VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**Trần Quốc Bảo – 521H0494**

**Bùi Hải Dương – 521H0220**

**Lê Nguyễn Viết Hiệp – 521H0398**

**Bùi Anh Phú – 521H0508**

**Nguyễn Hoàng Phúc – 521H0511**

**MIDTERM REPORT**

**MINING MASSIVE DATA SETS**

**HO CHI MINH CITY, 2024**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**Trần Quốc Bảo – 521H0494**

**Bùi Hải Dương – 521H0220**

**Lê Nguyễn Viết Hiệp – 521H0398**

**Bùi Anh Phú – 521H0508**

**Nguyễn Hoàng Phúc – 521H0511**

**MIDTERM REPORT**

**MINING MASSIVE DATA SETS**

Instructed by

**MCs. Nguyen Thanh An**

**HO CHI MINH CITY, 2024**

ACKNOWLEDGMENT

We would like to sincerely thank teacher Nguyen Thanh An for imparting basic knowledge and wholeheartedly guiding us so that we can perform this midterm essay well.

*Ho Chi Minh City, March 14th, 2024*

*Authors*

**DECLARATION OF AUTHORSHIP**

We now declare that this thesis was carried out by myself under the guidance and supervision of Mr.Nguyen Thanh An; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by our work and have been duly acknowledged in the reference part.

In addition, other comments, reviews, and data used by other authors, and organizations have been acknowledged, and explicitly cited.

**We will take full responsibility for any fraud detected in my thesis**. Ton Duc Thang University is unrelated to any copyright infringement caused by our work (if any).

*Ho Chi Minh City, March 14th, 2024*

*Authors*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh City, March 14th, 2024*

*Authors*

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

ABSTRACT

All tasks in the essay are done in Python language, performed on Google Collab, and using the PySpark library to perform.

* In task 1, leverage RDDs to handle the raw data. It reads the baskets.csv file, parses each line as a basket, and potentially saves the processed data for further analysis.
* Task 2, Employs DataFrames for structured data manipulation
* In task 3, we implement the PCY algorithm using PySpark for frequent itemset mining

CONTENTS

[ACKNOWLEDGMENT i](#_Toc161220261)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc161220262)

[ABSTRACT iv](#_Toc161220263)

[CONTENTS 1](#_Toc161220264)

[ABBREVIATIONS 2](#_Toc161220265)

[CHAPTER 1 – RDD 3](#_Toc161220266)

[1.1 Overview of RDD of PySpark library 3](#_Toc161220267)

[1.2 Solution and pseudo-code 7](#_Toc161220268)

[1.3 Table of complete 7](#_Toc161220269)

[CHAPTER 2 – DATA FRAME 10](#_Toc161220270)

[2.1 Overview of Data Frame 10](#_Toc161220271)

[2.2 Solution and pseudo-code 10](#_Toc161220272)

[2.3 Table of complete 10](#_Toc161220273)

[CHAPTER 3 – PCY 11](#_Toc161220274)

[3.1 Overview of PCY algorithm 11](#_Toc161220275)

[3.2 Solution and pseudo-code 11](#_Toc161220276)

[3.3 Table of complete 11](#_Toc161220277)

ABBREVIATIONS

**DANH MỤC HÌNH**

[Hình 2.1: Kiến trúc FTP 1](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 1](#_Toc387689363)

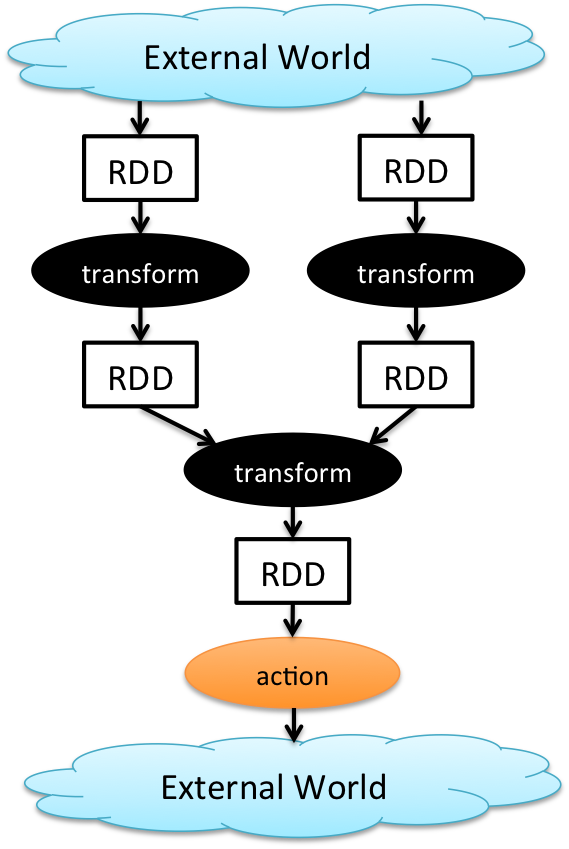
CHAPTER 1 – RDD

1.1 Overview of RDD of PySpark library

PySpark is the Python API for Apache Spark. It allows you to use Python for the processing of time and massive volumes of data in a Distributed Environment. With the power of Apache Spark, PySpark is a combination of Python knowledge and ease of use that makes it possible to perform data processing and analysis at any size for anyone familiar with Python language. The PySpark library supports all Spark features such as SQL, DataFrames, Structured Streaming, Machine LearningMLlib, and SparkCore.

Resilient Distributed Datasets (RDDs) are a fundamental building block of PySpark. They are collections of data elements that are distributed across many computing nodes and can be processed in parallel. Once you create an RDD, you cannot change it. Each record in RDD is divided into logical partitions, which can be computed on different nodes of the cluster.

RDDS can be performed on two types of operations: Transformation and Actions



*How do transformations and actions work? (h1)*

* **Transformation:** In PySpark, transformations are fundamental operations that manipulate and reshape RDDs to prepare them for analysis. Unlike actions (which return a value or write data to external storage), transformations create new RDDs based on existing ones

*Key Characteristics of Transformations:*

Create new RDDs without changing originals (immutable).

Defined but not run until needed (lazy evaluation).

Often use functions to process elements (functional style).

*Common transformation operations:*

* + map(): Applies a function to each element of an RDD, creating a new RDD with the transformed elements
  + filter(): Selects elements that satisfy a filtering condition, returning a new RDD with the filtered elements.
  + distinct(): Returns a new RDD containing unique elements from the original RDD.
  + flatMap(func): Similar to map, but the function can return multiple elements per input element, resulting in a flattened RDD.
  + groupBy(func): Groups elements with the same key (determined by the function) into key-value pairs, creating a Pair RDD.
* **Actions**: Actions are the operations that trigger actual computations on the data and return results to the driver program. Unlike transformations (which create new RDDs), actions force the execution of the entire lineage of transformations leading up to them.

*Key Points:*

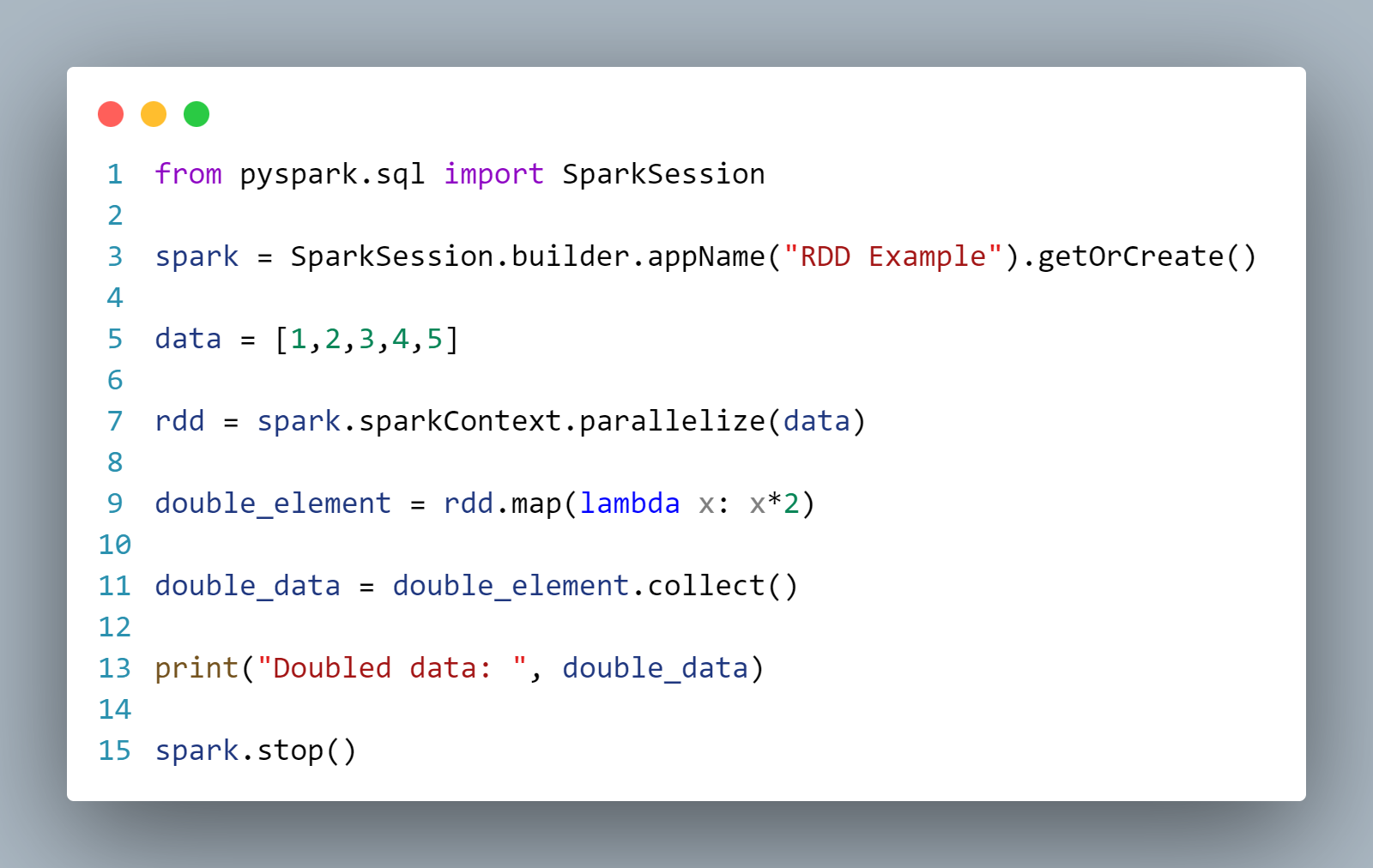
* + Materialization: Actions force the RDD computations to happen, returning a value or writing data to external storage.
  + Trigger Point: They mark the end of a data processing pipeline within an RDD.
  + Diverse Results: Actions can return a single value (e.g., count), a collection (e.g., collect), or write data to external systems (e.g., saveAsTextFile).

*Common Actions:*

* + collect(): Gathers all elements of the RDD into a list in the driver program's memory (caution: use with smaller datasets).
  + count(): Returns the total number of elements in the RDD.
  + first(): Retrieves the first element of the RDD.
  + take(n): Returns the first n elements of the RDD.
  + saveAsTextFile(path): Saves the RDD as a text file.

|  |  |
| --- | --- |
| **Pros And Cons** | |
| ***Fault Tolerance:*** RDDs are designed to be resilient. In case of node failures, Spark can efficiently recompute lost partitions across the cluster.  ***Immutability***: Transformations create new RDDs, leaving the originals unchanged. This simplifies debugging and ensures data integrity.  ***Lazy Evaluation***: Spark only executes transformations when an action is triggered, optimizing performance for complex pipelines  ***Distributed Processing***: RDDs leverage Spark's distributed architecture, enabling efficient processing of large datasets across multiple nodes. | ***Verbosity:*** Low-level API can lead to more verbose code compared to higher-level abstractions like DataFrames or Datasets.  ***Limited Optimization:*** Spark cannot always optimize transformations within RDDs due to the black-box nature of lambda functions.  ***Memory Management:*** Large RDDs might not fit entirely in memory, potentially leading to disk spills and performance degradation.  ***Less Intuitive:*** The functional style and low-level nature can have a steeper learning curve compared to more SQL-like approaches. |
|
|

Example of basic RDD operation (transformation):



Explain:

Import SparkSession to create a connection to the Spark cluster.

Define a list of numbers (data)

Use spark.sparkContext.parallize() to convert the list into an RDD, making it distributable across the cluster.

Utilize rdd.map()transformation to double each element in the RDD using a lambda function.

The collect() method retrieves the results back to the driver program

Output:



1.2 Solution and pseudo-code

1.3 Table of complete

|  |  |  |
| --- | --- | --- |
| Function | Requirements | Percentage |
| f1 | Find the list of distinct products.  Results are sorted in the ascending order of product names.  3. Print down 10 first and 10 last products on the screen.  4. Save to folder f1. | 100% |
| f2 | * Find the list of distinct products and their frequency of being purchased.   Results are sorted in the ascending order of product names.  3. Select the top 100 products with the highest frequency.  4. Draw a bar chart to visualize their frequency.  5. Print the result on the screen  6. Save to folder f2. | 100% |
| f3 | 1. Find the number of baskets for each member. A basket is a set of distinct products bought by a member on a date.  2. Results are sorted in descending order of the number of baskets.  3. Select the top 100 members with the largest number of baskets.  4. Draw a bar chart to visualize the number of  baskets.  5. Print the result on the screen.  6. Save to folder f3. | 100% |
| f4 | 1. Find the member that bought the largest number of distinct products.  2. Find the product that is bought by the most members.  3. Print the result on the screen.  4. Save to folder f4 | 100% |

CHAPTER 2 – DATA FRAME

2.1 Overview of Data Frame

2.2 Solution and pseudo-code

2.3 Table of complete

CHAPTER 3 – PCY

3.1 Overview of PCY algorithm

3.2 Solution and pseudo-code

3.3 Table of complete

ASSIGNED TASKS AND COMPLETE PERCENTAGE

|  |  |  |
| --- | --- | --- |
| **Task** | **Name** | **Complete Percentage** |
| Task 1 | Tran Quoc Bao | 100% |
| Le Nguyen Viet Hiep |
| Task 2 | Bui Hai Duong | 0% |
| Task 3 | Bui Anh Phu | 0% |
| Nguyen Hoang Phuc |
| Task 4 | Members | 0% |

REFERENCES

**Vietnamese**

**English**

[1] Apache Spark RDD Tutorial | Learn with Scala Examples [Online]. Spark By Examples.<https://sparkbyexamples.com/pyspark-tutorial/> (accessed Mar 9, 2024)

[2] PySpark - RDD [Online]. Tutorialspoint. tutorialspoint.com.<https://www.tutorialspoint.com/pyspark/pyspark_rdd.htm> (accessed Mar 9, 2024)

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn