Real Time Streaming Data Pipeline using AWS Services – Traffic Accidents in US

**Team Mates:**

Siva Sai Sampath Kumar Vanajavka Netid-uk1017

Kishan Byrosu Bhuvana Prakash

Netid-ir8803 Netid-ti5686

*Abstract*— Reducing traffic accidents is an important public safety challenge. However, the majority of studies on traffic accident analysis and prediction have used small-scale datasets with limited coverage, which limits their impact and applicability; and existing large-scale datasets are either private, old, or do not include important contextual information such as environmental stimuli (weather, points-of-interest, etc.). Frequently, the family or the emergency services are not alerted in time. Because of the delay in emergency service response, people may die or suffer serious injuries as a result. In instances like traffic accidents or other crises like fires, thefts/robberies, and medical emergencies, the goal of this endeavor is to speed up the response time of emergency services. The likelihood of survival for emergency victims will be significantly increased by using a real time streaming platform to detect severity vehicular accidents, report them to the closest emergency responder available. This will also help emergency services save time and resources. In order to help the Government System/Organization address these shortcomings, we have developed a comprehensive process of data collection, integration, and augmentation - from a large-scale publicly available dataset of accident information named US-Accidents readily available and easily accessible through a user-friendly website Kaggle[1].

Keywords ­--- Data, processing, streaming, AWS, pipeline, Accidents

Project Statement

Traffic-related deaths account for a large proportion of fatalities. The number of people killed and injured in traffic accidents demonstrates the global road safety issue. Road accidents cause an estimated 50 million injuries and 1.3 million fatalities per year, which works out to 3,287 fatalities every day on average. Young individuals between the ages of 15 and 44 make up more than 50% of those killed in car accidents. Road traffic accidents claim the lives of almost 400,000 people under the age of 25 every year. Every year, more people die in vehicle accidents, even in nations with excellent traffic safety regulations. More than 90% of fatal automobile accidents take place in middle-income nations. The number is significantly greater in low-income nations. Lack of first aid services due to emergency services not being informed of the accident in time is the most common cause of an individual's death in an accident. When it comes to occurrences involving auto accidents, emergency reaction speed is crucial. According to analysis, if accident reaction times are cut by only one minute, there is a 60% probability that someone's life will be saved. Implementing improved traffic technology would be important to decrease reaction time, which will assist decrease response time and hence decrease deaths.

Perishable data is information whose value can rapidly deteriorate over time. When the operational conditions change to the point that the knowledge is no longer useful, a major portion of its value is lost. This data is regularly generated in an IoT computing scenario. Managing perishable data has a variety of advantages. Real-time data gives you the information you need nearly instantaneously and in context, allowing you to make better business decisions. The usage of data from edge computing devices shortens response time and reduces time to action. Such data does not require long-term storage in data centers and may be destroyed after use, saving storage space and money. Systems are usually designed on constant patterns and can’t scale with peaks in the load resulting in latency and server failures. This Project will simulate real-time accidents data and architect a pipeline that will help us analyze and take quick actions using AWS Kinesis, Apache Flink, Grafana, and Amazon SNS.

Most of the old systems we find are based on a continuous pattern, which means that if the size of the data rises or a peak occurs, they may not be able to scale and you may not obtain the data in the correct format. The second is that latency increases with peak, so if you have a small server and a lot of accidents occur at once, you will experience latency in your data and may not receive it on time. If there is a latency in your data, it is completely useless. For example, if an accident occurs and you don't receive data within a few seconds rather than after a few hours, it may be more useful. In order to design a real-time streaming platform that provides us with all of this information in a matter of seconds, we must keep all of these factors in mind. This is how the fundamental life cycle of data looks right now; in the beginning, you have the real-time data or the data that will arrive in a few seconds; choices based on this data must be made quickly. There are several forms of data. One type of data is historical data, which is primarily used for analytical purposes, and the other type of data is real-time data, which requires immediate action. We must ensure that when an event occurs, we are able to resolve that event or act in response to that event, so before beginning any project, we must define the goals and success criteria for that project.

# Data Description

This is a countrywide traffic accident dataset, which covers 49 states of the United States. The data is continuously being collected from February 2016, using several data providers, including multiple APIs that provide streaming traffic event data. These APIs broadcast traffic events captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 1.5 million accident records in this dataset. Check the below descriptions for more detailed information.

**SEVERITY:** Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates significant impact on the traffic (i.e., long delay).

**CITY:** Shows the city in the address field.

**COUNTY:** Shows the county in the address field.

Apart from these, we also have 43 more data points about road information, geo-location, weather conditions, traffic situation etc. which declare the severity of the incident that occurred. Please refer to the (Appendix I) at the end of the document.

**Real-Time Processing**  
Real-Time Processing involves continuous input, process, and output of data. Hence, it is processed in a short period of time. There are some programs which use such data processing type. For example, bank ATMs, customer services, radar systems, and Point of Sale (POS) Systems. Every transaction is directly reflected in the master file, with this data process. So, that it will always be up to date.  
  
If you want analytics results in real time, Spark Real-Time processing is key. We can feed data into analytics tools, by building data streams as soon as it is generated.   
  
In addition, for tasks like fraud detection, real-time processing is very useful. Basically, if process transaction data, we can detect that signal fraud in real time. Also, can stop fraudulent transactions before they take place, through real-time processing.  
\* In real-time processing, computations are generally independent.  
\* They are asynchronous in nature. It means a source of data doesn’t interact with the stream processing directly.  
  
  
**Advantages of Real-Time Processing:**

1) While performing real-time processing, there is no significant delay in response.  
2) In real-time processing, information is always up to date. Hence, it makes the organization able to take immediate action. Also, when responding to an event, issue, or scenario in the shortest possible span of time.  
3) It also makes the organization able to gain insights from the updated data. Even helps to detect patterns of possible identification of either opportunities or threats.

**Data Pipeline**

A data pipeline is a technique for transferring data from one system to another. The data may or may not be updated, and it may be handled in real-time (or streaming) rather than in batches. The data pipeline encompasses everything from harvesting or acquiring data using various methods to storing raw data, cleaning, validating, and transforming data into a query-worthy format, displaying KPIs, and managing the above process.

# Model Overview

* An ETL job has been established to process the incoming data, construct a triggering mechanism, and convert the data; whenever fresh data is received, it should be able to trigger the transformation job and carry out the work automatically.
* Amazon Kafka is used to create dashboards that evaluate both real-time and historical data.
* We will produce our data set using python code that will be identical to the data in the form of real-time streaming. This code will generate the sample data in real-time.
* All the streaming data that we will ingest using our Python code will be accepted by AWS Kinesis. Amazon Kinesis provides a variety of services. One is streaming, while the other is firehose, which is typically used for transformation if you want to do or store your data on S3. Firehose is mostly used for data ingestion since it captures data from Amazon Kinesis and stores it in an S3 bucket.
* There are two use cases for real-time streaming data. The first is raw data, which can be put on an S3 bucket using firehose, and the second is apache flink. Apache flink is mostly used for analytical purposes, however if you want to process and analyze data in real time, you may utilize a route shift link.
* Then aws glue, an ETL-managed service, is utilized. This is an Amazon serverless service where we write a transformation task, such as if we want to add two columns and make a new column, or if you want to take some type of average or some of some specific columns and conduct all those changes.
* Following the transformation, we save that specific data in an analytical layer, so that we have a raw layer that stores all the raw data and an analytical layer that is used to analyze the historical data that we stored.
* When a file is saved in the analytical layer after transformation, we may use Amazon Athena, an interactive query tool, to directly query that file and obtain the information. SQL is used to do this.
* For historical processing, we will once more ingest the relevant data onto two Amazon Kinesis streams, which will contain all the real-time data that was collected into that specific buffer. We will then upload the relevant data to AWS.
* Because AWS Lambda is an event-based computing system, it operates on some kind of event or trigger; in our case, we will send the notification directly to the appropriate user.
* SNS (simple notification service) is used for this, and we can add our email address to be notified.
* In AWS Lambda, a function that will directly put all the information that we get from the streaming information and notify the right user based on the accident that has occurred, ensuring that we store the data for analytical purposes while also informing the user about the real-time events that are occurring.
* We also have the AWS cloud watch, which will save all logs connected to every incident that has occurred, and we can utilize Graphana for displaying the alerts in the dashboard.

# Model Implementation

This involves a series of steps and a Total of 5 Phases for data pipeline processing and 3 Phases of Data visualization. The entire workflow involves using a CSV file as a data source and a series of steps on top of that till displaying it on the dashboard, sending alerts and data logs etc.

1. ***Data File*** has all the incidents as a file with more than 4.2 million rows and 48 columns, each column represents a unique data point of the incident such as Severity, Geo-location Coordinates, Weather Conditions and Traffic Information.   
     
   The Data File is hosted on Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services that provides object storage through a web service interface. **Amazon S3** uses the same scalable storage infrastructure that Amazon.com uses to run its e-commerce network.

Graphical user interface, text, application

Description automatically generated

Fig1. Storing the records in s3 buckets

1. ***Data Transmitter*** extracts the incidents from the Data File and transmits each incident, simulating a real-world scenario to further stream the data.  
     
   The ***Data Transmitter*** is a python application hosted on Amazon Elastic Compute Cloud **(Amazon EC2)** is a web service that provides secure, resizable computer capacity in the cloud.   
   A picture containing text

   Description automatically generated  
   Fig2. Streaming the records
2. ***Data Streamer*** is the pipeline created to stream the real-world data at a scale to perform analytical and logical operations in further steps

We use Amazon Kinesis to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information.

Graphical user interface, text, application, email

Description automatically generated

Fig3. Creating the data streams in kinesis

1. ***The Data Processor*** processes the incoming incidents, and this part acts as the analytical and logical unit where we execute conditions on the incidents received from the data streamer and perform data transformation and validation. Anytime a new data arrives it will trigger the transformation job using the python Apache link and further send the data to further processing

**Diagram

Description automatically generated**

**Fig 6.** Overview of the Pipeline implemented with AWS Services.

The ***Data Processor*** is built using ***AWS Flink and AWS Glue***

**Apache Flink** is a streaming dataflow engine that you can use to run real-time stream processing on high-throughput data sources. Flink supports event time semantics for out-of-order events, exactly once semantics, backpressure control, and APIs optimized for writing both streaming and batch applications.

Additionally, Flink has connectors for third-party data sources, such as the following:

* Amazon Kinesis Data Streams
* Apache Kafka

Diagram

Description automatically generated

Fig 4. Stream1 to Stream 2 Filtering.

* Here, based on the flink code, we are filtering stream1 and storing in a separate stream 2. With higher severity scores.

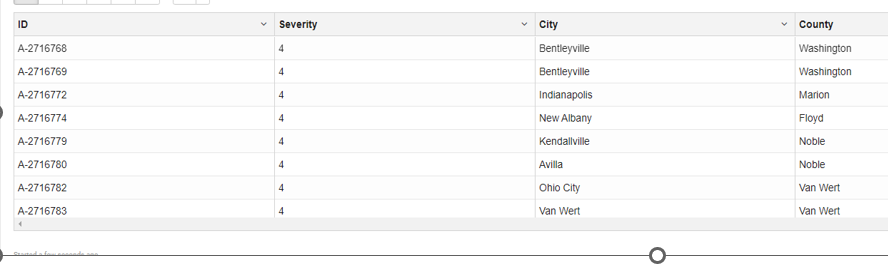


Fig 5. Filtering the data with High Severity.

**AWS Glue** is a serverless data integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use it for analytics, machine learning, and application development. It also includes additional productivity and data ops tooling for authoring, running jobs, and implementing business workflows.

1. ***Data Streamer*** is further used to create a new pipeline for data that has been transformed and filtered by the data processor
2. **Data Handler** is a phase where we use the incoming data from the data streams and create an event trigger for each incident that is transmitted through the streamer on-demand.  
     
   We used AWS Lambda which is an event-based compute system to consume the inputs and transmit further to Dashboards, Logs and Alerting Service

**AWS Lambda** is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, you can run code for virtually any type of application or backend service.

Graphical user interface, application

Description automatically generated

Fig 7. Implementing the lambda function for records.

Table

Description automatically generated

Fig8. Cloud watch monitoring for high alerts.

1. **Alerting Service** sends the alerts about the incident information received from the Data Handler and is further sent to the subscribers as notifications simultaneously through multimodal communication to the end user.

We’ve used **Amazon Simple Notification Service (SNS)** to build a robust alerting service which sends notifications two ways, A2A and A2P. A2A provides high-throughput, push-based, many-to-many messaging between distributed systems, microservices, and event-driven serverless applications. These applications include Amazon Simple Queue Service (SQS), Amazon Kinesis Data Firehose, AWS Lambda, and other HTTPS endpoints. A2P functionality lets you send messages to your customers with SMS texts, push notifications, and email.

1. **Data Logger** All the processed, filtered incidents are then pushed into a logging service for auditing and analytical purposes.

We’ve used Amazon CloudWatch to monitor your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

1. **Data Dashboard** We render the incidents on the dashboard w.r.t their severity, location, and the timestamp. This makes it very easy to understand, estimate and allocate resources to meet the demands.

We’ve used AWS Grafana to build visualizations for the data

Graphical user interface

Description automatically generated with medium confidence

Fig 9. Graphana Dashboards displaying alerts.

# Justification

To justify this model, we use the cloud services from AWS for the following reasons.

* Generally, on-prem systems are designed for constant pattern, they fail when
* Volume of the data increases during peak time
* Latency in the data leads to useless data
* Emergency services will not be available if the on-prem system fails during disaster situations.
* Prevents the loss of the data and reduces the costs of the infrastructure.

Chart

Description automatically generated

Fig 10. Chart displaying the peak data

In the above picture, it clearly indicates the volume of calls during the peak times. Our system is designed to handle any higher volumes of calls with the auto scalability features available in the cloud.

**Time** is key for emergency accidents, as the time elapses, the data will become useless. Mike Gualtieri has provided some valuable insights about how the value of the data diminishes over time [2]. As the data loses its value over time, it is very important to react in no time, and further the records are used for business intelligence for future analysis.

Timeline

Description automatically generated

Fig 11 . Chart displaying the importance of streaming data.

# Each Individual report

Link: https://github.com/Kishan19961998/StreamingPlatForm\_CarAccident

Kishan Byrosu

* Worked on Data processing of incoming incidents from the Data Streamer and Transmitter.
* Implemented analytical and logical unit in which the conditions on the incidents received from the data streamer and perform data transformation and validation are executed using Flink and AWS Glue.
* Built the flink code, that includes filtering stream1 and storing in a separate stream 2.
* Worked on creating a pipeline to stream the real-world data to perform analytical and logical operations and obtain the timely insights using Amazon Kineses.

Bhuvana Prakash

* Used data Streamer to create a new pipeline for the data that has been transformed and filtered by the data processor.
* Worked on building an event-based compute system using AWS Lambda to pass the inputs and transmit further to Dashboards, Logs and Alerting Service.
* Implemented the alerting service to send alerts about incident information received from Data handler by using Amazon Simple Notification Service (SNS) to build a robust alerting service which sends notifications two ways, A2A and A2P.
* Built the lambda function for records.

Siva Sai Sampath Kumar Vanajavaka

* Implemented the rendering of incidents on dashboard with respect to their severity, location, and the timestamp.
* Worked extensively on building visualizations for the data by using AWS Grafana.
* Handled processing, filtering of incidents that had to be pushed into a logging service for auditing and analytical purposes.
* Worked on monitoring CloudWatch to collect and track metrics, which are variables to measure the resources and applications.

# Conclusion

In this model, we considered usability as one of the key performance metrics and the final dashboard in the Grafana will alert for a highly rated severity in accidents also it will notify the people with simple notifications like email, which will help to take quick action, which is an important public safety challenge. Also, Scalability is a bottleneck situation during peak hours, considering the above factors as the benchmark indicators, we are implementing the project to solve the public safety challenge.

Source Code Download: https://github.com/Kishan19961998/StreamingPlatForm\_CarAccident

##### **References**

1. <https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents>
2. <https://advance.biz-tech-insights.com/whitepaper/Perishable-Insights-Stop-Wasting-Money-On-Unactionable-Analytics.pdf>
3. <https://smoosavi.org/datasets/us_accidents>
4. <https://docs.aws.amazon.com/>
5. <https://aws.amazon.com/kinesis/>