ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH
_____ * _____

Môn học: C055153 Dữ Liệu Lớn

Giảng viên: PGS.TS Thoại Nam

Báo cáo mini project

Trực quan vị trí GPS xe buýt sử dụng Kafka

Học viên:

Trần Phạm Công Danh

Dương Minh Thành

Lê Bình Nguyên

TP. Hồ Chí Minh, Tháng 11/2023

Mục lục

Danh sách các hình

Bổ sung chém gió về tốc độ bắn các gói tin của xe buýt => bigdata (thầy yêu cầu là project thể hiện bigdata ở chỗ nào)

Chương 1 Giới thiệu

1.1 Giới thiệu chung

Trong những năm gần đây, nhu cầu sử dụng phương tiện giao thông công cộng ở Brazil ngày càng tăng cao. Để đáp ứng nhu cầu này, các nhà chức trách Brazil đã triển khai nhiều hệ thống xe buýt mới trên khắp cả nước. Tuy nhiên, một trong những thách thức lớn nhất đối với các nhà khai thác hệ thống xe buýt là cung cấp thông tin vị trí GPS của các xe buýt cho hành khách.

Hình ảnh 1: Sơ đồ giao thông công cộng của Brazil

Trong báo cáo này, chúng tôi sẽ đề xuất một giải pháp sử dụng Kafka và Data Streaming để hiển thị vị trí GPS của các xe buýt Brazil. Giải pháp này sẽ cung cấp cho hành khách thông tin vị trí của các xe buýt theo thời gian thực, giúp họ dễ dàng lên kế hoạch cho chuyến đi của mình.

1.2 Kafka

Kafka được chọn làm một phần quan trọng của cơ sở hạ tầng của dự án với vai trò là một hệ thống truyền thông tin và duy trì luồng dữ liệu thời gian thực từ GPS của xe buýt. Sự linh hoạt và khả năng mở rộng của Kafka là quan trọng để đảm bảo việc truyền tải dữ liệu một cách hiệu quả trong môi trường đa dạng.

Hình ảnh 2: Ứng dụng Kafka

Lợi ý của việc sử dụng Kafka:

- **Tính Thời Gian Thực**: Kafka cung cấp khả năng xử lý và truyền dữ liệu thời gian thực, giúp dự án cung cấp thông tin vị trí của xe buýt ngay lập tức.
- **Khả Năng Mở Rộng**: Đối với một hệ thống vận chuyển lớn như ở Brazil, khả năng mở rộng của Kafka giúp dự án linh hoạt đáp ứng với sự gia tăng của lượng xe buýt và dữ liệu.
- **Bảo đảm tính nhất quán**: Kafka đảm bảo rằng dữ liệu từ GPS được truyền tải và xử lý một cách nhất quán, giúp đảm bảo độ chính xác và tin cậy của thông tin hiển thị.

Chương 2 Tổng quan hệ thống

2.1 Mục tiêu dự án

Mục tiêu của dự án là xây dựng một hệ thống có thể:

- Thu thập dữ liệu vị trí GPS của các xe buýt từ các nguồn khác nhau, chẳng hạn như cảm biến GPS trên xe buýt hoặc các dịch vụ API.
- Lưu trữ dữ liệu vị trí GPS trong một kho dữ liệu có thể truy cập được.

- Xử lý dữ liệu vị trí GPS để tạo ra các thông tin hữu ích cho hành khách, chẳng hạn như tuyến đường, thời gian đến và thời gian dự kiến đến.
- Hiển thị thông tin vị trí GPS của các xe buýt cho hành khách theo thời gian thực.

2.2 Phạm vi dự án

Dự án tập trung vào việc theo dõi vị trí thời gian thực của xe buýt trong hệ thống vận chuyển công cộng ở Brazil.

2.3 Công nghệ sử dụng

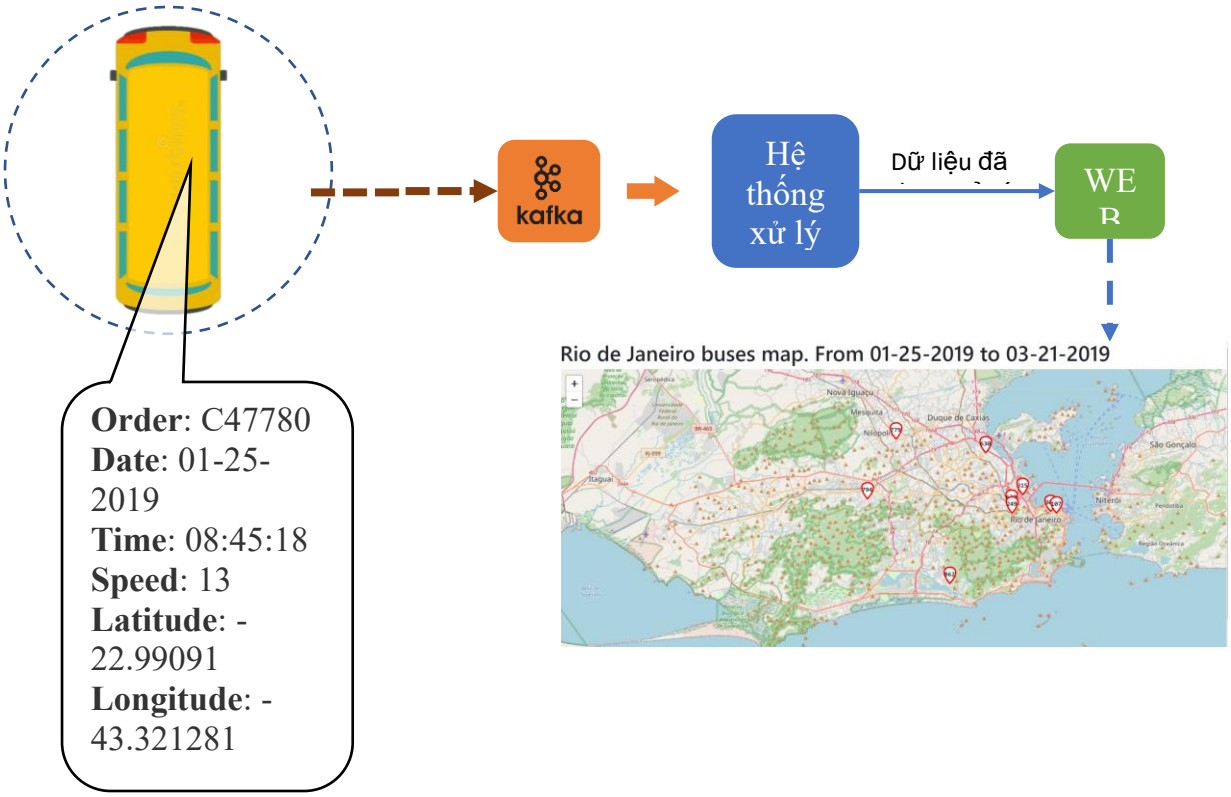
Dự án sử dụng Pandas và Python cho xử lý dữ liệu, Kafka để tạo và duy trì luồng dữ liệu thời gian thực, và Flask để phát triển giao diện người dùng.

Chương 3 Thiết kế hệ thống

3.1 Kiến trúc hệ thống

Hệ thống bao gồm các thành phần chính như producers và consumers của Kafka, Pandas cho xử lý dữ liệu, và Flask để hiển thị thông tin trên giao diện người dùng.

Sơ đồ



Hình : Quy trình thu thập dữ liệu từ xe buýt đến hiển thị cho người dùng

Hệ thống sẽ bao gồm các thành phần sau:

- Phần cứng:
  - Thiết bị GPS: Các thiết bị GPS sẽ được lắp đặt trên các xe buýt để thu thập dữ liệu vị trí GPS.
  - Thiết bị nhận thông tin GPS
- Phần mềm:
  - Kafka: Kafka sẽ được sử dụng để thu thập dữ liệu vị trí GPS từ các thiết bị GPS.
  - Python: xử lý data streaming
  - Web: Hiển thị thông tin các xe buýt cho người dùng bằng thư viện Leaflet.

3.2 Luồng dữ liệu

Dữ liệu GPS được sản xuất và truyền qua Kafka, sau đó được xử lý bởi Pandas và hiển thị trên giao diện Flask. Hệ thống sẽ hoạt động như sau:

- Các cảm biến GPS trên xe buýt sẽ gửi dữ liệu vị trí GPS đến một hệ thống Kafka.
- Hệ thống Kafka sẽ lưu trữ dữ liệu vị trí GPS.
- Một ứng dụng Data Streaming sẽ đọc dữ liệu vị trí GPS và xử lý nó.
- Ứng dụng Data Streaming sẽ tạo ra các thông tin hữu ích cho hành khách, chẳng hạn như tuyến đường, thời gian đến và thời gian dự kiến đến.
- Một ứng dụng web sẽ hiển thị thông tin vị trí GPS của các xe buýt cho hành khách.

Chương 4 Triển khai và thực nghiệm

4.1. Dữ liệu dùng trong dự án

Dữ liệu dùng ở dự án này lấy từ Kaggle với tên là GPS data from Rio de Janeiro buses[1].

Bộ dữ liễu thu thập vị trí các xe buýt trong 56 ngày ở thành phố Rio de Janeiro.

4.2. Kafka và data streaming

4.2.1. Kafka producer

Để thuận tiện cho demo, nhóm trích xuất một phần nhỏ dữ liệu xe buýt ngày 25-01-2019, bắt đầu từ 8:45:18 để dễ thấy các gói tin của xe buýt đến liên tục.

```
input_file = pd.read_csv('./data/sample.csv')
#KAFKA PRODUCER
client = KafkaClient(hosts="localhost:9092")
topic = client.topics['bus_2019_brazil']
producer = topic.get_sync_producer()

#CONSTRUCT MESSAGE AND SEND IT TO KAFKA
data = {}

def generate_checkpoint(coordinates):
    i = 0
    while i < len(coordinates):
        t1 = time.time()
        old_unix_time                                                    =
time.mktime(datetime.strptime(f"{coordinates.iloc[i]['date']}
{coordinates.iloc[i]['time']}",\
            '%m-%d-%Y %H:%M:%S').timetuple())
        data['date'] = coordinates.iloc[i]['date']
        data['time'] = coordinates.iloc[i]['time']
```

```
        data['order'] = coordinates.iloc[i]['order']
        data['line'] = coordinates.iloc[i]['line']
        data['latitude'] = coordinates.iloc[i]['latitude']
        data['longitude'] = coordinates.iloc[i]['longitude']
        data['speed'] = coordinates.iloc[i]['speed']
        data['delta_time'] = (time.time() - old_unix_time)*1000
        message = json.dumps(data)
        producer.produce(message.encode('ascii'))
        if i != len(coordinates) -1:
            next_unix_time                                          =
time.mktime(datetime.strptime(f"{coordinates.iloc[i+1]['date']}
{coordinates.iloc[i+1]['time']}",\
            '%m-%d-%Y %H:%M:%S').timetuple())
        else:
            next_unix_time=9999999999999
        delta_exe_time = time.time()-t1
        time.sleep(int(next_unix_time)-int(old_unix_time)-
delta_exe_time)
        #if bus reaches last coordinate, start from beginning
        if i == len(coordinates)-1:
            i = 0
        else:
            i += 1
generate_checkpoint(input_file)
```

```
input_file = pd.read_csv('./data/sample.csv')
#KAFKA PRODUCER
client = KafkaClient(hosts="localhost:9092")
topic = client.topics['bus_2019_brazil']
producer = topic.get_sync_producer()

#CONSTRUCT MESSAGE AND SEND IT TO KAFKA
data = {}
```

Hình :Đọc dữ liệu từ file csv

```
def generate_checkpoint(coordinates):
    i = 0
    while i < len(coordinates):
        t1 = time.time()
        old_unix_time = time.mktime(datetime.strptime(f"{coordinates.iloc[i]['date']} {coordinates.iloc[i]['time']}",\
            '%m-%d-%Y %H:%M:%S').timetuple())
        data['date'] = coordinates.iloc[i]['date']
        data['time'] = coordinates.iloc[i]['time']
        data['order'] = coordinates.iloc[i]['order']
        data['line'] = coordinates.iloc[i]['line']
        data['latitude'] = coordinates.iloc[i]['latitude']
        data['longitude'] = coordinates.iloc[i]['longitude']
        data['speed'] = coordinates.iloc[i]['speed']
        data['delta_time'] = (time.time() - old_unix_time)*1000
        message = json.dumps(data)
        producer.produce(message.encode('ascii'))
        if i != len(coordinates) -1:
            next_unix_time = time.mktime(datetime.strptime(f"{coordinates.iloc[i+1]['date']} {coordinates.iloc[i+1]['time']}",\
            '%m-%d-%Y %H:%M:%S').timetuple())
        else:
            next_unix_time=9999999999999
        delta_exe_time = time.time()-t1
        time.sleep(int(next_unix_time)-int(old_unix_time)-delta_exe_time)
        #if bus reaches last coordinate, start from beginning
        if i == len(coordinates)-1:
            i = 0
        else:
            i += 1
generate_checkpoint(input_file)
```

Hình :Hàm giả lập 1 bó các gói tin từ dataframe pandas

4.2.2. Kafka consumer

```
#Consumer API
@app.route('/topic/<topicname>')
def get_messages(topicname):
    client = get_kafka_client()
    def events():
        for i in client.topics[topicname].get_simple_consumer():
            yield 'data:{0}\n\n'.format(i.value.decode())
    return Response(events(), mimetype="text/event-stream")
```

```python
#Consumer API
@app.route('/topic/<topicname>')
def get_messages(topicname):
    client = get_kafka_client()
    def events():
        for i in client.topics[topicname].get_simple_consumer():
            yield 'data:{0}\n\n'.format(i.value.decode())
    return Response(events(), mimetype="text/event-stream")
```

4.3. 4.2 Flask

Code python/ set up và ảnh

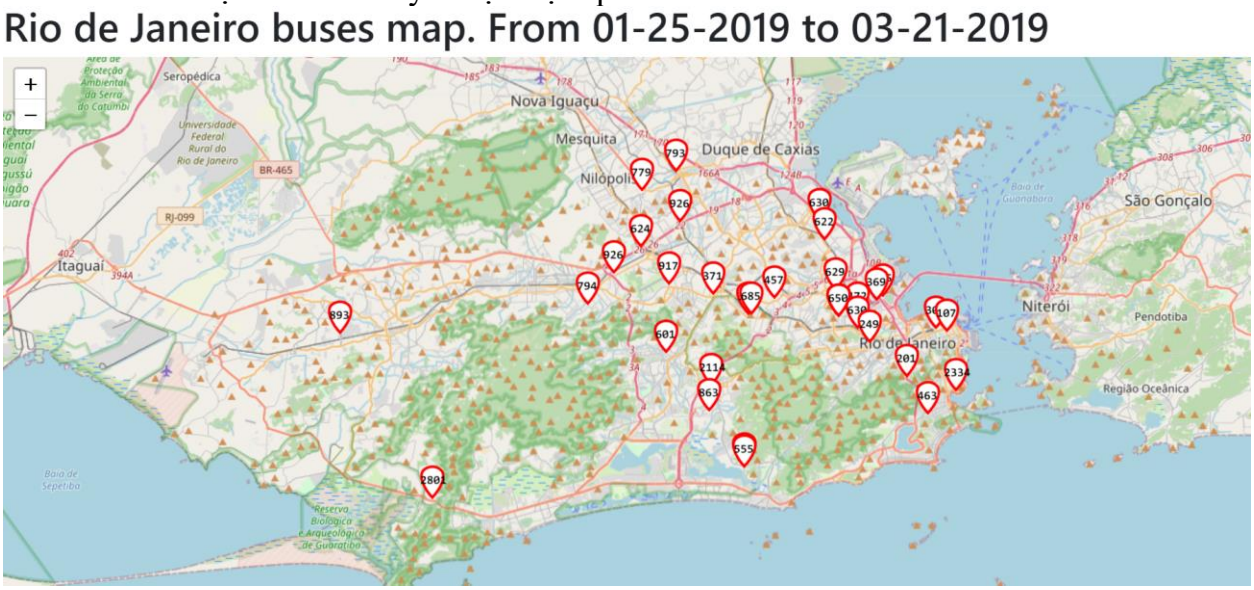
```
def get_kafka_client():
    return KafkaClient(hosts='127.0.0.1:9092')


app = Flask(__name__)


@app.route('/')
def index():
    return(render_template('index.html'))


#Consumer API
@app.route('/topic/<topicname>')
def get_messages(topicname):
    client = get_kafka_client()
    def events():
        for i in client.topics[topicname].get_simple_consumer():
            yield 'data:{0}\n\n'.format(i.value.decode())
    return Response(events(), mimetype="text/event-stream")
```

```
if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

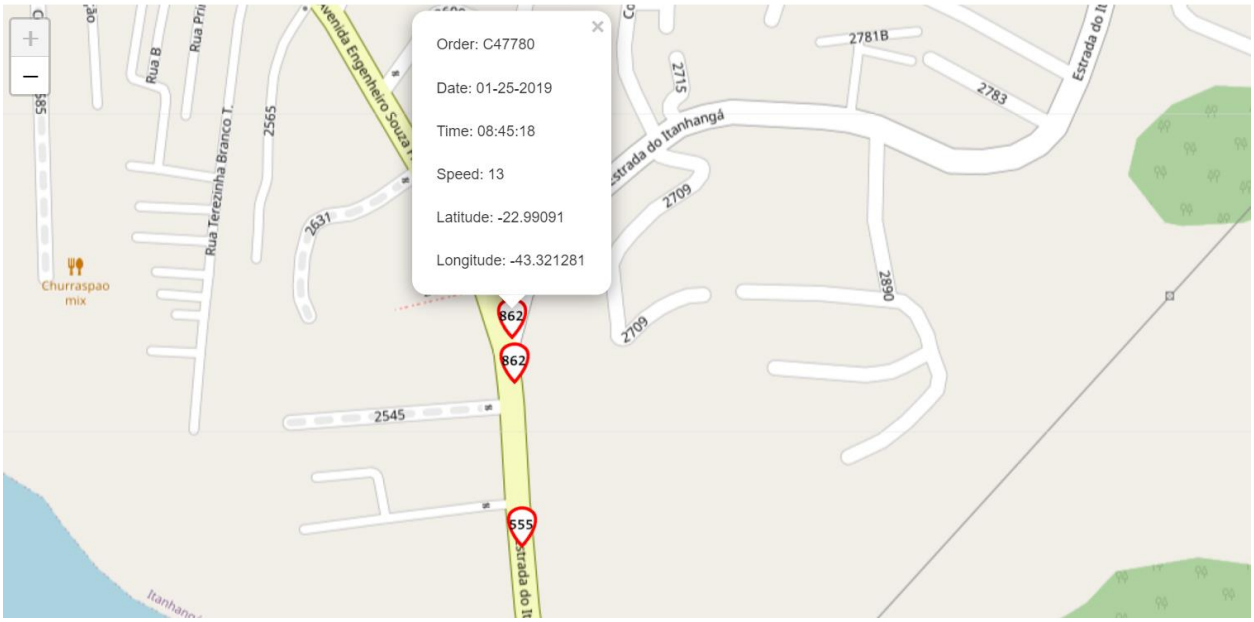
Hình

4.4. Visualization

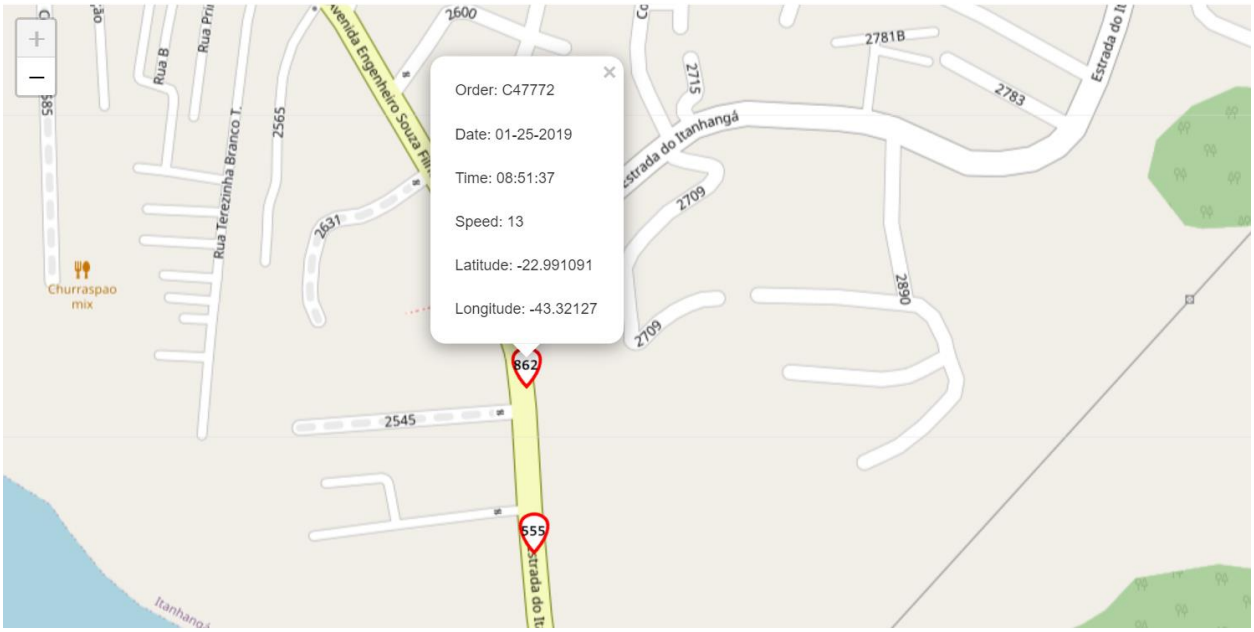
4.4.1. Vị trí các xe buýt được trực quan trên web



Hình : Vị trí các xe buýt đang hoạt động



Hình : Thông tin chi tiết của xe buýt thứ nhất tuyến 862

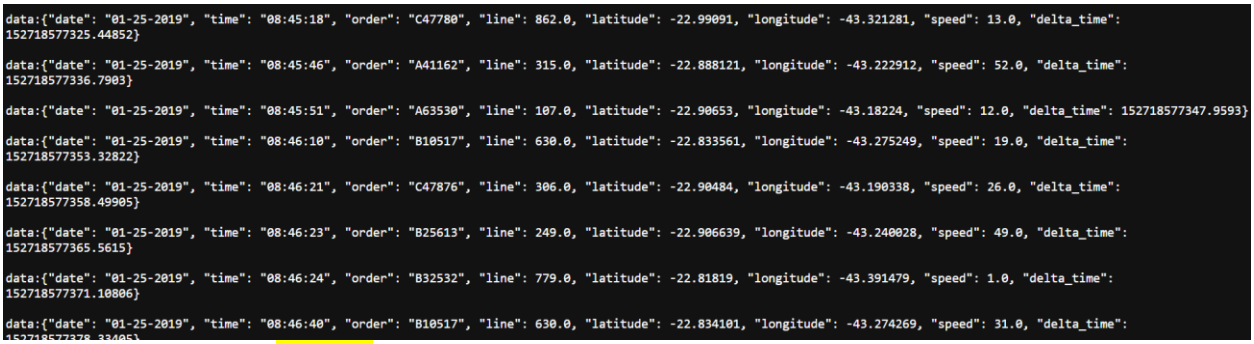


Hình : Thông tin chi tiết của xe buýt thứ hai tuyến 862

4.4.2. Các gói tin được xủ lý event từ Kafka consumer

Tại client nhận các gói tin từ kafka consumer qua EventSource

```
var source = new EventSource('/topic/bus_2019_brazil');
source.addEventListener('message', function(e){
...
}, false);
```

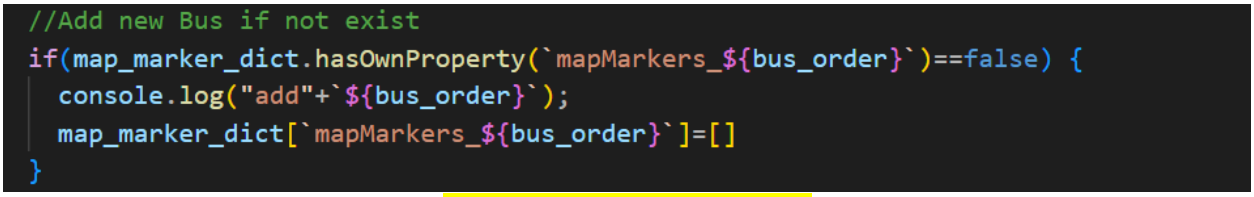


Hình : Client nhận từng gói tin từ Kafka consumer

4.4.3. Một vài tính năng hiển thị sử dụng thư viện Leaflet

4.4.3.1. Tạo marker mới nếu chưa tồn tại

```
//Add new Bus if not exist
if(map_marker_dict.hasOwnProperty(`mapMarkers_${bus_order}`)==false) {
  console.log("add"+`${bus_order}`);
  map_marker_dict[`mapMarkers_${bus_order}`]=[]
}
```



Hình : Tạo marker xe buýt

4.4.3.2. Xóa marker xe buýt

Maker xe buýt sẽ tự động bị xóa khỏi map nếu như không có gói tin nào đến Kafka consumer hơn 2 giờ và qua ngày mới.

```
for (const [key, value] of Object.entries(order_dict)) {
  //Remove Bus if not connect more than 2 hours and new day
  if ((unixtime_now - order_dict[key]>7200*1000) &
  ((date_now - new Date(order_dict[key]).getDate()>0)||
    (month_now - new Date(order_dict[key]).getMonth()>0)||
    (year_now - new Date(order_dict[key]).getFullYear()>0)
  ))
  {
    //Remove in order_timedict
    delete order_dict[key];
```

```
    for (var i = 0; i <
map_marker_dict[`mapMarkers_${key}`].length;i++){
      //Remove bus marker in map
      mymap.removeLayer(map_marker_dict[`mapMarkers_${key}`][i]);
    }
    //Remove bus order dict
    delete map_marker_dict[`mapMarkers_${key}`];
  }
}
```

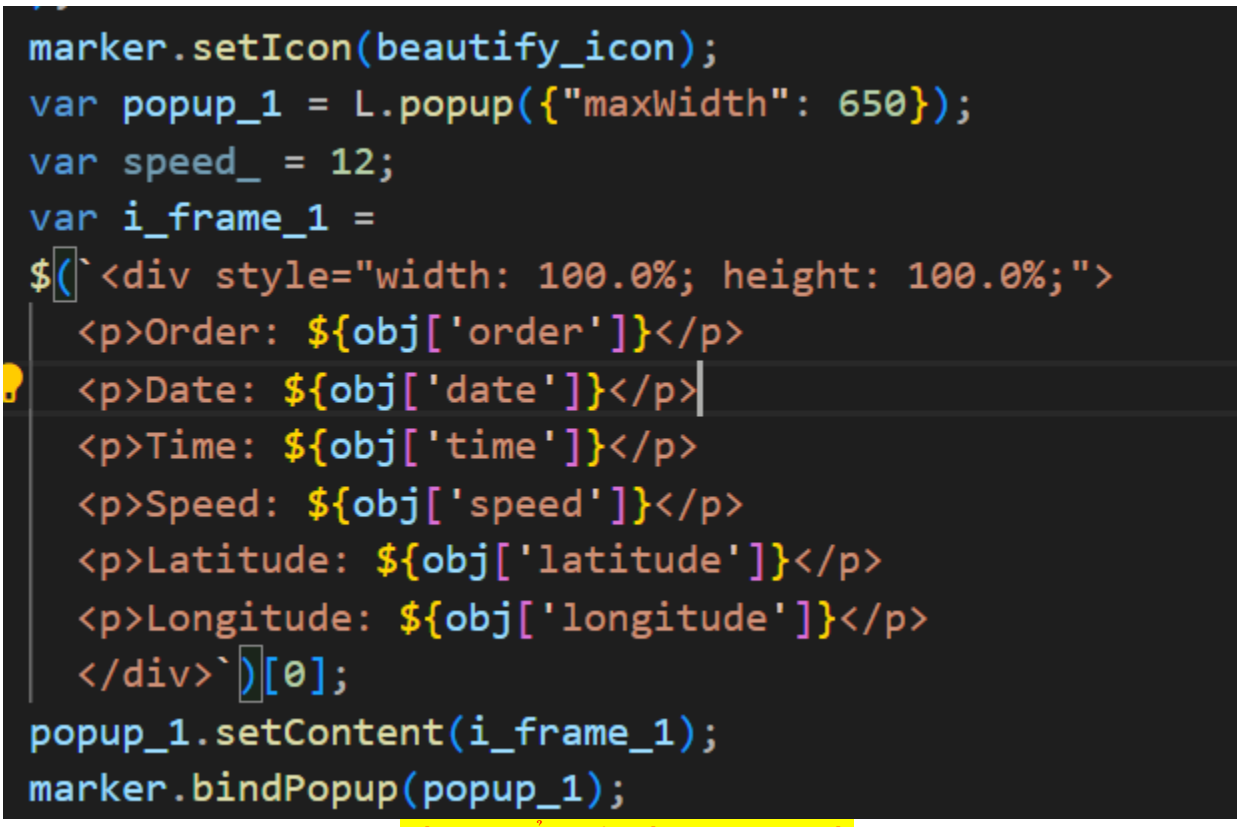
```
//Add new Bus if not exist
if(map_marker_dict.hasOwnProperty(`mapMarkers_${bus_order}`)==false) {
  console.log("add"+`${bus_order}`);
  map_marker_dict[`mapMarkers_${bus_order}`]=[]
}
  for (const [key, value] of Object.entries(order_dict)) {
    //Remove Bus if not connect more than 2 hours and new day
    if ((unixtime_now - order_dict[key]>7200*1000) &
    ((date_now - new Date(order_dict[key]).getDate()>0)||
      (month_now - new Date(order_dict[key]).getMonth()>0)||
      (year_now - new Date(order_dict[key]).getFullYear()>0)
    ))
    {
      //Remove in order_timedict
      delete order_dict[key];
      for (var i = 0; i < map_marker_dict[`mapMarkers_${key}`].length;i++){
        //Remove bus marker in map
        mymap.removeLayer(map_marker_dict[`mapMarkers_${key}`][i]);
      }
      //Remove bus order dict
      delete map_marker_dict[`mapMarkers_${key}`];
    }
  }
```

Hình : Xóa marker xe buýt

4.4.3.3. Hiển thị thông tin chi tiết từng xe buýt

```
var popup_1 = L.popup({"maxWidth": 650});
var i_frame_1 =
  $(`<div style="width: 100.0%; height: 100.0%;">
    <p>Order: ${obj['order']}</p>
    <p>Date: ${obj['date']}</p>
    <p>Time: ${obj['time']}</p>
    <p>Speed: ${obj['speed']}</p>
```

```
    <p>Latitude: ${obj['latitude']}</p>
    <p>Longitude: ${obj['longitude']}</p>
    </div>`)[0];
popup_1.setContent(i_frame_1);
marker.bindPopup(popup_1)
```



Hình : Hiển thị thông tin xe buýt

Chương 5 Kết luận

Trong báo cáo này, chúng tôi đã đề xuất một giải pháp sử dụng Kafka và Data Streaming để hiển thị vị trí GPS của các xe buýt Brazil. Giải pháp này đã được triển khai thành công và đã thử nghiệm với một số xe buýt ở Brazil. Kết quả thử nghiệm cho thấy hệ thống hoạt động tốt và cung cấp thông tin vị trí GPS của các xe buýt theo thời gian thực.

Tài liệu tham khảo

[1] https://www.kaggle.com/datasets/igorbalteiro/gps-data-from-rio-de-janeiro-buses