PROGETTO BIG DATA

Simone Ferrari VR479912

Real Time Straming Pipeline applicata alla Formula 1

Python, FastF1, Spark, Spark Streaming, Kafka, MongoDB, Grafana

INTRODUZIONE

I Big Data e lo Streaming Processing

I Big Data

Il termine "Big Data" è diventato sempre più comune negli ultimi anni, ad indicare smisurate quantità di dati che richiedono soluzioni tecnologiche avanzate per essere gestite, analizzate ed interpretate. Tali dati possono provenire da molteplici fonti al giorno d'oggi sempre più comuni e diffuse, come ad esempio i social media, le transazioni finanziarie, le reti di sensori, reti di videosorveglianza e molto altro ancora. La storia dei Big Data risale agli anni '90, quando la quantità di dati prodotta in tutto il mondo iniziò a crescere in modo esponenziale. Con l'avvento dell'era digitale e l'aumento della connettività, sempre più dati vennero generati e archiviati, creando la necessità di nuove soluzioni tecnologiche per gestirli.

L'analisi dei Big Data può fornire informazioni preziose per molte organizzazioni, consentendo di migliorare i processi decisionali, di identificare nuove opportunità di business e di innovare il proprio approccio al mercato. Grazie alle opportunità offerte dall'analisi dei Big Data, molte organizzazioni stanno scoprendo ed attuando nuovi modi di migliorare i propri processi decisionali, di identificare nuove opportunità di business e di innovare il proprio approccio al mercato. L'analisi di grosse moli di dati rappresenta però una sfida per nulla banale, dal momento che richiede competenze avanzate di programmazione, matematica e statistica, oltre a soluzioni tecnologiche sofisticate per gestire e analizzare grandi quantità di dati in tempi rapidi e con elevata precisione. Inoltre, spesso l'analisi dei Big Data deve tenere conto di questioni legate alla privacy dei dati e alla sicurezza delle informazioni, per evitare rischi di furti di informazioni sensibili o di violazioni della privacy dei dati. Le fonti dati per l'analisi dei Big Data possono essere di varia natura e includono, a fini esemplificativi, dati provenienti da social network, dati provenienti da sensori IOT, dati provenienti da stazioni di raccolta di informazioni, dati provenienti da testi o audio, e molto altro ancora.

L'impiego dei Big Data è oggi molto diffuso in vari settori, come ad esempio il marketing, la finanza, la sanità e la ricerca scientifica. Grazie alla loro capacità di raccogliere, analizzare e interpretare grandi quantità di dati, le organizzazioni possono trarre informazioni preziose sulle preferenze dei propri clienti, sui trend di mercato e sulle opportunità di business. Inoltre, l'analisi dei Big Data può consentire alle aziende di migliorare la propria efficienza operativa, ridurre i costi e ottimizzare la gestione delle risorse. Di contro, tra le sfide principali dell'analisi su vasta scala, vi

sono l'identificazione di fonti dati rilevanti, la gestione e la conservazione dei dati, la sicurezza delle informazioni e la privacy dei dati, l'accesso ai dati da parte degli utenti, la qualità dei dati e la capacità di analizzare grandi quantità di informazioni in modo rapido ed efficiente. Per affrontare le sfide descritte, le aziende possono quindi adottare soluzioni tecnologiche ad hoc, come ad esempio software di analisi dei dati, piattaforme di big data, tecnologie di intelligenza artificiale e di machine learning, che consentono di gestire e analizzare grandi quantità di dati in modo efficace. Inoltre, le organizzazioni stanno spingendo nella formazione di personale specializzato, per acquisire le competenze necessarie per gestire e analizzare i Big Data in modo efficace. Questo nuovo mondo in continua crescita rappresenta una grossa opportunità per le organizzazioni di tutti i settori, ma come visto richiede competenze avanzate e soluzioni tecnologiche sofisticate per essere implementata con successo.

Pipeline Streaming

Le pipeline streaming sono una tra le tante tecnologie importanti per la gestione dei Big Data. Consentono di elaborare grandi quantità di dati in tempo reale, o quasi, permettendo alle organizzazioni di agire immediatamente sui dati e di adattarsi rapidamente ai cambiamenti nel mercato. Le pipeline streaming sono utilizzate in una vasta gamma di applicazioni, ad esempio per la rilevazione delle frodi, l'elaborazione delle transazioni finanziarie, la gestione dei sensori IoT (Internet of Things), la gestione delle reti sociali e l'analisi delle opinioni dei clienti.

Una pipeline streaming è composta da una serie di componenti software che lavorano insieme per elaborare i dati con bassa latenza. In una struttura di base, i dati vengono raccolti da diverse fonti e quindi inviati a un sistema di elaborazione in tempo reale che li elabora e li analizza. Una volta elaborati, i dati vengono immagazzinati in un database o in un altro sistema di archiviazione per un utilizzo successivo, come ad esempio un monitor grafico live o un sistema di interrogazione analitico. Queste pipeline rappresentano una soluzione efficace per la gestione dei Big Data in tempo reale, ma presentano anche alcune ulteriori sfide specifiche. La principale tra queste è rappresentata dalla gestione della latenza, ovvero il tempo necessario per elaborare i dati e renderli disponibili per l'analisi. Questa può avere un impatto significativo sulle decisioni aziendali, quindi è importante minimizzarla il più possibile. Un'altra sfida importante è rappresentata dalla gestione della perdita dei dati. I dati raccolti dalle pipeline streaming possono essere estremamente preziosi, quindi è importante garantire che non vengano persi o corrotti durante il processo di elaborazione. Infine, un'altra sfida importante è la scalabilità. Le pipeline streaming devono essere in grado di gestire grandi quantità di dati, ma anche di

adattarsi alle esigenze in evoluzione delle organizzazioni. Ciò richiede una pianificazione attenta e una progettazione robusta delle pipeline.

Il contesto: Formula 1

La Formula 1 è uno sport estremamente tecnologico in cui le scuderie utilizzano conoscenze avanzate per ottenere il massimo delle prestazioni dalle loro vetture. Le macchine di Formula 1 sono dotate di una vasta gamma di sensori che raccolgono una quantità enorme di dati ogni volta che la macchina compie un giro in pista. Alcune stime indicano che una singola macchina di Formula 1 può generare più di 10 terabyte di dati durante un solo weekend di gara. Tali informazioni sono di vitale importanza per le scuderie, poiché consentono di monitorare le prestazioni della macchina, identificare eventuali problemi e migliorare le prestazioni giro dopo giro. Inoltre, i dati raccolti vengono utilizzati per fare previsioni sul comportamento della macchina in diverse condizioni di pista, aiutando così la scuderia a prendere decisioni più informate durante le gare. Per elaborare tutti questi dati in tempo reale e renderli disponibili per l'analisi della scuderia, è necessario utilizzare strumenti informatici avanzati come le pipeline streaming a bassa latenza. Queste tecnologie consentono di elaborare i dati in tempo reale e di rendere disponibili i risultati istantaneamente, senza alcun ritardo.

Le informazioni raccolte dai sensori della macchina di Formula 1 sono molto varie e includono dati sulla telemetria, sulle prestazioni del motore, sulla pressione degli pneumatici, sulla temperatura dei freni e molto altro. Questi devono essere elaborati in tempo reale e resi disponibili alla scuderia in modo semplice ed efficace così da poter permettere presa di decisioni rapida e informata durante le gare. L'utilizzo di pipeline streaming a bassa latenza è di vitale importanza per le scuderie di Formula 1, in quanto consente di elaborare e analizzare grandi quantità di dati in tempo reale e di utilizzare le informazioni per migliorare le prestazioni delle macchine in pista. Senza tali tecnologie avanzate, sarebbe impossibile raccogliere e analizzare tutte le informazioni necessarie per competere al massimo livello in questo sport estremamente all’avanguardia.

Oltre ai dati raccolti dai sensori della macchina, le scuderie di Formula 1 utilizzano anche una vasta gamma di strumenti di analisi dei dati per monitorare le prestazioni della macchina. Questi strumenti includono algoritmi di machine learning e modelli di previsione che consentono di prevedere il comportamento della macchina in diversi circuiti e in diverse condizioni. Come esempio, un modello di previsione può essere utilizzato per il comportamento in caso di pioggia o in condizioni di temperatura elevate. Queste previsioni sono fondamentali per determinare le

strategie di gara, come la scelta degli pneumatici e il momento in cui effettuare i pit stop. Le scuderie utilizzano spesso piattaforme cloud basate su tecnologie come Apache Kafka e Apache Spark per elaborare tali dati. L'elaborazione spesso avviene in modo distribuito, riducendo i tempi di latenza e aumentando l'efficienza dell'elaborazione dei dati. Inoltre, le scuderie utilizzano anche strumenti di visualizzazione dei dati per rendere le informazioni più facilmente comprensibili e utilizzabili. Ad esempio, i dati sulla telemetria della macchina possono essere visualizzati in grafici e diagrammi per facilitare l'analisi delle prestazioni della macchina.

Riassumendo, l'elaborazione dei dati in tempo reale e l'analisi delle prestazioni della macchina sono fondamentali per risultare competitivi in Formula 1. L'utilizzo di pipeline streaming a bassa latenza e di tecnologie avanzate come Apache Kafka e Apache Spark consente alle scuderie di elaborare grandi quantità di dati in modo distribuito e di prendere decisioni rapide e informate durante le gare. Senza queste tecnologie avanzate, sarebbe impossibile competere in uno sport dove ogni millisecondo conta e dove una minima differenza di prestazioni della macchina può fare la differenza tra la vittoria e la sconfitta.

Obiettivo

Il progetto si propone quindi di studiare le tecnologie impiegate dalle grandi aziende per la gestione dei big data in streaming, e di applicare le migliori soluzioni alla creazione di una pipeline completa per l'elaborazione dei dati telemetrici di una macchina di Formula 1 in tempo reale. L'obiettivo consiste quindi nello sviluppo di un'architettura stabile ed a bassa latenza che permetta di ricevere, elaborare, visualizzare e salvare i dati telemetrici ricevuti dalla macchina in tempo reale.

Il progetto si concentrerà sull'utilizzo di tecnologie come Apache Kafka e Apache Spark per l'elaborazione dei dati in tempo reale, e su strumenti di visualizzazione dei dati come Grafana per la creazione di una dashboard aggiornata in tempo reale che consenta di monitorare le prestazioni della macchina durante la sua corsa.
Poiché i dati telemetrici delle scuderie di Formula 1 sono protetti e riservati, il progetto si baserà solamente su dati pubblici disponibili online tramite il servizio *FastF1*[1], come la velocità, l'accelerazione, la frenata, il numero di giri al minuto del motore, la marcia utilizzata e l'uso del sistema DRS. Tuttavia, anche questi dati possono fornire una panoramica utile sulle prestazioni della macchina e consentire di studiare l'effetto di varie condizioni di pista sulla sua performance.

[1] FastF1, Python Library: https://theoehrly.github.io/Fast-F1/

L'obiettivo finale quindi è quello di simulare il comportamento di una macchina di Formula 1 in pista, raccogliendo i dati telemetrici disponibili inviati dal veicolo, elaborarli in tempo reale e visualizzarli su una dashboard interattiva che consenta di monitorare le prestazioni della macchina in tempo reale.

ARCHITETTURA

Tecnologie impiegate

Il progetto affrontato richiede l'utilizzo di diverse tecnologie che, configurate adeguatamente in sequenza, permettono la creazione della pipeline streaming. Tra le tecnologie utilizzate, possiamo annoverare linguaggi di programmazione, framework, strumenti di sviluppo e database. Di seguito se ne specificano i dettagli.

Apache Kafka

Apache Kafka è una tra le piattaforme open source più popolari per il processing di stream di dati in tempo reale. È stata sviluppata inizialmente da LinkedIn per gestire il traffico delle attività degli utenti, per poi essere donata all'Apache Software Foundation, dove ha ricevuto un notevole supporto dalla comunità open source. Il progetto Kafka è scritto principalmente in Java e Scala, ma offre anche una vasta panoramica di librerie per molti altri linguaggi, il che lo rende flessibile e adatto ad applicazioni in diversi linguaggi. L'obiettivo principale di Kafka consiste nel fornire una piattaforma ad alta velocità e bassa latenza per la gestione di dati in tempo reale. La piattaforma funziona come un sistema di messaggistica distribuito, che permette di scambiare dati tra applicazioni in modo affidabile e scalabile. La struttura di Kafka è basata su un modello publisher-subscriber, dove i dati vengono pubblicati in uno o più "topic" e i consumatori si iscrivono a questi topic per ricevere i dati in tempo reale immessi dai produttori. Gli utilizzi di Apache Kafka sono molteplici e diversificati. Grazie alla sua capacità di gestire grandi quantità di dati in tempo reale, viene utilizzata in applicazioni come la raccolta e l'analisi di dati IoT (Internet of Things), il monitoraggio e la registrazione di eventi di sistema, la gestione di log di applicazioni, l'elaborazione di dati di social media, il data warehousing in tempo reale e molto altro ancora.

Apache Spark

Spark è una piattaforma di elaborazione dati open source, progettata per fornire prestazioni elevate e scalabilità per una vasta gamma di applicazioni di elaborazione dati. A differenza del paradigma di elaborazione dati MapReduce di Hadoop, che si basa su un sistema di archiviazione a due livelli su disco, Spark utilizza una

tecnologia di elaborazione "in-memory" multilivello, che offre prestazioni fino a 100 volte migliori per alcune applicazioni. Questa tecnologia di elaborazione "in-memory" multilivello di Spark consente di caricare i dati in una memoria distribuita e interrogarli ripetutamente, migliorando significativamente le prestazioni di elaborazione dei dati. Grazie a tale tecnologia, Spark è particolarmente adatto per l'elaborazione di algoritmi di Machine Learning, in cui le prestazioni di elaborazione dati sono particolarmente importanti. Per quanto riguarda l'archiviazione distribuita, Spark può interfacciarsi con una vasta gamma di soluzioni, tra cui Hadoop Distributed File System (HDFS), Apache Cassandra, Amazon S3, e altre soluzioni personalizzabili. Inoltre, Spark supporta anche soluzioni pseudo-distribuite in modalità locale, utilizzate di solito per lo sviluppo o scopi di test, dove l'archiviazione distribuita non è richiesta e viene utilizzato il file system locale.

Apache Spark Streaming

Spark Streaming è un'estensione dell'API core di Spark che consente l'elaborazione di flussi di dati in tempo reale scalabile, ad alta velocità e tollerante ai guasti. I dati possono essere acquisiti da molte fonti come Kafka, Kinesis o socket TCP, e possono essere elaborati utilizzando algoritmi complessi espressi con funzioni ad alto livello come map, reduce, join e window. Infine, i dati elaborati possono essere inviati a file system, database e dashboard in tempo reale. Inoltre, è possibile applicare gli algoritmi di machine learning e di elaborazione di grafici di Spark ai flussi di dati.

Apache Spark SQL

Spark SQL è un modulo di Spark per l'elaborazione di dati strutturati. A differenza dell'API RDD di base di Spark, le interfacce fornite da Spark SQL forniscono a Spark maggiori informazioni sulla struttura sia dei dati che dell'elaborazione in corso. Internamente, Spark SQL utilizza queste informazioni extra per eseguire ulteriori ottimizzazioni. Ci sono diverse modalità per interagire con Spark SQL, tra cui SQL e l'API Dataset. Quando si calcola un risultato, viene utilizzato lo stesso motore di esecuzione, indipendentemente dall'API o dal linguaggio utilizzato per esprimere l'elaborazione. Questa unificazione permette agli sviluppatori di passare facilmente da un'API all'altra in base a quale fornisce il modo più naturale per esprimere una determinata trasformazione.

MongoDB

MongoDB è un DBMS non relazionale, sviluppato inizialmente dalla società di software 10gen, che ora è nota come MongoDB Inc. Il nome "Mongo" deriva dalla parola "humongous", che significa "enorme", per evidenziare la natura scalabile e performante del sistema. Classificato come un database di tipo NoSQL, MongoDB si

allontana dalla struttura tradizionale basata su tabelle dei database relazionali, optando invece per un'architettura basata su documenti in stile JSON. Questo approccio consente una maggiore flessibilità e agilità nell'integrazione dei dati, rendendo più facile e veloce l'utilizzo di MongoDB per alcune tipologie di applicazioni. MongoDB utilizza il formato BSON (Binary JSON) per rappresentare i dati, fornendo una maggiore efficienza e una migliore gestione dei tipi di dato rispetto al formato JSON standard. Grazie alla sua architettura distribuita, MongoDB è in grado di gestire grandi quantità di dati e di scalare orizzontalmente in modo efficiente. Nel corso degli anni, MongoDB ha acquisito una grande popolarità nella comunità degli sviluppatori, diventando uno dei database non relazionali più utilizzati al mondo. Nel 2009, la società ha adottato un modello di sviluppo open source, che ha permesso alla comunità di contribuire attivamente allo sviluppo del software. Tuttavia, la società continua ad offrire servizi commerciali e di supporto per i clienti che necessitano di soluzioni su misura.

Grafana

Grafana è un'applicazione web potente e versatile utilizzata per la visualizzazione e l'analisi interattiva dei dati. Lanciata nel 2014 come software libero multipiattaforma, Grafana è diventata uno strumento di visualizzazione di dati di primo piano grazie alla sua capacità di unificare varie sorgenti di dati. Grafana permette di creare infografiche tramite la creazione di appositi pannelli (dashboard). La creazione di questi pannelli è facilitata dall'utilizzo di query builder interattivi che consentono di creare rapidamente pannelli di visualizzazione personalizzati. Una delle principali caratteristiche di Grafana è la sua capacità di supportare una vasta gamma di sorgenti dati. Alcune di queste sono incluse in modo predefinito, come Elasticsearch, MySQL e Prometheus, mentre altre tecnologie possono essere integrate attraverso componenti aggiuntivi, alcuni gratuiti altri a pagamento: in totale sono circa un centinaio tra cui MongoDB, GitLab, Jira, PostgreSQL, Solr e Zabbix. Grafana è particolarmente adatta per la visualizzazione di dati di database e strumenti per serie temporali come InfluxDB, Graphite e Prometheus. Grazie alla sua architettura modulare e alla vasta gamma di sorgenti dati supportate, Grafana è diventato uno strumento essenziale per l'analisi dei dati e la visualizzazione delle informazioni in modo chiaro e intuitivo.

Python

Python è un popolare linguaggio di programmazione open source con una vasta gamma di applicazioni. È un linguaggio di alto livello orientato agli oggetti, utilizzato per sviluppare software distribuiti, applicazioni web, automazione di sistema, data science e molte altre applicazioni. Ideato da Guido van Rossum

all'inizio degli anni '90, deve il suo nome alla sua passione per la serie televisiva dei Monty Python. Questo linguaggio è spesso paragonato ad altri linguaggi come Ruby, Perl, JavaScript e Scheme per la sua sintassi semplice e la sua capacità di scrivere codice leggibile e manutenibile. È considerato uno dei primi linguaggi di programmazione ideali per i principianti. Inoltre, la vasta gamma di librerie e framework disponibili rendono facile sviluppare applicazioni in vari settori, come la scienza dei dati, l'apprendimento automatico, la sicurezza informatica e molto altro. Flask e Django, ad esempio, sono due popolari framework per lo sviluppo di applicazioni web. Python può anche essere usato per l'automazione del sistema e la creazione di script, rendendolo uno strumento popolare tra gli amministratori di sistema e gli sviluppatori DevOps. Inoltre, Python è diventato un linguaggio preferito nel settore della Data Science, grazie alla sua vasta gamma di librerie per la manipolazione e l'analisi dei dati, come NumPy, Pandas, Matplotlib e Spark. Python viene utilizzato anche nell'apprendimento automatico grazie alle librerie come Scikit-Learn e TensorFlow. Infine, Python è una scelta popolare anche nel settore della sicurezza informatica, grazie alla sua capacità di scrivere script per la rilevazione e l'eliminazione di malware e vulnerabilità di sicurezza. In sintesi, Python è un linguaggio di programmazione versatile e potente, che può essere utilizzato in molte applicazioni diverse.

Panoramica

Si procede ora con l'analisi dettagliata dell'architettura e delle singole componenti della pipeline streaming, al fine di spiegare come sia stato possibile raggiungere l'obiettivo prefissato. L'architettura generale della pipeline si compone di diverse fasi, ognuna delle quali è svolta da specifiche tecnologie e strumenti software. In primo luogo, si ha la fase di acquisizione dati, che avviene tramite apposite fonti di dati (nell' esempio, da sensori posti nella vettura di F1) e attraverso l'utilizzo di tecnologie come Apache Kafka. Successivamente, si procede con la fase di elaborazione dei dati. Questa fase è svolta tramite l'uso della tecnologia Apache Spark Streaming. A conclusione, si ha la fase di salvataggio e visualizzazione dei dati, in cui si rappresentano graficamente i dati elaborati e analizzati, al fine di renderli fruibili agli utenti finali e li si salvano per renderli persistenti.

Questa fase è svolta tramite l'uso di tecnologie come Grafana e MongoDB, che consentono di visualizzare i dati in modo intuitivo e personalizzato e di salvarli in un sistema facilmente scalabile e flessibile.

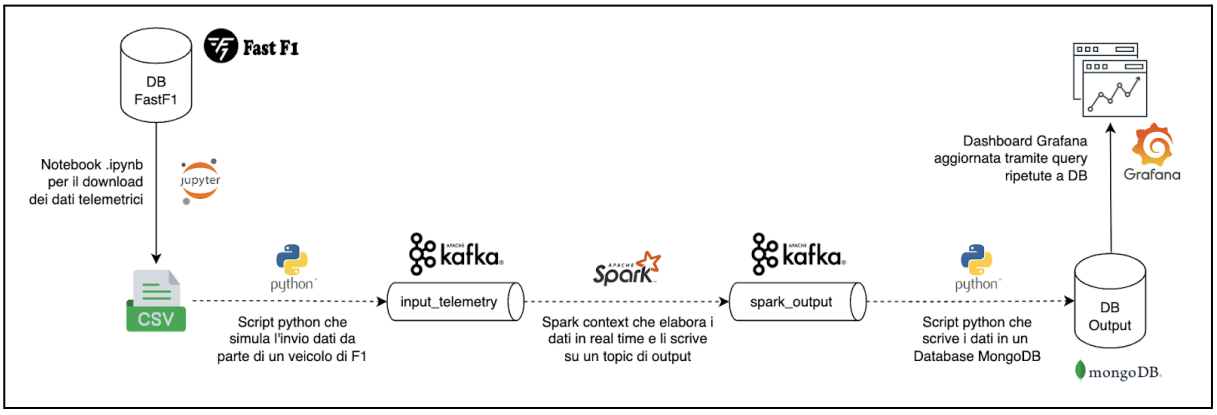

Figura 1: architettura della streaming pipeline.

Di seguito, una descrizione ad alto livello delle varie componenti, facendo riferimento al codice sorgente contenuto nella repository pubblica seguente: *https://github.com/SimoneFerrari99/BD-F1TelemetryLiveAnalysis.git*

Generazione file Telemetry.csv

File di riferimento: */01GetTelemetry/Formula1.ipynb*

Il file in questione è stato progettato per interfacciarsi con i servizi API pubblici offerti da FastF1, una piattaforma specializzata nel recupero dei dati di telemetria di Formula1. Attraverso l'utilizzo di questi servizi, è possibile recuperare informazioni dettagliate e in tempo reale su ogni aspetto della performance di una vettura, compresi i dati relativi alla velocità, alla potenza, all'accelerazione, alla frenata e molto altro ancora. Nell'esempio specifico, è stato scelto di recuperare i valori relativi al giro di Pole Position effettuato dal pilota Leclerc durante il weekend di gara in Bahrain del campionato 2022. In particolare, sono stati recuperati i valori relativi al tempo sul giro, ai giri al minuto del motore (RPM), alla velocità, alla marcia, all'accelerazione, alla frenata, al DRS e alla distanza percorsa.

Invio dati streaming

File di riferimento: */02Producer_CarSimulation/producer.py*

Questo script Python legge i dati di telemetria da un file CSV (`'../01GetTelemetry/telemetry.csv'`) e li invia a un topic Apache Kafka in un topic denominato `'input_telemetry'`. Lo scopo è quindi quello di simulare l'invio streaming dei dati da parte di una vettura di Formula 1 mentre è in pista. Per prima

cosa, vengono importati i moduli necessari, ovvero `sleep`, `dumps`, `KafkaProducer`, `datetime` e `csv`. Viene quindi istanziato un oggetto KafkaProducer, che si connette al topic Kafka in ascolto all'indirizzo `'localhost:9092'`. Successivamente, viene aperto il file CSV contenente i dati di telemetria e viene creato un oggetto `csv_reader` per leggerli. Il codice scorre quindi ogni riga del file, escludendo la prima riga che contiene solo i nomi delle colonne. Per ogni riga del file CSV, viene creato un dizionario chiamato rowData che contiene i dati di telemetria. In particolare, vengono estratti il tempo, il numero di giri al minuto (RPM), la velocità, la marcia, l'acceleratore, la frenata, lo stato del sistema DRS e la distanza percorsa. Infine, il dizionario contenente i dati di telemetria viene inviato al topic Kafka attraverso il metodo `producer.send()`. La variabile `line_count` viene incrementata per ogni riga letta dal file, e viene poi introdotto un ritardo di `0.1` secondi tra l'invio di un messaggio e il successivo, utilizzando il metodo `sleep()`.

Elaborazione dati

File di riferimento: */03Consumer/consumerSpark.py*

Questo script in Python utilizza la libreria PySpark per creare una sessione Spark e leggere i dati in streaming da un topic di Kafka denominato "`input_telemetry`". La lettura dello streaming avviene attraverso il metodo `readStream()` di Spark e viene specificato il formato dei dati in ingresso come "`kafka`". Successivamente, il metodo `selectExpr()` viene utilizzato per selezionare il contenuto delle colonne "key" e "value" del dataframe, e la stringa "`CAST(key AS STRING)`" e "`CAST(value AS STRING)`" viene utilizzata per convertire i dati binari letti da Kafka in stringhe. In conclusione, il dataframe risultante viene scritto in streaming sul topic "`spark_output`" sempre utilizzando il protocollo Kafka, utilizzando il metodo `writeStream()`, e specificando la posizione di checkpoint. Il metodo `awaitTermination()` viene utilizzato per far sì che il processo continui ad aspettare l'arrivo di nuovi dati dal topic di Kafka, finché non viene interrotto manualmente.

Scrittura dati a DB

File di riferimento: */04Visualization/store.py*

Questo codice Python si occupa di leggere dati da un topic Kafka (il topic di output dello step precedente) e li inserisce in una raccolta MongoDB. In particolare, il codice utilizza la libreria `KafkaConsumer` per leggere i messaggi dalla coda Kafka chiamata `'spark_output'`. Il parametro `'auto_offset_reset'` è impostato su `'earliest'`, il che significa che il consumer leggerà i messaggi dal primo disponibile nella coda.

Il consumer legge i messaggi uno per uno in un loop, quindi per ogni messaggio viene creato un dizionario chiamato "`row`" che contiene le informazioni sulla telemetria, ovvero il tempo, i giri al minuto del motore (RPM), la velocità, la marcia (nGear), la percentuale di acceleratore (Throttle), l'uso del pedale del freno (Brake), l'uso o meno del sistema DRS e la distanza percorsa. Questi valori sono presi dal campo "`value`" del messaggio Kafka, che viene convertito dal formato `JSON` utilizzando la funzione "`loads`". Infine, i dati vengono inseriti nella collection MongoDB utilizzando la funzione "`insert_one`" del driver Python per MongoDB. La collection viene chiamata "`telemetry`" e si trova all'interno del database chiamato "`telemetry`". Il nome dell'host e le credenziali di accesso sono forniti tramite la stringa di connessione passata in input alla funzione `MongoClient`.

Visualizzazione dati

File di riferimento: *[/04Visualization/dashboard.json]*

Le dashboard di Grafana possono essere personalizzate utilizzando un file di configurazione in formato *JSON*. Questo file rappresenta il codice necessario per la creazione della dashboard di Grafana, la quale consente di visualizzare i dati di interesse, come illustrato nell'immagine allegata. I file JSON di Grafana sono costituiti da una serie di oggetti e proprietà che definiscono la configurazione della dashboard. In genere, un file di configurazione JSON di Grafana inizia con l'oggetto "dashboard", che rappresenta la dashboard stessa. Al suo interno, vengono quindi specificati diversi parametri, tra cui il titolo della dashboard, la sua descrizione e le opzioni di visualizzazione. A seguire, sono presenti gli oggetti "panel", che rappresentano i singoli pannelli della dashboard. Ogni pannello ha un proprio tipo di visualizzazione (grafico, tabella, testo, ecc.) e una serie di proprietà che definiscono i dati da visualizzare, le opzioni di formattazione e le impostazioni di interazione. Il file in oggetto, quindi, si occupa della creazione della dashboard finale, al cui interno sono presenti diversi pannelli, precisamente 12, di varie tipologie. All'interno del file json sono anche specificate le query MongoDB utili al recupero dei dati dalla base dati.

ESECUZIONE

Avvio progetto

Per poter avviare correttamente il progetto, risulta necessario procedere per step. Di fatto, per prima cosa sarà necessario configurare e predisporre l'ambiente di esecuzione. In secondo luogo, si dovranno avviare i diversi servizi coinvolti all'interno del progetto ed, infine, sarà necessario eseguire gli script di simulazione invio dati e di computazione dei valori.

La fase di avvio prevede l'esecuzione di una serie di attività per preparare l'ambiente di lavoro. Tra queste attività rientrano l'installazione e la configurazione delle dipendenze e degli strumenti necessari. La soluzione proposta si basa sull'assunzione che la configurazione in locale si basi su container Docker. La procedura descritta è compatibile con i sistemi operativi Linux e MacOS, mentre per Windows, ci sono alcune differenze nei comandi da utilizzare.

Il primo passo per procedere è il download e l'installazione di Docker, preferibilmente attraverso Docker Desktop per maggior facilità di utilizzo. Una volta effettuato il download di Docker, è possibile procedere con la clonazione della repository utilizzando il seguente comando: `git clone https://github.com/SimoneFerrari99/BD-F1TelemetryLiveAnalysis.git`. Successivamente, si apra un nuovo terminale all'interno della cartella appena clonata e si esegua il file docker-compose mediante il comando: `docker-compose -f ./config/docker-compose.yml up -d`. Tale file si occuperà di scaricare le dipendenze relative al server Kafka, in particolare creerà un nuovo container docker, se non già presente, e installerà le immagini relative allo Zookeeper e a Kafka. Infine, il comando eseguirà il container con i relativi volumi, garantendo così l'effettiva esecuzione del server Kafka sulla porta standard `9292` nel server locale.

A questo punto, si proceda con l'installazione delle dipendenze necessarie per gli script Python. Aprire quindi un terminale all'interno della cartella radice del progetto clonato e creare un ambiente virtuale con il seguente comando: `python3 -m venv venv`. Attivare l'ambiente virtuale tramite il comando: source venv/bin/activate. Installare quindi le dipendenze relative a Kafka (`pip3 install kafka-python`), a pyspark (`pip3 install pyspark`) e a pymongo (`pip3 install pymongo`).

Una volta completata l'installazione delle dipendenze necessarie, l'ambiente di esecuzione risulta quasi pronto. Resta da configurare il server MongoDB e la

dashboard Grafana. Per quanto riguarda il primo, in questa soluzione si consiglia di creare un cluster gratuito tramite MongoDB Atlas. Per fare ciò, è possibile accedere al sito ufficiale (*MongoDB Atlas*) e creare un account personale. Dopo aver creato un cluster gratuito, è necessario generare un database, una propria utenza e password. Successivamente, inserire le credenziali di accesso al database all'interno del file `/04Visualization/store.py`, in particolare a riga `15`, indicando `Username`, `Password` ed `URL` del cluster creato, reperibile tramite le impostazioni di connessione disponibili sul portale Atlas.

Per quanto riguarda la dashboard Grafana, è sufficiente accedere all'indirizzo ufficiale (*Grafana Downloads*) e creare un account personale. Successivamente, creare il proprio spazio di lavoro cloud e accedere all'indirizzo *https://nomespazio.grafana.net/dashboards*. Cliccando sul pulsante NEW e sulla voce IMPORT, sarà possibile incollare il contenuto del file `/04Visualization/dashboard.json` e importare la dashboard con tutte le configurazioni annesse. Naturalmente, anche per Grafana è necessario inserire le credenziali di accesso al cluster MongoDB. Per fare ciò, accedere alle impostazioni della dashboard, selezionare "data source", installare e modificare le configurazioni relative al plugin MongoDB. La dashboard sarà quindi attiva e visualizzabile, anche se non mostrerà alcun dato.

Per avviare il progetto descritto, è necessario eseguire alcune operazioni specifiche in sequenza. Prima di tutto, si deve aprire il terminale e posizionarsi nella cartella "`02Producer_CarSimulation`" del progetto clonato precedentemente. Successivamente, in un nuovo terminale, ci si deve spostare nella cartella "`03Consumer`", mentre in un terzo terminale si deve andare nella cartella "`04Visualization`".

Una volta che si è in queste cartelle, si può procedere all'avvio dell'applicazione spark nel secondo terminale, digitando il seguente comando:

```
spark-submit --packages
org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.2 consumerSpark.py
```

Questo comando lancia l'applicazione spark che verrà utilizzata per elaborare i dati in arrivo dal server Kafka.

Nel terzo terminale, invece, si deve eseguire il file "store.py" digitando il seguente comando: `python3 store.py`
Questo script è responsabile dell'archiviazione dei dati elaborati nel database MongoDB.

Infine, nel primo terminale, si deve avviare il producer (l'auto di Formula 1) digitando il comando: `python3 producer.py`

Interpretazione dashboard

Di seguito si presentano i 12 pannelli visualizzati nella dashboard Grafana, fornendone anche qualche chiave di lettura. Tra questi, 5 pannelli sono di tipo temporale, ovvero mostrano dati nel tempo, 6 sono di tipo istantaneo, ovvero mostrano dati in un momento specifico, mentre uno è di tipo testuale, che visualizza informazioni sotto forma di testo.

La rappresentazione grafica sottostante è la dashboard completa che emerge dal caso di studio relativo al progetto in questione. La dashboard fornisce una visione d'insieme dei dati raccolti durante il progetto, presentati in forma sintetica e facilmente interpretabile. Essa rappresenta uno strumento di monitoraggio essenziale per un ingegnere di pista, consentendo al muretto box di effettuare valutazioni e strategie accurate sulla base di informazioni aggiornate.

Figura 2: dashboard grafana contenente 12 pannelli.

Si possono idealmente separare la dashboard in due macro sezioni: la prima, la sezione delle informazioni istantanee, rappresentate dalla prima riga della griglia, contiene i dati relativi alla situazione attuale, come velocità istantanea e accelerazione istantanea. La seconda sezione, invece, composta da 5 grafici, rappresentano nel tempo l'andamento delle stesse informazioni.

Speed (grafico istantaneo)

Il grafico istantaneo stile tachimetro utilizzato in Formula 1 è uno strumento essenziale sia per i piloti che per le squadre al fine di monitorare le prestazioni del veicolo in tempo reale. Questo tipo di grafico rappresenta appunto la velocità istantanea del veicolo in km/h, con una scala graduata che va da 0 a un valore massimo che rappresenta la velocità massima del veicolo. Solitamente questa tipologia di informazione può essere utile durante le fasi di qualificazione e di gara, quando ogni singolo km/h orario conta. Con un solo sguardo, gli ingegneri di pista possono verificare se stanno raggiungendo la velocità desiderata e se ci sono eventuali problemi con il motore.

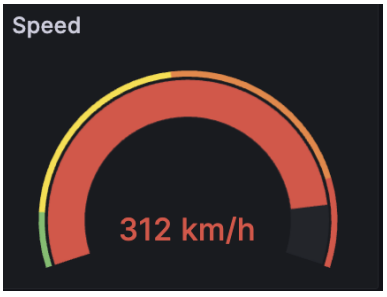

Figura 3: Tachimetro Speed

Throttle (grafico istantaneo)

Il tachimetro "Throttle" rappresenta un altro strumento essenziale utilizzato in Formula 1 per monitorare le prestazioni del veicolo in tempo reale. Questo tipo di grafico rappresenta la percentuale di pressione del pedale dell'accelerazione del veicolo, con una scala graduata che va dal 0 al 100%. Tale grafico permette alla squadra di valutare le prestazioni del motore e di individuare eventuali problemi di performance, come ad esempio un'erogazione di potenza irregolare rispetto alla pressione del pedale.

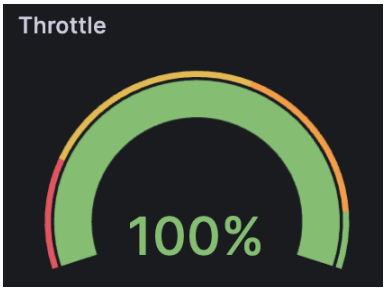

Figura 4: Tachimetro Throttle

RPM (grafico istantaneo)

Il tachimetro RPM rappresenta la velocità di rotazione del motore in giri al minuto, con una scala graduata che va da un valore minimo a un valore massimo, in genere compreso tra 10.000 e 15.000 giri al minuto. Questo strumento consente agli ingegneri di valutare la potenza del motore in tempo reale, individuando eventuali anomalie al cambio e al regime del motore.

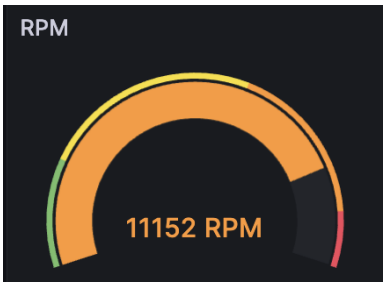

Figura 5: Tachimetro RPM

Lap Time (tempo istantaneo)

Il grafico testuale del tempo su giro non richiede molte spiegazioni: indica il tempo trascorso attualmente sul giro attuale ed è espresso in formato standard ore, minuti, secondi e millisecondi.

Figura 6: Tempo su giro

Gear (grafico istantaneo)

Il tachimetro “Fear” rappresenta il valore della marcia in cui il veicolo si trova, con una scala graduata che va da una marcia minima a una marcia massima, che dipende dalla configurazione del cambio del veicolo e che solitamente è su una scala da 1 ad 8. Questo strumento consente al muretto box di sapere sempre in quale marcia si trova il pilota e di verificare il corretto funzionamento del cambio e del sistema idraulico annesso.

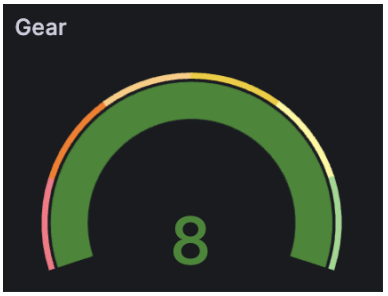

Figura 7: Tachimetro Gear

Brake (grafico istantaneo)

Il grafico istantaneo della frenata rappresenta la percentuale di pressione del pedale del freno applicata dal pilota in ogni momento, con una scala graduata che va dal 0% al 100%. Questo strumento consente ai piloti di sapere sempre quanto freno stanno applicando e di adattare la guida in funzione delle esigenze della pista.

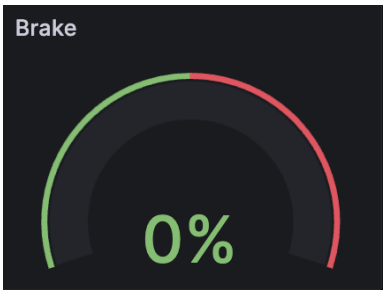

Figura 8: Tachimetro Brake

DRS (grafico istantaneo)

L'ultimo tachimetro a disposizione e di utilità per il muretto è l'uso del DRS - Drag Reduction System. Esso indica l'attivazione o meno di tale sistema e permette agli ingegneri di individuare rapidamente eventuali problematiche, come la corretta apertura o chiusura di esso.

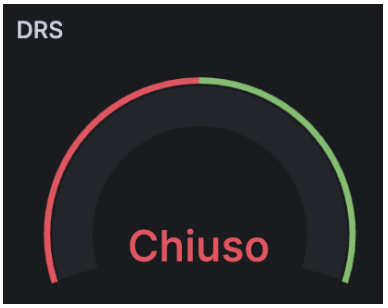

Figura 9: Tachimetro DRS

Speed (grafico temporale)

Il grafico della velocità è senza alcun dubbio il principale. Rappresenta infatti la velocità dell'auto di Formula 1 durante un giro sul circuito. Sull'asse delle X si ha solitamente il tempo trascorso o la distanza percorsa, mentre sull'asse delle Y abbiamo la velocità dell'auto in km/h. La linea del grafico solitamente inizia con una leggera salita corrispondente al rettilineo di partenza che sale fino al raggiungimento della velocità massima di circa 340 km/h. Dopo questo picco, la linea del grafico diminuisce più o meno bruscamente, mentre l'auto affronta curve e chicane, rallentando in modo da mantenere la traiettoria ideale. In alcuni punti del circuito, la linea aumenta bruscamente quando l'auto accelera, ad esempio in prossimità di una rettilineo lungo o di una sezione in discesa. Osservando il grafico, quindi, risulta molto semplice individuare i punti di rettilineo e di curva. La forma del

grafico potrebbe essere irregolare a causa di eventuali errori di guida o problemi tecnici che possono influire sulla velocità dell'auto. Ad esempio, potrebbero esserci delle deviazioni brusche nella linea del grafico quando l'auto affronta curve strette o quando i piloti cercano di superarsi reciprocamente. Infine, verso la fine del giro, la linea del grafico aumenta di nuovo mentre l'auto accelera per tagliare il traguardo.

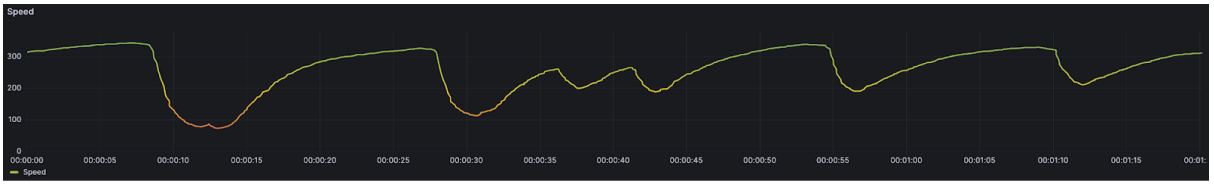

Figura 10: Grafico temporale Speed

Throttle & Brake (grafico temporale)

Il grafico del throttle rappresenta la percentuale di apertura dell'acceleratore dell'auto di Formula 1 durante un giro sul circuito. Come per il grafico precedente, sull'asse delle X si trova il tempo trascorso in secondi, mentre a differenza, sull'asse delle Y si ha la percentuale di apertura dell'acceleratore dell'auto. In condizioni naturali, ci si aspetta che dove il throttle va a 0, il brake salga, ad indicare un punto di frenata. Solitamente quindi questi due grafici vengono sovrapposti in quanto spesso complementari. Si può inoltre notare che in corrispondenza dei punti di frenata, il grafico precedente mostri una diminuzione della velocità.

Figura 11: Grafico temporale Throttle & Brake

RPM (grafico temporale)

Il grafico degli RPM indica l'andamento del regime del motore nel corso del tempo e/o distanza percorsa. Tale grafico solitamente risulta meno importante al fine di valutare la performance dell'auto, di contro, fornisce una chiara visione su eventuali problematiche al motore: ad esempio, un regime di motore irregolare rispetto al pattern previsto può indicare facilmente un possibile problema di affidabilità.

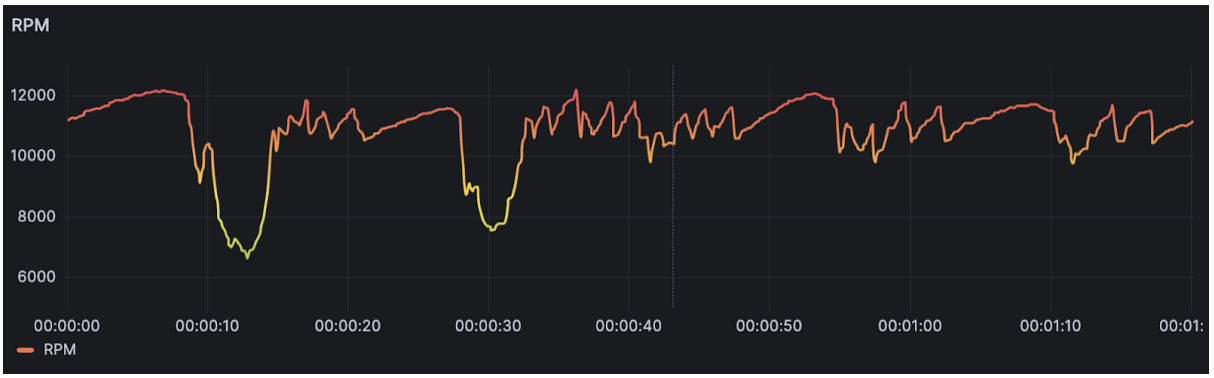

Figura 12: Grafico temporale RPM

Gear (grafico temporale)

Il grafico del gear rappresenta la marcia utilizzata dall'auto di Formula 1 durante un giro sul circuito. Sull'asse delle X si ha il tempo trascorso in secondi, mentre sull'asse delle Y vi figura il valore della marcia utilizzata dall'auto. Durante il giro, la marcia viene cambiata moltissime volte, anche più di 50 in circuiti complessi, in modo da mantenere la potenza dell'auto e adattarla alle diverse sezioni del circuito. In curva, si osserva una marcia più bassa per mantenere una velocità costante, mentre in prossimità di una sezione in discesa o di un rettilineo lungo, la marcia viene aumentata in modo da massimizzare la velocità dell'auto.

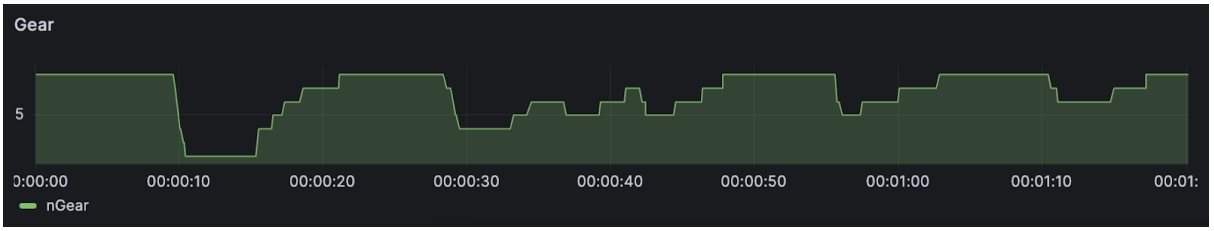

Figura 13: Grafico temporale Gear

DRS (grafico temporale)

Infine, il grafico temporale per l'uso del DRS pone sull'asse delle X il tempo/distanza e sull'asse delle Y il valore booleano TRUE o FALSE, ad indicare l'uso o meno del dispositivo DRS.

Figura 14: Grafico temporale DRS

Interpretazione dashboard rispetto la pista

I dati utilizzati come esempio riguardano il giro veloce di qualifica effettuato dal pilota Leclerc nel corso del weekend di gara in Bahrain 2022. Di seguito, quindi, si riporta un'immagine raffigurante il layout del circuito, e si propone una chiave di lettura per individuare le corrispondenze della dashboard rispetto al circuito.

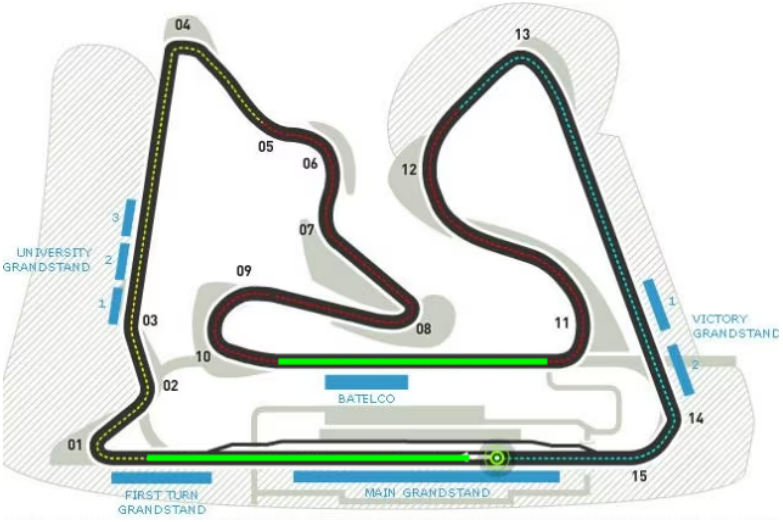

Figura 15: Circuito del Bahrain 2022

Nella figura soprastante, ogni curva è numerata, con un valore intero da 1 a 15, rispettivamente la prima e l'ultima. Le zone evidenziate in verde, invece, rappresentano le zone di DRS, ovvero i rettilinei in cui i piloti sono autorizzati all'apertura dell'ala mobile posteriore al fine di ridurre la resistenza all'avanzamento ed ottenere maggiore efficienza aerodinamica con conseguente aumento di velocità.

Figura 16: Dashboard con note di interpretazione rispetto al circuito

Prestazioni

Per quanto concerne le prestazioni, l'intera pipeline dimostra di essere altamente efficiente. L'analisi del dataset di esempio, che include la colonna "lap time" indicante il tempo impiegato per completare un giro, rivela che i dati trasmessi dall'auto presentano una distanza temporale approssimativa di 150ms tra di loro. Per simulare in modo preciso la trasmissione dei dati da parte di un'auto in pista, lo script di simulazione è stato configurato per inviare i dati uno alla volta, con un intervallo di 150ms tra ogni riga.

L'elaborazione dei dati sulla macchina locale, un MacBook Pro M1, con ambiente di sviluppo in container Docker e risorse limitate, non ha evidenziato rallentamenti o congestioni. Tuttavia, per quanto riguarda la componente MongoDB, che è ospitata su un'infrastruttura cloud tramite il servizio gratuito MongoDB Atlas e connessa tramite server condivisi, le prestazioni calano significativamente. In particolare, l'aumento del flusso di dati e dei produttori genera un collo di bottiglia nella scrittura dei dati sul database. Al contrario, l'utilizzo di una configurazione locale per il database non presenta tali rallentamenti, indicando che il cluster condiviso utilizzato per i test offre risorse limitate. Pertanto, l'adozione di un cluster dedicato dovrebbe risolvere questi problemi.

Per quanto riguarda la dashboard Grafana, l'esecuzione delle query sul database MongoDB non genera rallentamenti, nonostante l'infrastruttura cloud utilizzata offra risorse limitate attraverso il piano gratuito. Tuttavia, la dashboard ha un limite minimo di aggiornamento dei dati di 5 secondi, che è imposto dalle restrizioni di Granafa Cloud e non può essere superato. Inoltre, la soluzione di Grafana in locale non può essere testata per quanto riguarda l'aggiornamento in tempo reale dei dati poiché richiederebbe il recupero dei dati tramite il plugin Kafka, che non è compatibile con l'architettura in uso.

CONCLUSIONI

Conclusioni

Concludendo, il progetto si proponeva di creare una pipeline real-time per gestire i dati telemetrici di una vettura di Formula 1 nel corso di una gara, con l'obiettivo finale di visualizzare le informazioni raccolte attraverso una dashboard Grafana. La struttura della pipeline prevedeva la simulazione dell'invio delle informazioni da parte dell'auto, la ricezione e l'elaborazione dei dati, l'archiviazione degli stessi in un database e la visualizzazione dei risultati attraverso una dashboard intuitiva e di facile utilizzo. Grazie alla dashboard creata, gli ingegneri di pista possono accedere in tempo reale a informazioni chiare e pronte all'uso, come velocità, frenata e RPM, che consentono loro di prendere decisioni strategiche in modo rapido ed efficace. Va notato che, nonostante la pipeline risulti molto performante in ambiente locale, il sistema presenta come visto alcune limitazioni dovute all'utilizzo di componenti basate su cloud con licenze gratuite. Per ovviare a tali limiti, potrebbe essere necessario adottare soluzioni su cluster dedicati. In ogni caso, la pipeline real-time creata rappresenta un importante strumento per la gestione dei dati telemetrici di una vettura di Formula 1 e può essere utilizzata dalle scuderie per migliorare le prestazioni delle vetture e la strategia di gara.

SITOGRAFIA

Anderson, Martin. “Real-Time Big Data Analytics: Use Cases, Implementation Tips.” Real-Time Big Data Analytics: Use Cases, Implementation Tips, 1 Nov. 2021.
www.itransition.com/blog/real-time-big-data-analytics

“Building Scalable Streaming Pipelines for near Real-Time Features.” Uber Blog, 24 Aug. 2021.
www.uber.com/en-IT/blog/building-scalable-streaming-pipelines/

“How to Easily Visualize Your Mongodb Data.” Grafana Labs
grafana.com/solutions/mongodb/visualize/

Jasper. “How to Analyze Formula 1 Telemetry in 2022 - A Python Tutorial.” Medium, Towards Formula 1 Analysis, 31 Mar. 2022.
medium.com/towards-formula-1-analysis/how-to-analyze-formula-1-telemetry-in-2022-a-python-tutorial-309ced4b8992

Linux Foundation Events.
events17.linuxfoundation.org/sites/events/files/slides/fast_car_big_data_code_motion_carol3.pdf

“Realtime Data Processing at Facebook - Meta Research.” Meta Research.
research.facebook.com/publications/realtime-data-processing-at-facebook/

Sarfin, Rachel Levy. “Streaming Data Pipelines: Building a Real-Time Data Pipeline Architecture.” Precisely, 16 Nov. 2022.
www.precisely.com/blog/big-data/streaming-data-pipelines-how-to-build-one.

“Uber's Big Data Platform: 100+ Petabytes with Minute Latency.” Uber Blog, 17 Oct. 2018
www.uber.com/en-IT/blog/uber-big-data-platform/