Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

з лабораторної роботи № 4

дисципліни

“ТЕХНОЛОГІЇ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ В УМОВАХ ВЕЛИКИХ ДАНИХ”

на тему:

„Big Data з використанням засобів Apache Spark”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконали студенти** |  | * *ІП-01мн Семченко Андрій* * *ІП-01мн Кошовець Євген* * *ІТ-01мн Васюк Владислав* * *ІТ-01мн Минзар Богдан* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | *доц. Жереб К. А.* |  |  |
|  |  | (прізвище, ім’я, по батькові ) |  |  |

Київ 2021

**ЗМІСТ**

[1 Постановка задачі 3](#_Toc91059620)

[2 ВИКОРИСТАНІ БІБЛІОТЕКИ, ФРЕЙМВОРКИ 4](#_Toc91059621)

[3 ОПИС РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 5](#_Toc91059622)

[3.1 Загальна концепція 5](#_Toc91059623)

[3.2 Деталі роботи програмного забезпечення 5](#_Toc91059624)

[4 ОТРИМАНІ РЕЗУЛЬТАТИ 7](#_Toc91059625)

[o Вимірювання часу роботи в залежності від параметрів 7](#_Toc91059626)

[5 ВИСНОВОК 9](#_Toc91059627)

[5 ПОСИЛАННЯ 10](#_Toc91059628)

[6 ДОДАТОК 1 - ЛОГ РОБОТИ ПРОГРАМИ 11](#_Toc91059629)

# Постановка задачі

Необхідно реалізувати вирішення обраної задачі з використанням технології Apache Spark. Можна запустити реалізацію локально, але якщо є можливість запустити на розподіленій системі – за це можна отримати додаткові бали. Порівняти реалізації лабораторних робіт No3 та No4. Результатом виконання даної лабораторної роботи є працююча програма, а також звіт про використані технології та можливості, з результатами вимірів.

Ідея полягає у тому, щоб шляхом аналізу вмісту публічного репозиторію знайти потенційні вразливості, що дозволяють втрутитись у роботу програмного забезпечення. Причому вразливості не тільки у самій реалізації програмного забезпечення, але і вразливості, спричинені недбалим обігом sensitive data, наприклад:

* Зберігання ключів доступу у файлах, що відстежуються VCS
* Зберігання ключів доступу прямо у тексті програмного забезпечення
* Зберігання бекапів у файлах, що відстежуються VCS

Зберігання sensitive data у файлах, що відстежуються системою контролю версій призводить до того, що будь хто може завантажити ці дані і використати для втручання у роботу програмного забезпечення, викрадення даних користувачів, тощо.

# ВИКОРИСТАНІ БІБЛІОТЕКИ, ФРЕЙМВОРКИ

Розроблена програма використовує фреймворк PySpark та написана на мові програмування Python 3.7 у вигляді Jupyter ноутбука. Мотивація викоритовувати Python – вже є досвід роботи з PySpark, тож це значно спростить процес розробки та тестування коду. Для роботи з Github було використано бібліотеку GitPython [1].

# ОПИС РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Загальна концепція

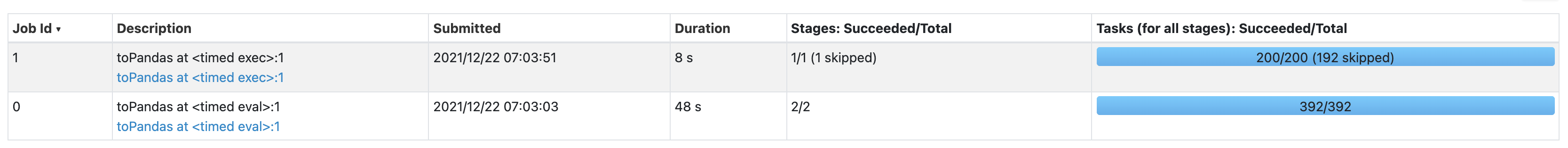
Програмне забезпечення завантажує вміст публічного Github-репозиторію у тимчасову директорію. Далі, кожен файл даної директорії аналізується на предмет наявності в ньому певних патернів, що можуть свідчити про вразливість. Сам паттерн описується за допомогою регулярного виразу.

## Деталі роботи програмного забезпечення

Загальний принцип роботи:

1. Програма запускається у вигляді Python ноутбука з використанням Jupyter. Налаштування підключення до Spark кластеру задано в коді програми та у файлі spark-defaults.conf.
2. Після створення контексту Spark, програма створює DataFrame зі списком посилань на git-репозиторії, що був попередньо заданий в коді програми (однак на вхід можна використати інші джерела).
3. За допомогою UDF [2] виконується завантаження git-репозиторіїв в тимчасові папки кожного з воркерів, після чого зміст та шлях кожного файла в репозиторії зберігається в DataFrame. На виході маємо таблицю з полями: repo (шлях до репозиторію), path (відносний шлях до файлу в репозиторії), content (зміст файлу якщо він текстовий).
4. DataFrame з попереднього кроку за допомогою ще однієї UDF аналізуємо на вразливості: перевіряється шлях до кожного файлу та вміст. На виході отримуємо DataFrame з полями: repo (шлях до репозиторію), path (відносний шлях до файлу в репозиторії), issueType (тип вразливості), issueDescription (опис вразливості), lineNumber (номер рядку в файлі де знайдено вразливість).
5. Результати роботи програми експортується у форматі JSON за допомогою бібліотеки pandas.

Так виглядає список Completed Jobs в Spark UI після виконання:



# ОТРИМАНІ РЕЗУЛЬТАТИ

## Вимірювання часу роботи в залежності від параметрів

Всі тести проводились на Kubernetes кластері з 5 машин (на базі процесора Intel(R) Xeon(R) E-2286G CPU @ 4.00GHz, 6 ядер, 12 потоків; 64 GB оперативної памʼяті на кожній)

В кожному тесті змінено два основні параметри: максимальна кількість воркерів (spark.executor.instances) та кількість використаних ядер на кожному воркері (spark.executor.cores).

|  |  |  |
| --- | --- | --- |
| spark.executor.instances | spark.executor.cores | **time (seconds)** |
| **1** | **1** | **56** |
| **1** | **2** | **36** |
| **1** | **4** | **20.2** |
| **2** | **1** | **33.2** |
| **2** | **2** | **19.34** |
| **2** | **4** | **18.4** |
| **4** | **1** | **23.4** |
| **4** | **2** | **18.02** |
| **4** | **4** | **17.7** |
| **8** | **1** | **20.5** |
| **8** | **2** | **16.6** |
| **8** | **4** | **16.6** |

Як бачимо, найшвидше програма відпрацьовує з кількістю воркерів = 8 та кількістю ядер більше 2. Також можна помітити, що в програмі є «вузьке» місце – завантаження репозиторіїв; при подальших тестах на більшій кількісті воркерів стало зрозуміло, що час не скорочується, оскільки завантаження найбільшого репозиторію займає близько 15 секунд.

# 5 ВИСНОВОК

В рамках даної лабораторної роботи було розроблено програмне забезпечення на основі Apache Spark, що проводить аналіз Github-репозиторіїв на наявність в них типових вразливостей.

В рамках реалізованого алгоритму вдалося отримати пришвидшення в порівнянні з однопоточною реалізацією у 3.5 рази на конкретно взятому прикладі навантаження.

Було порівняно результати роботи програми з різними налаштуваннями паралелізму, найкращих результатів (16.6с) досягли при використанні 8 воркерів та 2+ ядер CPU на кожному воркері.

# ПОСИЛАННЯ

1. <https://gitpython.readthedocs.io/en/stable/>

[2] <https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.functions.udf.html>

# ДОДАТОК 1 - ЛОГ РОБОТИ ПРОГРАМИ

