521	2,8404	-	-	-	
lda-2	ML3	9,2749	8,34	2,3975	-	-	-	
svd	ML4	68,0799	67,95	5,468	-	-	-	
pca	ML5	12,0248	11,5915	1,2043	-	-	-	
summary-statistics	ML6	0,7353	0,534	0,6182	-	-	-	
block-matrix-mult	ML7	0,6029	0,4525	0,3668	-	-	-	
pearson	ML8	5,3467	5,1565	0,6886	-	-	-	
spearman	ML9	7,2964	6,675	2,0575	-	-	-	
chi-sq-feature	ML10	5,4397	5,374	0,2851	-	-	-	
word2vec	ML11	10,4339	10,1	1,0866	-	-	-	
fp-growth	ML12	64,3223	63,915	1,3784	-	-	-	
prefix-span	ML13	1,1702	0,9885	0,617	-	-	-	
glm-regression	ML14	0,1014	0,079	0,049	3,3699	2,568	1,8947	0,0301
glm-classification-1	ML15	0,084	0,0725	0,0271	3,1078	2,7295	1,3607	0,027
glm-classification-2	ML16	0,0929	0,085	0,0317	8,7502	7,584	2,8527	0,0106

Таблиця 20 – Порівняння часу навчання на гомогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

Ідентифікатор тесту	Серед. час навчання, с	Відношення

	2 обчисл. вузли	3 обчисл. вузли	
DTR1	2,1292	2,0680	0,9713
DTR2	2,7242	2,5752	0,9453
DTR3	2,2199	1,9800	0,8919
DTR4	15,8561	14,9217	0,9411
DTR5	6,2000	5,5558	0,8961
DTR6	22,3699	19,0045	0,8496
DTR7	6,2603	5,6504	0,9026
DTR8	60,5524	53,1322	0,8775
DTR9	2,1411	1,9680	0,9192
DTR10	2,6058	2,3276	0,8932
DTR11	2,1943	2,1006	0,9573
DTR12	16,4109	15,4948	0,9442
DTR13	4,4476	3,8171	0,8582
DTR14	6,6381	5,8492	0,8812
DTR15	4,5846	3,9426	0,8600
DTR16	52,7224	47,1632	0,8946

Середня різниця приросту продуктивності при навчанні сягає близько 14%.

Рисунок 57 – Графік для порівняння часу навчання на гомогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

Таблиця 21 – Порівняння часу тестування на гомогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

Ідентифікатор тесту	Серед. час тестування, с		Відношення
	2 обчисл. вузли	3 обчисл. вузли	

DTR1	0,0971	0,08	0,8239
DTR2	0,1239	0,1106	0,8927
DTR3	0,1125	0,096	0,8533
DTR4	0,1165	0,0912	0,7828
DTR5	0,1568	0,145	0,9247
DTR6	0,4309	0,4084	0,9478
DTR7	0,1464	0,1205	0,8231
DTR8	0,4459	0,3909	0,8767
DTR9	0,1065	0,085	0,7981
DTR10	0,1531	0,1246	0,8138
DTR11	0,1037	0,09	0,8679
DTR12	0,1125	0,1025	0,9111
DTR13	0,1408	0,1356	0,9631
DTR14	0,2543	0,2388	0,9390
DTR15	0,1507	0,1335	0,8859
DTR16	0,3694	0,33	0,8933

Середня різниця приросту продуктивності при навчанні сягає близько 15%.

Рисунок 58 – Графік для порівняння часу тестування на гомогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

3.3.2. Гетерогенні кластери

3.3.2.1 2 master-вузли «Standard_D12_v2» та 2 worker-вузли «Standard_D13_v2»

Таблиця 22 – Опис установки

№ п/п	Назва характеристики	Значення
1	Конфігурація майстер-вузлів	2 x Standard_D12_v2
2	Конфігурація воркер-вузлів	2 x Standard_D13_v2
3	Планувальник ресурсів	Capacity Scheduler
4	Ступінь паралелізму	48
5	Кількість вузлів	4
6	Застосовано ядер	2x4 + 2x8 = 24
7	Застосовано оперативної пам’яті	2x28 + 2x56 = 168
8	Процесор на вузлах	2x8 = 16
9	Оперативна пам’ять на вузлах	2x56 = 112

Таблиця 23 – Результати роботи тестового набору «Дерево рішень» (Dtr)

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
1	DTR1	0,1277	0,121	0,0518	2,6746	2,2115	1,5734	0,0477
2	DTR2	0,16	0,1345	0,0842	3,298	2,7185	1,5778	0,0485
3	DTR3	0,114	0,097	0,0491	2,5772	2,0595	1,5048	0,0442
4	DTR4	0,1067	0,095	0,0461	19,7472	18,5975	3,0501	0,0054
5	DTR5	0,1554	0,1445	0,0297	6,6432	6,205	1,7485	0,0234
6	DTR6	0,4163	0,3915	0,0655	22,3865	21,778	2,3448	0,0186
7	DTR7	0,1512	0,142	0,029	6,6944	6,1005	1,8998	0,0226
8	DTR8	0,4643	0,404	0,1594	66,9183	66,013	2,4748	0,0069
9	DTR9	0,107	0,089	0,043	2,4696	2,0425	1,2756	0,0433
10	DTR10	0,1344	0,123	0,0518	2,8034	2,2225	1,4427	0,0479
11	DTR11	0,1171	0,0895	0,0867	2,7494	2,1245	1,7632	0,0426
12	DTR12	0,1154	0,098	0,0449	19,4095	18,7905	2,5	0,0059
13	DTR13	0,1652	0,1475	0,0662	5,1167	4,473	1,7271	0,0323
14	DTR14	0,2367	0,2215	0,0532	6,9887	6,373	1,876	0,0339
15	DTR15	0,1489	0,137	0,0397	5,3098	4,9465	1,6153	0,028
16	DTR16	0,3364	0,311	0,0916	60,2058	59,1115	2,8524	0,0056

Таблиця 24 – Результати роботи тестового набору «MLlib»

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
als	ML1	0,5398	0,494	0,1289	2,438	2,0985	0,9224	0,2214
lda-1	ML2	18,2088	17,1565	3,784	-	-	-	
lda-2	ML3	9,7672	8,983	2,3063	-	-	-	
svd	ML4	63,9423	63,326	5,3186	-	-	-	
pca	ML5	11,5531	10,964	1,3863	-	-	-	
summary-statistics	ML6	0,7109	0,5225	0,6192	-	-	-	
block-matrix-mult	ML7	1,5314	1,487	0,288	-	-	-	
pearson	ML8	5,1737	4,897	0,7339	-	-	-	
spearman	ML9	8,119	7,2965	2,6199	-	-	-	
chi-sq-feature	ML10	5,4999	5,3515	0,5008	-	-	-	
word2vec	ML11	9,7645	9,3	1,5292	-	-	-	
fp-growth	ML12	69,833	69,039	1,7408	-	-	-	
prefix-span	ML13	1,3926	1,1515	0,7054	-	-	-	
glm-regression	ML14	0,0942	0,083	0,0237	3,4031	2,7015	2,1102	0,0277
glm-classification-1	ML15	0,0963	0,0835	0,0315	3,4103	2,848	1,454	0,0282
glm-classification-2	ML16	0,0964	0,085	0,0465	8,2667	7,328	3,0263	0,0117

3.3.2.2 2 master-вузли «Standard_D12_v2» та 3 worker-вузли «Standard_D13_v2»

Таблиця 25 – Опис установки

№ п/п	Назва характеристики	Значення
1	Конфігурація майстер-вузлів	2 x Standard_D12_v2
2	Конфігурація воркер-вузлів	3 x Standard_D13_v2
3	Планувальник ресурсів	Capacity Scheduler
4	Ступінь паралелізму	64
5	Кількість вузлів	5
6	Застосовано ядер	2x4 + 3x8 = 32
7	Застосовано оперативної пам’яті	2x28 + 3x56 = 224
8	Процесор на вузлах	3x8 = 24
9	Оперативна пам’ять на вузлах	3x56 = 168

Таблиця 26 – Результати роботи тестового набору «Дерево рішень» (Dtr)

№	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
1	DTR1	0,1061	0,099	0,0343	2,1784	1,818	1,1105	0,0487
2	DTR2	0,1317	0,106	0,0655	2,7923	2,4045	1,3532	0,0472
3	DTR3	0,1004	0,0875	0,0389	2,2272	1,862	1,1368	0,0451
4	DTR4	0,0914	0,098	0,0365	17,3074	16,5415	2,6651	0,0064
5	DTR5	0,1258	0,1655	0,1167	5,7753	4,813	2,073	0,0367
6	DTR6	0,4085	0,39	0,0583	16,4465	15,857	1,7365	0,0248
7	DTR7	0,1324	0,1445	0,0332	5,6087	4,9975	1,6385	0,0273
8	DTR8	0,4374	0,3875	0,1165	55,6258	53,3885	4,6424	0,0079
9	DTR9	0,0865	0,095	0,0521	2,1714	1,7465	1,1393	0,0488
10	DTR10	0,1318	0,1135	0,0446	2,5651	2,096	1,3227	0,0514
11	DTR11	0,0999	0,089	0,0371	2,2829	1,836	1,2607	0,0438
12	DTR12	0,1129	0,1005	0,0358	17,5106	16,506	3,0733	0,0064
13	DTR13	0,1402	0,138	0,0892	4,4882	3,7465	1,7729	0,0379
14	DTR14	0,2316	0,212	0,0509	5,9187	5,442	1,6501	0,0391
15	DTR15	0,1256	0,157	0,0807	4,7227	4,1025	1,8071	0,0371
16	DTR16	0,2965	0,308	0,1	49,345	48,165	4,5763	0,0072

Таблиця 27 – Результати роботи тестового набору «MLlib»

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
als	ML1	0,5624	0,5815	0,1194	2,7009	2,5615	0,8796	0,2082
lda-1	ML2	15,5694	14,415	3,2071	-	-	-	
lda-2	ML3	9,036	8,4355	2,1171	-	-	-	
svd	ML4	63,1744	62,956	5,7117	-	-	-	
pca	ML5	11,4606	10,9875	1,4769	-	-	-	
summary-statistics	ML6	0,7669	0,4935	0,845	-	-	-	
block-matrix-mult	ML7	0,5783	0,4575	0,3724	-	-	-	

Назва тесту	Id	Тестування			Навчання			Серед. зн. тест. / серед. зн. навч.
		Серед. зн., с	Медіана, с	СКВ	Серед. зн., с	Медіана, с	СКВ	
pearson	ML8	5,0412	4,806	0,6905	-	-	-	
spearman	ML9	7,0741	6,5075	1,8997	-	-	-	
chi-sq-feature	ML10	5,1979	5,0635	0,4478	-	-	-	
word2vec	ML11	9,9814	9,4825	1,2726	-	-	-	
fp-growth	ML12	61,9484	61,5795	1,3969	-	-	-	
prefix-span	ML13	1,2658	1,0535	0,6271	-	-	-	
glm-regression	ML14	0,0948	0,082	0,034	3,3957	2,661	1,997	0,0279
glm-classification-1	ML15	0,1013	0,092	0,0337	3,4001	2,939	1,3187	0,0298
glm-classification-2	ML16	0,1006	0,0925	0,0389	8,1728	7,325	2,7251	0,0123

Таблиця 28 – Порівняння часу навчання на гетерогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

Ідентифікатор тесту	Серед. час навчання, с		Відношення
	2 обчисл. вузли	3 обчисл. вузли	
DTR1	2,6746	2,1784	0,8145
DTR2	3,298	2,7923	0,8467
DTR3	2,5772	2,2272	0,8642
DTR4	19,7472	17,3074	0,8764
DTR5	6,6432	5,7753	0,8694
DTR6	22,3865	16,4465	0,7347
DTR7	6,6944	5,6087	0,8378
DTR8	66,9183	55,6258	0,8312
DTR9	2,4696	2,1714	0,8793
DTR10	2,8034	2,5651	0,9150
DTR11	2,7494	2,2829	0,8303
DTR12	19,4095	17,5106	0,9022
DTR13	5,1167	4,4882	0,8772
DTR14	6,9887	5,9187	0,8469

Ідентифікатор тесту	Серед. час навчання, с		Відношення
	2 обчисл. вузли	3 обчисл. вузли	
DTR15	5,3098	4,7227	0,8894
DTR16	60,2058	49,345	0,8196

Середня різниця приросту продуктивності при навчанні сягає близько 17%.

Рисунок 59 – Графік для порівняння часу навчання на гетерогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

Таблиця 29 – Порівняння часу тестування на гетерогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

Ідентифікатор тесту	Серед. час тестування, с		Відношення
	2 обчисл. вузли	3 обчисл. вузли	
DTR1	0,1277	0,1061	0,8309
DTR2	0,16	0,1317	0,8231
DTR3	0,114	0,1004	0,8807
DTR4	0,1067	0,0914	0,8566

DTR5	0,1554	0,1258	0,8095
DTR6	0,4163	0,4085	0,9813
DTR7	0,1512	0,1324	0,8757
DTR8	0,4643	0,4374	0,9421
DTR9	0,107	0,0865	0,8084
DTR10	0,1344	0,1318	0,9807
DTR11	0,1171	0,0999	0,8531
DTR12	0,1154	0,1129	0,9783
DTR13	0,1652	0,1402	0,8487
DTR14	0,2367	0,2316	0,9785
DTR15	0,1489	0,1256	0,8435
DTR16	0,3364	0,2965	0,8814

Середня різниця приросту продуктивності при навчанні сягає близько 10%.

Рисунок 60 – Графік для порівняння часу тестування на гетерогенних кластерах з 2-ма та 3-ма обчислювальних вузлами

ВИСНОВКИ

Під час виконання дипломної роботи було:

- виконано дослідження продуктивності роботи кластера Apache Spark на платформі Azure;
- розроблені допоміжні програмні засоби для обробки результатів;
- розроблено інформаційне забезпечення для моніторингу кластеру та порівняння різних показників кластеру;
- проаналізовано отримані результати дослідження.

При виконанні дослідження були використані такі програмні засоби: платформа Microsoft Azure, фреймворк Apache Spark, бенчмарк Spark-Perf, система моніторингу Grafana, система моніторингу та зберігання метрик Graphite, програми для обробки результатів AmbariMetricsReport, SparkMeasureReport, ambari2graphite.

Результати дослідження можуть бути застосовані для вибору конфігурації кластеру для розв’язання конкретних задач машинного навчання. Також існує багато перспектив подальших досліджень, таких як тестування та аналіз результатів на інших тестах продуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Apache Spark™ [Електронний ресурс] – Режим доступу: https://spark.apache.org/.
2. RDD programming guide. [Електронний ресурс] – Режим доступу: https://spark.apache.org/docs/latest/rdd-programming-guide.html.
3. Apache Spark Ecosystem and Spark Components. [Електронний ресурс] – Режим доступу: https://data-flair.training/blogs/apache-spark-ecosystem-components/.
4. Streaming [Електронний ресурс] – Режим доступу: https://spark.apache.org/streaming/.
5. Apache Spark: Config Cheatsheet [Електронний ресурс] – Режим доступу: https://c2fo.io/c2fo/spark/aws/emr/2016/07/06/apache-spark-config-cheatsheet/.
6. Apache Spark: Config Cheatsheet (Part 2) [Електронний ресурс] – Режим доступу: https://c2fo.io/c2fo/spark/aws/emr/2016/09/01/apache-spark-config-cheatsheet-part2/.
7. How Data Partitioning in Spark helps achieve more parallelism? [Електронний ресурс] – Режим доступу: https://www.dezyre.com/article/how-data-partitioning-in-spark-helps-achieve-more-parallelism/297.
8. Apache Spark on YARN – Performance and Bottlenecks [Електронний ресурс] – Режим доступу: http://www.treselle.com/blog/apache-spark-on-yarn-performance-and-bottlenecks/.
9. Running Spark on YARN + settings Yarn [Електронний ресурс] – Режим доступу: https://spark.apache.org/docs/latest/running-on-yarn.html.
10. Hadoop: Capacity Scheduler [Електронний ресурс] – Режим доступу: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html.
11. Best Machine Learning as a Service Companies [Електроний ресурс] - Режим доступу: https://jelvix.com/blog/machine-learning-as-service-companies
12. WHAT IS MACHINE LEARNING AS A SERVICE (MLAAS)? [Електроний ресурс] - Режим доступу: https://analyticsindiamag.com/what-is-machine-learning-as-a-service-mlaas/
13. Top 5 Machine Learning-as-a-Service providers [Електроний ресурс] - Режим доступу: https://jaxenter.com/top-5-machine-learning-service-providers-141275.html
14. Machine Learning as a Service (MLaaS) [Електроний ресурс] - Режим доступу: https://www.techopedia.com/definition/32434/machine-learning-as-a-service-mlaas
15. AWS vs Azure vs Google Cloud Platform – Analytics & Big Data [Електроний ресурс] - Режим доступу: https://blogs.endjin.com/2016/08/aws-vs-azure-vs-google-cloud-platform-analytics-big-data/

16. Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watson [Електроний ресурс] - Режим доступу: https://www.topbots.com/comparing-machine-learning-as-a-service-platforms/

17. Сведения об Azure HDInsight [Електроний ресурс] - Режим доступу: https://docs.microsoft.com/ru-ru/azure////hdinsight/hdinsight-overview

18. Общие сведения о службах машинного обучения в HDInsight [Електроний ресурс] - Режим доступу: https://docs.microsoft.com/ru-ru/azure////hdinsight/r-server/r-server-overview

19. Apache Spark в Azure HDInsight [Електроний ресурс] - Режим доступу: https://docs.microsoft.com/ru-ru/azure/hdinsight/spark/apache-spark-overview

20. Apache Spark: что там под капотом? [Електроний ресурс] - Режим доступу: https://habr.com/ru/post/251507/

21. Apache Spark - What is Spark? [Електроний ресурс] - Режим доступу: https://databricks.com/spark/about

22. MLlib is Apache Spark's scalable machine learning library [Електроний ресурс] - Режим доступу: https://spark.apache.org/mllib/

23. Обучение на больших данных: Spark MLlib [Електроний ресурс] - Режим доступу: https://habr.com/ru/company/mlclass/blog/251471/

24. Machine Learning with Spark MLlib [Електроний ресурс] - Режим доступу: https://www.baeldung.com/spark-mlib-machine-learning

25. Spark MLlib – Machine Learning Library Of Apache Spark [Електроний ресурс] - Режим доступу: https://www.edureka.co/blog/spark-mllib/#MLlib_Algorithms

26. Machine Learning Library (MLlib) Guide [Електроний ресурс] - Режим доступу: https://spark.apache.org/docs/latest/ml-guide.html

27. Apache Spark Machine Learning Algorithm – Example & Clustering [Електроний ресурс] - Режим доступу: https://data-flair.training/blogs/spark-machine-learning-algorithm/

28. Apache Spark MLlib [Електроний ресурс] - Режим доступу: https://docs.databricks.com/applications/machine-learning/mllib/index.html

29. Spark-Perf - Performance tests for Apache Spark [Електроний ресурс] - Режим доступу: https://github.com/databricks/spark-perf

30. Spark-Bench - Spark-Bench is a flexible system for benchmarking and simulating Spark jobs. [Електроний ресурс] - Режим доступу: https://codait.github.io/spark-bench/

31. Apache Spark-Bench: Simulate, Test, Compare, Exercise, and Yes, Benchmark [Електроний ресурс] - Режим доступу: https://databricks.com/session/apache-spark-bench-simulate-test-compare-exercise-and-yes-benchmark

32. Standard Performance Evaluation Corporation. [Електроний ресурс] - Режим доступу: https://www.spec.org/

33. Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. BigBench: towards an industry standard benchmark for big data analytics. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13). ACM, New York, NY, USA, 1197-1208.

34. Alexander Alexandrov, Kostas Tzoumas, and Volker Markl. Myriad: scalable and expresive data generation. Proc. VLDB Endow. 5, 12 (August 2012), 1890-1893

35. Rabl, Tilmann, Michael Frank, Hatem Mousselly Sergieh, and Harald Kosch. A data generator for cloud-scale benchmarking. In Performance Evaluation, Measurement and Characterization of Complex Systems, pp. 41-56. Springer Berlin Heidelberg, 2011.

36. Linked Data Benchmark Council Social Network Benchmark (LDBC-SNB) Generator,
[Електроний ресурс] - Режим доступу: https://github.com/ldbc/ldbc_snb_datagen

37. Graph500 generator, [Електроний ресурс] - Режим доступу: http://www.graph500.org/specifications

38. DOTS: Database Opensource Test Suite. [Електроний ресурс] - Режим доступу: http://ltp.sourceforge.net/documentation/how-to/dots.php

39. Infor LN Baan. [Електроний ресурс] - Режим доступу: www.infor.com/product_summary/erp/ln/

40. T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan Linkbench: A database benchmark based on the facebook social graph In Proceedings of the 2013 ACM SIGMOD, pages 1185–1196, 2013.

41. Apache GridMix. [Електроний ресурс] - Режим доступу: http://hadoop.apache.org/docs/r1.2.1/gridmix.html.

42. Apache PigMix. [Електроний ресурс] - Режим доступу: https://cwiki.apache.org/confluence/display/PIG/PigMix.

43. Sort Benchmark. [Електроний ресурс] - Режим доступу: http://sortbenchmark.org/.

44. S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang. The HiBench benchmark suite: Characerization of the MapReduce-based data analysis. In 26th IEEE ICDEW, pages 41–51, March 2010

45. Performance portal for Apache Spark. http://01org.github.io/sparkscore/plaf1.html.

46. A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. A comparison of approaches to large-scale data analysis. In Proceedings of the 2009 ACM SIGMOD, pages 165–178, 2009.

47. L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia,Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu. Bigdatabench: A big data benchmark suite from internet services. In IEEE 20th HPCA, pages 488–499, Feb 2014.

48. AMPLab Big Data Benchmark. [Електроний ресурс] - Режим доступу: https://amplab.cs.berkeley.edu/benchmark/

49. M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the clouds: A study of emerging scaleout workloads on modern hardware. In Proceedings of the 17th ACM ASPLOS, pages 37–48, 2012

50. A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen. Bigbench: Towards an industry standard benchmark for big data analytics. In Proceedings of the 2013 ACM SIGMOD, pages 1197–1208, 2013

51. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In Proceedings of the 1st ACM SOCC, pages 143–154, 2010.

52. Min Li, Jian Tan, Yandong Wang, Li Zhang, and Valentina Salapura. SparkBench: a comprehensive benchmarking suite for in memory data analytic platform Spark. In Proceedings 20 of the 12th ACM International Conference on Computing Frontiers (CF '15). ACM, New York, NY, USA, , Article 53

53. P. A. Boncz, T. Neumann, and O. Erling. TPC-H Analyzed: Hidden Messages And Lessons Learned From An Influential Benchmark. Proceedings of the TPC Technology Conference on Performance Evaluation and Benchmarking TPCTC, 2013.

54. O. Erling, A. Averbuch, J. L. Larriba Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat, Minh-Duc Pham, Peter Boncz. The LDBC Social Network Benchmark: Interactive Work-load. Proceedings of SIGMOD 2015, Melbourne.

55. Mihai Capotă, Tim Hegeman, Alexandru Iosup, Arnau Prat, Orri Erling, Peter Boncz. Graphalytics: A Big Data Benchmark for Graph-Processing Platforms, Proceedings of GRADES2015, co-located with ACM SIGMOD/PODS 2015,

56. R. Angles, P. A. Boncz, J.-L. Larriba-Pey, I. Fundulaki, T. Neumann, O. Erling, P. Neubauer, N. Martínez-Bazan, V. Kotsev, I. Toma (2014): The linked data benchmark council: a graph and RDF industry benchmarking effort, SIGMOD Record 43(1): 27-31

57. PigMix. https://cwiki.apache.org/confluence/display/PIG/PigMix

58. Kim, Kiyoung, Kyungho Jeon, Hyuck Han, Shin-gyu Kim, Hyungsoo Jung, and Heon Y. Yeom. "MRBench: A benchmark for MapReduce framework." In IEEE ICPADS, 2008.3 Methods for Parallelization in Spark [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/3-methods-for-parallelization-in-spark-6a1a4333b473.

59. Sizes for Linux virtual machines in Azure [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/azure/virtual-machines/linux/sizes.

60. General purpose virtual machine sizes [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/azure/virtual-machines/linux/sizes-general.

61. Memory optimized virtual machine sizes [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/azure/virtual-machines/linux/sizes-memory.