ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

HỆ ĐIỀU HÀNH (CO2017) - Học phần mở rộng

Thu thập và phân tích hoạt động học tập của người học trên các Learning Management System

Giảng viên hướng dẫn: TS.Trần Quang Hùng
Sinh viên: Nguyễn Duy Khang - 2011364
Phan Phước Minh - 2010418
Nguyễn Đức An - 2010102
Đoàn Trần Cao Trí - 2010733

Thành phố Hồ Chí Minh, tháng 4 năm 2022

Mục lục

1 Giới thiệu

1.1 Nội dung đề tài

Giáo dục 4.0 và chuyển đổi số giáo dục đang là một mô hình giáo dục thông minh và có những thay đổi rất lớn trong công tác giáo dục ở Việt Nam nói riêng và cả thế giới nói chung. Mô hình giáo dục 4.0 được coi là động lực cho các em học sinh, sinh viên có điều kiện công bằng để tham gia học tập với các bạn cùng trang lứa. Càng thiết thực hơn nữa, trong giai đoạn giãn cách xã hội tại Việt Nam do dịch Covid chuyển biến phức tạp, càng nhiều tầng lớp xã hội tham gia vào giáo dục 4.0: từ các bạn học sinh, sinh viên đến những người công nhân, viên chức,...

Chính vì lẽ đó là các nền tảng hệ thống quản lý học tập như Moodle, Google Classroom, Canvas,... càng phải phát triển để đáp ứng lượng truy cập khổng lồ này, phục vụ phân tích, đánh giá chất lượng của người học trên hệ thống.

Với những thách thức đó, nhóm em chọn đề tài phát triển kiến trúc phần lõi thu gom dữ liệu của người học, các hoạt động trên hệ thống Learning Mangament System (LMS). Đồng thời, xây dựng các đánh giá hiệu quả của người học trên các khóa học và dự đoán kết quả của người học (dựa trên lịch sử). Cung cấp cho nhà quản lý nhà trường và khoa góc nhìn về toàn cục.

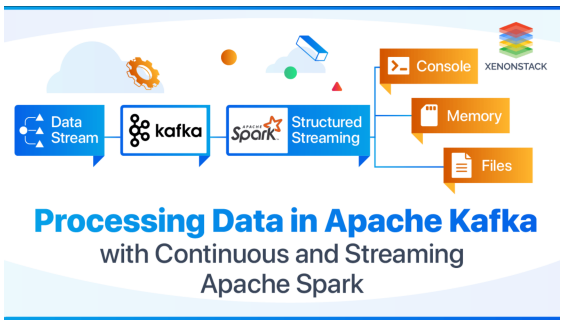
1.2 Ý nghĩa đề tài

Hệ thống được thực hiện nghiên cứu độc lập với các hệ thống Learning Mangament System (LMS). Nghĩa là hệ thống được xây dựng có thể phân tích và đưa ra kết quả từ các bộ dữ liệu thu thập được từ các LMS khác nhau.

Hỗ trợ đáp ứng nhu cầu về xử lý lượng lớn dữ liệu trong thời đại Big Data. Tốc độ phản hồi từ sever đến người dùng nhanh như đề xuất liên tục, thống kê; nâng cao trải nghiệm người dùng.

1.3 Yêu cầu đề tài

- Có sử dụng kiến thức về dữ liệu lớn, trí tuệ nhân tạo, Data Lake.
- Tìm hiểu về Learning Management System như Moodle
- Tìm hiểu về Kafka, Spark



Hình 1: Trình tự thực hiện

1.4 Kết quả đầu ra của đề tài

Khi hoàn thành xong đề tài này, nền tảng của nhóm có thể giúp giáo viên so sánh sự tham gia và kết quả với các khóa học khác.

Ngoài ra, cung cấp thông tin để giáo viên có thể thực hiện như liên hệ trực tiếp với học sinh, chỉ định nội dung đặc biệt, khuyến khích tham gia hoặc cung cấp dịch vụ dạy kèm đặc biệt.

Về phía học sinh, sinh viên, có thể tìm hiểu về hiệu suất học tập cá nhân, hoặc so sánh với nhóm.

2 Learning Management Systems (LMS)

2.1 Tổng quan

LMS, theo nghĩa tiếng Việt là *"hệ thống quản lý học tập"*, là những hệ thống hỗ trợ việc quản lý, giám sát người học thông qua việc cung cấp các file tài liệu, tổ chức các bài kiểm tra hay theo dõi hành vi của người học.

2.2 Moodle

Moodle [3] là LMS phổ biến nhất tại các trường thuộc hệ thống Đại học Quốc gia TPHCM ở Việt Nam. Moodle là một dự án mã nguồn mở, có thể dễ dàng mở rộng thêm các tính năng có sẵn thông qua các plugins.

2.3 Các loại dữ liệu từ học viên

- Những thông tin liên quan đến submission (nộp bài).
- Những thông tin về tương tác với khóa học (xem khóa học, xem các module).

2.4 xAPI

Ở đây, để thống nhất giữa các LMS khác nhau, người ta đưa ra một chuẩn có tên là xAPI.
Khi đó, những dữ liệu thu thập được mô tả ở phần trên sẽ được biểu diễn như sau:

```
{
  "actor": {
    "name": "Sally Glider",
    "mbox": "mailto:sally@example.com"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/completed",
    "display": {"en-US": "completed"}
  },
  "object": {
    "id": "http://example.com/activities/solo-hang-gliding",
    "definition": {
      "name": { "en-US": "Solo Hang Gliding" }
    }
  },
  "result": {
    "completion": true,
    "success": true,
    "score": {
      "scaled": .95
    }
  }
}
```

3 Kafka

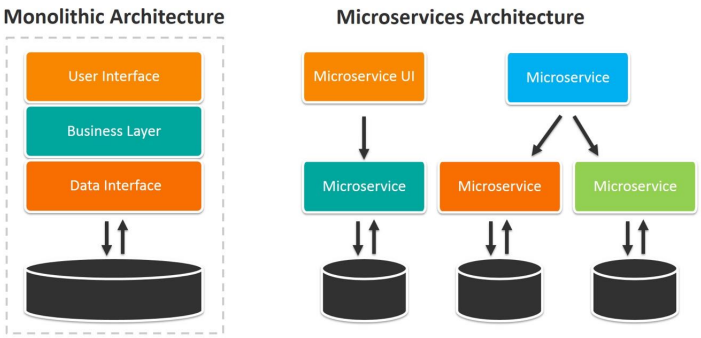
3.1 Giới thiệu về Kafka

3.1.1 Sơ lược nguyên nhân ra đời

Kiến trúc nguyên khối (monolithic architecture): là kiến trúc sơ khai khi người lập trình hiện thực một dự án. Khi sử dụng kiến trúc này, người dùng hiện thực các thành phần khép kín và kết nối với nhau. Điểm lợi của kiến trúc này là dễ sử dụng, dễ hiện thực cho người lập trình. Tuy nhiên nó lại có một số nhược điểm như sau:

- Với các dự án lớn, số lượng code lớn, sẽ gây ra mất kiểm soát, khó quản lí code.
- Khi muốn chỉnh sửa, cần phải đồng bộ giữa các thành phần, làm mất thời gian.
- Khi có một thành phần bị lỗi, toàn bộ hệ thống sẽ phải dừng lại.

Vì vậy, kiến trúc microservices được ra đời để giúp quản lí được các dự án lớn. Để xử lý được tốt các tương tác giữa các thành phần, Kafka được LinkedIn tạo ra như một môi trường để giao tiếp, tương tác chung.



Hình 2: Monolithic and Microservices Architecture

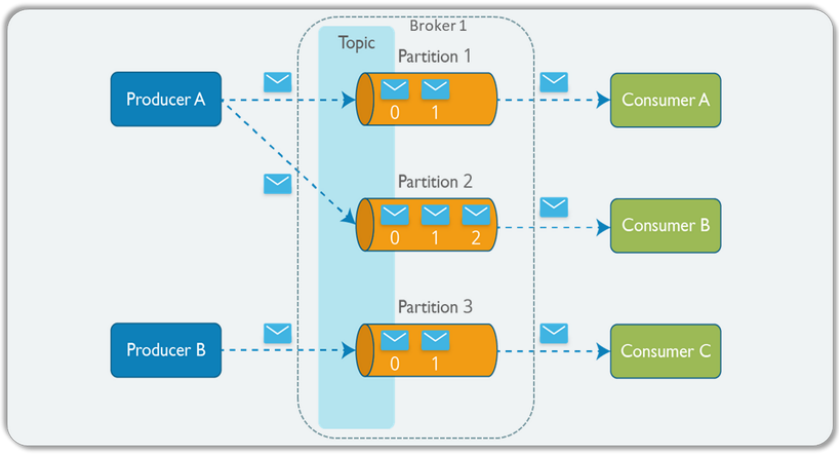
3.1.2 Định nghĩa, ưu điểm, tính năng nổi bật của Kafka

Định nghĩa: Kafka [1] là một hệ thống message theo cơ chế Pub-Sub. Nó cho phép các nhà sản xuất (gọi là producer) viết các message vào Kafka mà một, hoặc nhiều người tiêu dùng (gọi là consumer) có thể đọc, xử lý được những message đó.

Một số thành phần chính của Kafka:

- **Message**: Thông tin được gửi đi, có thể là text, binary, Json hoặc một định dạng format nào đó.
- **Broker**: Một host có thể chạy nhiều server kafka, mỗi server như vậy gọi là một broker. Các broker này cùng trỏ tới chung 1 zookeeper, gọi là cụm broker(hay là Clusters). Broker là nơi chứa các partition. Một broker có thể chứa nhiều partition.
- **Topic**: Là nơi chứa các message được publish từ Producer tới Kafka, nhìn về mặt database thì topic giống như một table trong cơ sở dữ liệu quan hệ, và mỗi message như một bản ghi của table đó.

- **Partition**: Nơi lưu trữ message của topic. Một topic có thể có nhiều partition. Khi khởi tạo topic cần set số partition cho topic đó. Partition càng nhiều thì khả năng làm việc song song cho đọc và ghi được thực hiện nhanh hơn. Các message trong partition được lưu theo thứ tự bất biến(offset). Một partition sẽ có tối thiểu 1 replica để đề phòng trường hợp bị lỗi. Số lượng replica luôn nhỏ hơn số lượng broker.
- **Producer**: Chương trình/service tạo ra message, đẩy message publish vào Topic.
- **Consumer**: Chương trình/service có chức năng subscribe vào một Topic để tiêu thụ, xử lý các message đó.



Hình 3: Mô hình cấu trúc Kafka chi tiết

Với thiết kế riêng của mình, Kafka có những ưu điểm sau:

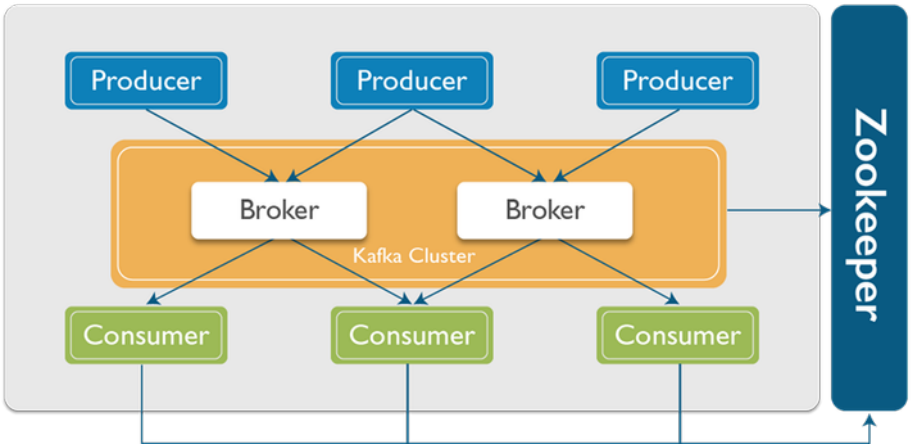
- Xử lý dữ liệu thời gian thực (real-time)
- Dữ liệu phân tán (distributed): dữ liệu được phân tán ra nhiều cụm khác nhau trong cluster.
- Khả năng phục hồi (resilent): Khi hệ thống gặp lỗi, có thể phục hồi được dữ liệu ngay trước khi lỗi.
- Khả năng chịu lỗi (fault torelant): Khi có một broker bị lỗi thì các broker khác đều có thể tiếp tục làm việc.
- Khả năng mở rộng (scalable).

3.1.3 Ứng dụng Kafka

Một số ứng dụng của Kafka là:

- Vận hành hệ thống: lưu trữ bộ số dữ liệu, theo dõi hoạt động, tổng hợp các hoạt động truy cập,...
- Phân tích dữ liệu: Xử lý dữ liệu dòng, thu thập dữ liệu,...
- Công nghệ phần mềm: Hệ thống tin nhắn, triển khai các thay đổi trên hệ thống thông qua các event,...

3.1.4 Sơ lược về Zookeeper



Hình 4: Mô hình Kafka cơ bản

Định nghĩa: Zookeeper là một dịch vụ tập trung để duy trì và cấu hình dữ liệu, cung cấp đồng bộ hóa linh hoạt và mạnh mẽ trong hệ thống phân tán.

Liên hệ giữa Zookeeper và Kafka: Zookeeper theo dõi, lưu trữ trạng thái của Kafka brokers, topics, partions,... và thông báo mọi thay đổi của Kafka. Kafka **không thể sử dụng** nếu không có Zookeeper.

3.2 Kafka Connect

Khi nói tới việc ứng dụng Kafka để xây dựng Data pipelines, ta có thể hình dung ra 2 trường hợp:

- Thứ nhất, data pipeline với Apache Kafka là một trong hai endpoint. Ví dụ: lấy dữ liệu từ Kafka đẩy vào S3(Amazon Simple Storage Service) , hoặc lấy dữ liệu từ Database đẩy vào Kafka
- Thứ hai, Sử dụng kafka để xây dựng một pipeline giữa 2 hệ thống riêng biệt. Ví dụ: lấy data từ Twitter đưa vào hệ thống Elasticsearch bao gồm việc lấy data từ Twitter đẩy vào Kafka sau đó từ kafka đưa vào Elasticsearch.

Apache Kafka có thể đóng vai trò là một buffer cực lớn và đáng tin cậy: sự độc lập giữa việc đọc và ghi, giữa chính các producer và các consumer kết hợp với khả năng bảo mật hiệu quả giúp Kafka đáp ứng yêu cầu của phần lớn data pipelines.

- **Timeliness**: Kafka - một streaming data platform với khả năng lưu trữ đáng tin cậy và khả năng mở rộng tốt có thể đáp ứng tốt từ data pipeline gần thời gian thực cho tới việc đồng bộ dữ liệu theo lô. Producer có thể ghi dữ liệu vào Kafka định kỳ hoặc bất cứ khi nào cần. Consumer có thể đọc dữ liệu mới nhất ngay khi nó được ghi vào, cũng có thể định kỳ đọc dữ liệu hàng giờ hoặc vài giờ một lần.
- **Reliability**: Đối với các hệ thống lớn, việc mất mát dữ liệu, trùng lặp dữ liệu có thể gây ra hậu quả lớn. Kafka đảm bảo đảm bảo yêu cầu "exactly-once" cho mỗi sự kiện được đưa vào hệ thống.

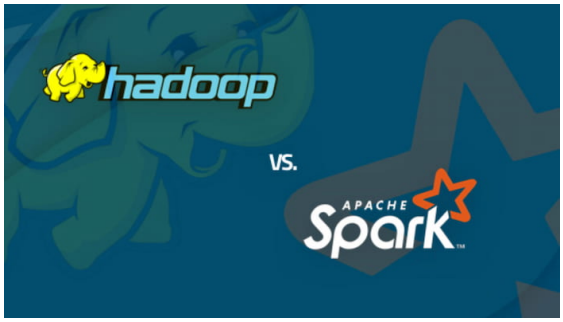
- **High and Varying Throughput**: Kafka rất linh hoạt trong việc thích ứng mỗi khi yêu cầu mở rộng hệ thống, các consumer và producer được thêm vào hệ thống một cách độc lập tùy thuộc vào yêu cầu đặt ra. Hoạt động như một bộ nhớ đệm cực lớn giữa producers và consumer, Kafka còn cung cấp một cơ chế backpressure giúp cho dữ liệu được giữ ở trong cụm cho đến khi consumer kịp xử lý.
- **Data Formats**: Việc lựa chọn định dữ liệu tùy thuộc vào yêu cầu đặt ra của mỗi hệ thống, nghiệp vụ khác nhau. Kafka hỗ trợ rất nhiều: XML, text, JSON, Avro, CSV, ... Producer và Consumer có thể sử dụng bất kì bộ serilizer nào để biểu diễn dữ liệu dưới bất kì định dạng nào bạn muốn.
- **Security**: Bảo mật luôn là vấn đề quan trọng. Kafka có thể đáp ứng tốt yêu cầu về bảo mật.Bạn có thể chắc chắn rằng dữ liệu của bạn khi đi qua Kafka có thể được mã hóa nếu bạn muốn vì bạn chủ động trong ghi dữ liệu. Kafka cũng hỗ trợ cơ chế xác thực (SASL), do đó bạn có thể yên tâm rằng những người có quyền đọc mới được đọc topic của bạn. Bạn cũng có thể custom thêm để ghi log lại các sự kiện mà bạn muốn theo dõi.

4 Apache Spark

4.1 Tìm hiểu lý thuyết Apache Spark

4.1.1 Lịch sử và truyền thống

Để tìm hiểu về Apache Spark [2], trước tiên ta nhìn lại về Hadoop, một framwork có truyền thống lâu đời trong việc phân tích và xử lý dữ liệu lớn. Ưu điểm lớn nhất của Hadoop được dựa trên một mô hình lập trình song song với xử lý dữ liệu lớn là MapReduce, mô hình này cho phép khả năng tính toán có thể mở rộng, linh hoạt, khả năng chịu lỗi và chi phí rẻ.



Hình 5: Hadoop và Spark

Dù có rất nhiều điểm mạnh về khả năng tính toán song song và khả năng chịu lỗi cao nhưng Apache Haddop có một nhược điểm là tất cả các thao tác đều phải thực hiện trên ổ đĩa cứng và điều này đã làm giảm tốc độ tính toán đi gấp nhiều lần.

Để khắc phục được nhược điểm này thì một hệ thống khác mang tên Apache Spark được ra đời bởi Matei Zaharia tại UC Berkeley's AMPLab vào năm 2009 và mở nguồn vào năm sau 2010. Apache Spark có thể chạy nhanh hơn 10 lần so với Hadoop ở trên đĩa cứng và 100 lần khi chạy trên bộ nhớ RAM.

4.1.2 Đặc điểm nổi bật của Spark

Sở dĩ Spark nhanh là vì nó xử lý mọi thứ ở RAM. Nhờ xử lý ở bộ nhớ nên Spark cung cấp các phân tích dữ liệu thời gian thực cho các chiến dịch quảng cáo, machine learning (học máy), hay các trang web mạng xã hội.

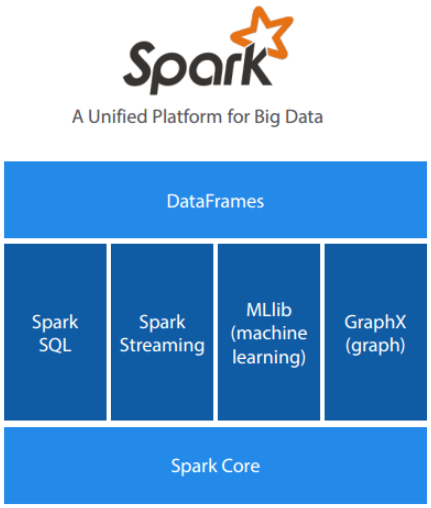
Spark nhận được nhiều sự hưởng ứng từ cộng đồng Big data trên thế giới do cung cấp khả năng tính toán nhanh và nhiều thư viện đi kèm hữu ích như Spark SQL (với kiểu dữ liệu DataFrames), Spark Streaming, MLlib (machine learning: classification, regression, clustering, collaborative filtering, và dimensionality reduction) và GraphX (biểu diễn đồ thị nhờ kết qủa tính toán song song).

- Xử lý dữ liệu: Spark xử lý dữ liệu theo lô và thời gian thực.
- Tính tương thích: Có thể tích hợp với tất cả các nguồn dữ liệu và định dạng tệp được hỗ trợ bởi cụm Hadoop.
- Hỗ trợ ngôn ngữ: hỗ trợ Java, Scala, Python và R.
- Xử lý thời gian thực: Apache Spark có thể xử lý các luồng sự kiện dữ liệu thời gian thực với tốc độ hàng triệu sự kiện mỗi giây.

Hiện nay, có rất nhiều hãng lớn đã dùng Spark cho các sản phẩm của mình như Yahoo, ebay, IBM, Cisco...

Hình 6: Những doanh nghiệp sử dụng Apache Spark

4.2 Sử dụng Spark bằng ngôn ngữ Python với PySpark



Hình 7: Các thành phần của PySpark [?]

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MlLib và GraphX, trong đó:

4.2.1 Spark Core

Là thành phần cốt lõi của spark, thành phần này hỗ trợ sử dụng spark trên một số ngôn ngữ như python, scala, java,...

4.2.2 Spark Streaming

Spark Streaming giúp cho kĩ sư và các nhà phân tích dữ liệu xử lý thông tin trong thời gian thực, khi mà các dữ liệu được truyền tải và cập nhật liên tục từ các nguồn như Kafka, Flume, và Amazon Kinesis,...

4.2.3 Spark SQL

Spark SQL là một module Spark dùng cho truy vấn cấu trúc dữ liệu. Module cung cấp nhiều hàm và phương thức hỗ trợ truy vấn dữ liệu dựa trên công cụ truy vấn SQL phân tán (distributed SQL query engine).

4.2.4 Spark DataFrame

Spark DataFrame hỗ trợ xây dựng dataframe từ tập dữ liệu nhưng với khả năng tối ưu hóa phong phú hơn.

4.2.5 MlLib

MlLib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Giúp thực hiện xây dựng model hiệu quả và đơn giản.

4.3 Code

Code được thực hiện theo các phần sau

4.3.1 Thêm thư viện và module

Cài đặt thư viện

```
!pip install pyspark
```

```
import pandas
import pyspark
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

4.3.2 Chia dữ liệu train và test

80% dữ liệu sẽ dùng để huấn luyện model và 20% còn lại để thử lại

```
df = pandas.read_csv(r'Final Dataset.csv').copy()
df_train = df[0: int(len(df) * 0.8)]
df_test = df[int(len(df) * 0.8) + 1:]
# print(df_train)
# print(df_test)
```

4.3.3 Xử lý dữ liệu

Để xây dựng cây quyết định (Desicion Tree) thì các giá trị phải ở dạng số. Nghĩa là đối với các ô giá trị kiểu chuỗi thì phải được hash về dạng số nào đó.

Mỗi giá trị kiểu chuỗi mới sẽ được chuyển đổi thành một số. Quy ước dãy số được chuyển đổi bắt đầu từ 0 và tăng dần thêm 1. Dữ liệu ở cột tên học sinh **ApplicantName** đang học không

```
defaultdict(<function <lambda> at 0x7f44a26f57a0>, {'Poor': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f20e0>, {'Medium': 0, 'Low': 1, 'High': 2})
defaultdict(<function <lambda> at 0x7f44a26f2ef0>, {'No': 0})
defaultdict(<function <lambda> at 0x7f44a26f2cb0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f2440>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f48c0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f4cb0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f4ef0>, {'Yes': 0, 'No': 1})
defaultdict(<function <lambda> at 0x7f44a26f4dd0>, {'Low': 0, 'High': 1, 'Medium': 2})
defaultdict(<function <lambda> at 0x7f44a26f4d40>, {'Medium': 0, 'Low': 1})
defaultdict(<function <lambda> at 0x7f44a26f4950>, {'Fail': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4e60>, {'Fail': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4680>, {'Fail': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4c20>, {'Poor': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4b00>, {'Poor': 0, 'Fair': 1, 'Good': 2, 'Excellent': 3, 'Very Good': 4, 'Adequate': 5})
defaultdict(<function <lambda> at 0x7f44a26f4b90>, {0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 8: 7})
defaultdict(<function <lambda> at 0x7f44a26f4a70>, {0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7, 13: 8})
```

Hình 8: Xử lý dữ liệu kiểu chuỗi

mang giá trị dự đoán ở cuối kết quả khóa học, nên trước khi đưa vào train model, ta phải bỏ qua cột giá trị này.

4.3.4 Xây dựng cây quyết định

```
X = df_train[features[:-1]] #remove Results
y = df_train[features[-1]]
dtree = DecisionTreeClassifier()
dtree = dtree.fit(X.values, y.values)
```

4.3.5 Thử dự đoán một số dữ liệu từ tập dữ liệu test

Dữ liệu dự đoán một số học sinh sau: Nhận xét thấy kết quả dự đoán tương đối giống với kết quả hiện có của tập dữ liệu **test**. Kết quả model dự đoán:

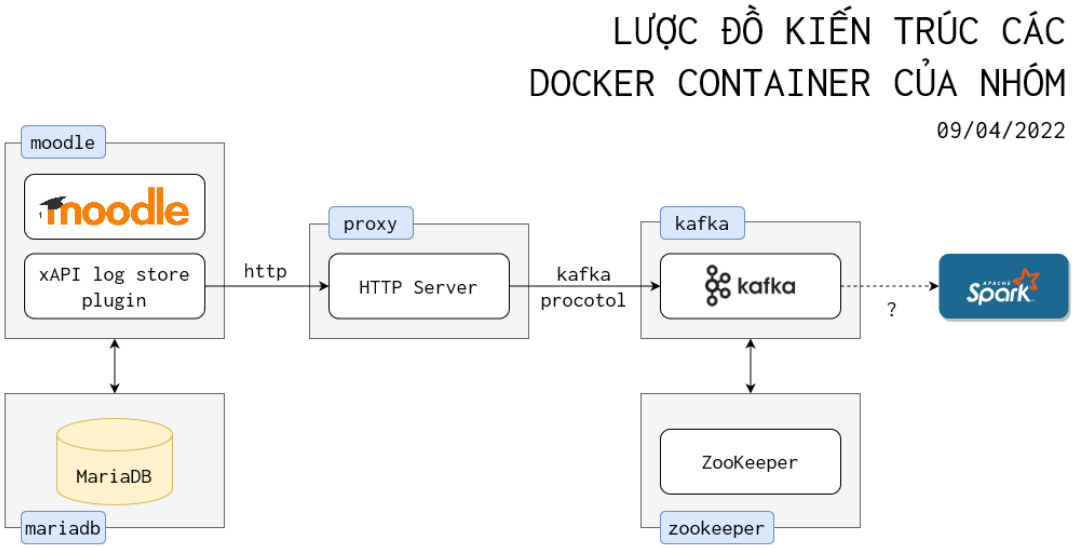
```
Frequency = 92.3076923076923%
```

```
Prediction Pass | Actually Pass
Student 305
Prediction Pass | Actually Pass
Student 306
Prediction Fail | Actually Pass
Student 307
Prediction Fail | Actually Fail
Student 308
Prediction Pass | Actually Pass
Student 309
Prediction Pass | Actually Pass
Student 310
Prediction Pass | Actually Pass
Student 311
Prediction Fail | Actually Fail
Student 312
Prediction Pass | Actually Pass
Student 313
Prediction Fail | Actually Fail
Student 314
Prediction Fail | Actually Fail
Student 315
Prediction Fail | Actually Pass
Student 316
Prediction Fail | Actually Pass
Student 317
Prediction Fail | Actually Fail
```

Hình 9: Kết quả dự đoán một số kết quả học tập của học sinh

5 Kiến trúc của hệ thống

Hình 10 mô tả kiến trúc mà nhóm nghiên cứu đề xuất cho đề tài này.



Hình 10: Kiến trúc của hệ thống

Đầu tiên, ở các hệ thống LMS, dữ liệu về hành vi của học viên sẽ được thu thập và gửi về Kafka. Dữ liệu này sẽ được gửi liên tục thành dòng. Ở đây, nhóm nghiên cứu chọn Moodle LMS để có thể dễ dàng mở rộng bằng các plugin.

Plugin được chọn ở đây là Moodle Logstore xAPI [4]. Dữ liệu được gửi vào một endpoint qua HTTP nên nhóm thiết lập một container `proxy` để một đầu nhận HTTP và đầu còn lại kết nối với Kafka.

Sau đó, nhóm sẽ sử dụng Spark để phân tích các dữ liệu. Hiện tại phần này vẫn chưa được hoàn thiện.

Tài liệu tham khảo

[1] Apache kafka. https://kafka.apache.org/.

[2] Apache spark™ - unified engine for large-scale data analytics. https://spark.apache.org/.

[3] Moodle - open-source learning platform. https://moodle.org/.

[4] Moodle logstore xapi. https://github.com/xAPI-vle/moodle-logstore_xapi, 2020.