F1 Live Timing

A Technology for Advanced Programming project

di Bonanno Giuseppe

COSA FA F1 LIVE TIMING?

Data ingestion dal server ufficiale F1

Pulizia e ripartizione nelle varie pipeline

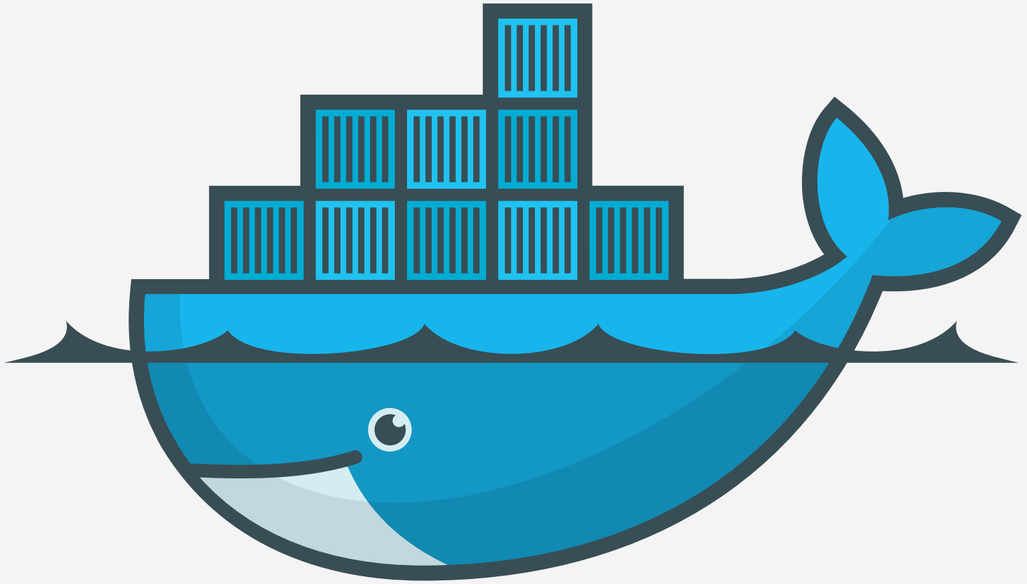
Gestione e lettura dello streaming

COSA FA F1 LIVE TIMING?

Visualizzazione di telemetria, passo gara, previsioni prossimo giro, gap col prossimo pilota e con il leader per ogni pilota e live.

Visualizzazione dei dati meteo della pista in tempo reale

Docker



Il progetto è tutto suddiviso
in Docker container.
Vediamo quali sono...

INIT-

Python + Azure SignalR (utilizzato dal servizio ufficiale di F1)

Attraverso lo script Python:

- l'iscrizione al servizio
- avvio sessione per la ricezione

Enormi quantità di dati in ingresso. Vengono filtrati, lasciando solo i **dati telemetrici, distanze e dati meteo.** Questi vengono spediti a **Logstash**.

Logstash
Suddivisione dei dati ricevuti in 3 pipeline:

1. Kafka (TimingData)
2. Elasticsearch (Gap)
3. Elasticsearch (Meteo)

Viene effettuato il casting del flusso nei tipi corretti e aggiunto un timetamp.

Init-Kafka
Aspetta l'avvio di kafka e ne crea un Topic se questo non presente.
Topic: "LiveTimingData".

Broker
Kafka riceve i singoli json e li accoda nel Topic "LiveTimingData" al quale accinge Spark.
Il broker si appoggia al container di Zookeeper.

PySpark (driver)

- Si iscrive al topic: LiveTimingData;
- Usa Structured Streaming per l'elaborazione del flusso;
- Ad ogni microbatch memorizza e categorizza in base al pilota
- Usa MLlib per predire il giro successivo per ogni pilota.
- Manda a Elasticsearch

Apache Spark (cluster)

- Cluster formato da master e 4 worker;
- Il driver (PySpark) si rifà al cluster per scalare orizzontalmente la potenza di elaborazione;
- Essenziale se si vuole avere l'analisi dei piloti in tempo reale.

Elasticsearch

Riceve da fonti diverse e suddivide i dati in 4 indici:

- lastlaptime index, con i dati dell'ultimo giro compiuto dai piloti;
- prediction index, con le previsioni di MLlib di ogni pilota;
- gaps index, contenente i distacchi tra i piloti;
- weather index, che contiene i dati meteo della pista.

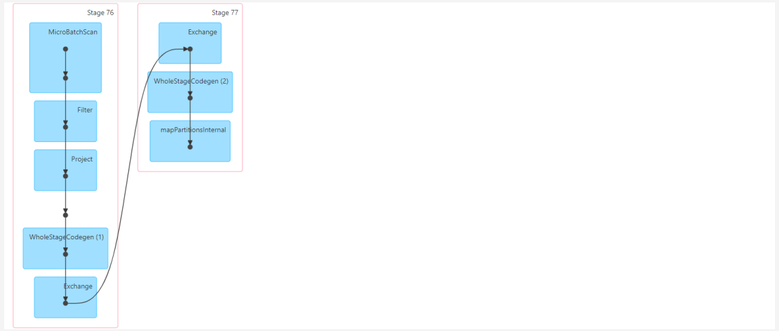
Kibana

Si rifà a Elasticsearch e mostra una dashboard live con i seguenti grafici:

- Telemetria passo gara;
- Ultimo giro per pilota;
- Giro di gara in corso;
- Previsione prossimo giro per pilota;
- Distacchi dal leader e dal pilota che precede;
- Ultimi dati meteo della pista.

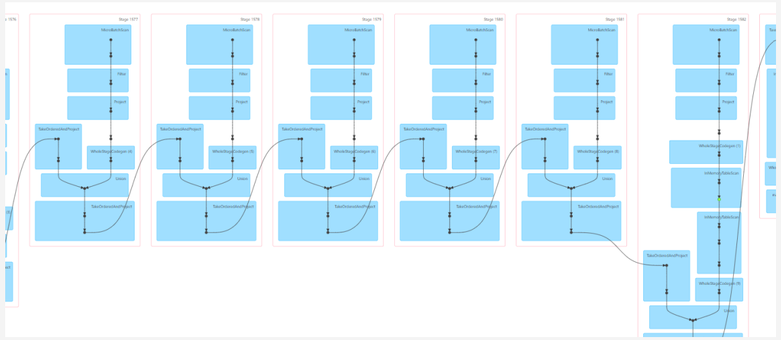
Problemi di sviluppo e soluzioni

Problema ottimizzazione Spark Structured Streaming



DAG al giro 2 di un pilota

All'inizio il calcolo e le operazioni di shuffle sono istantanee, il grafico si presenta normale.



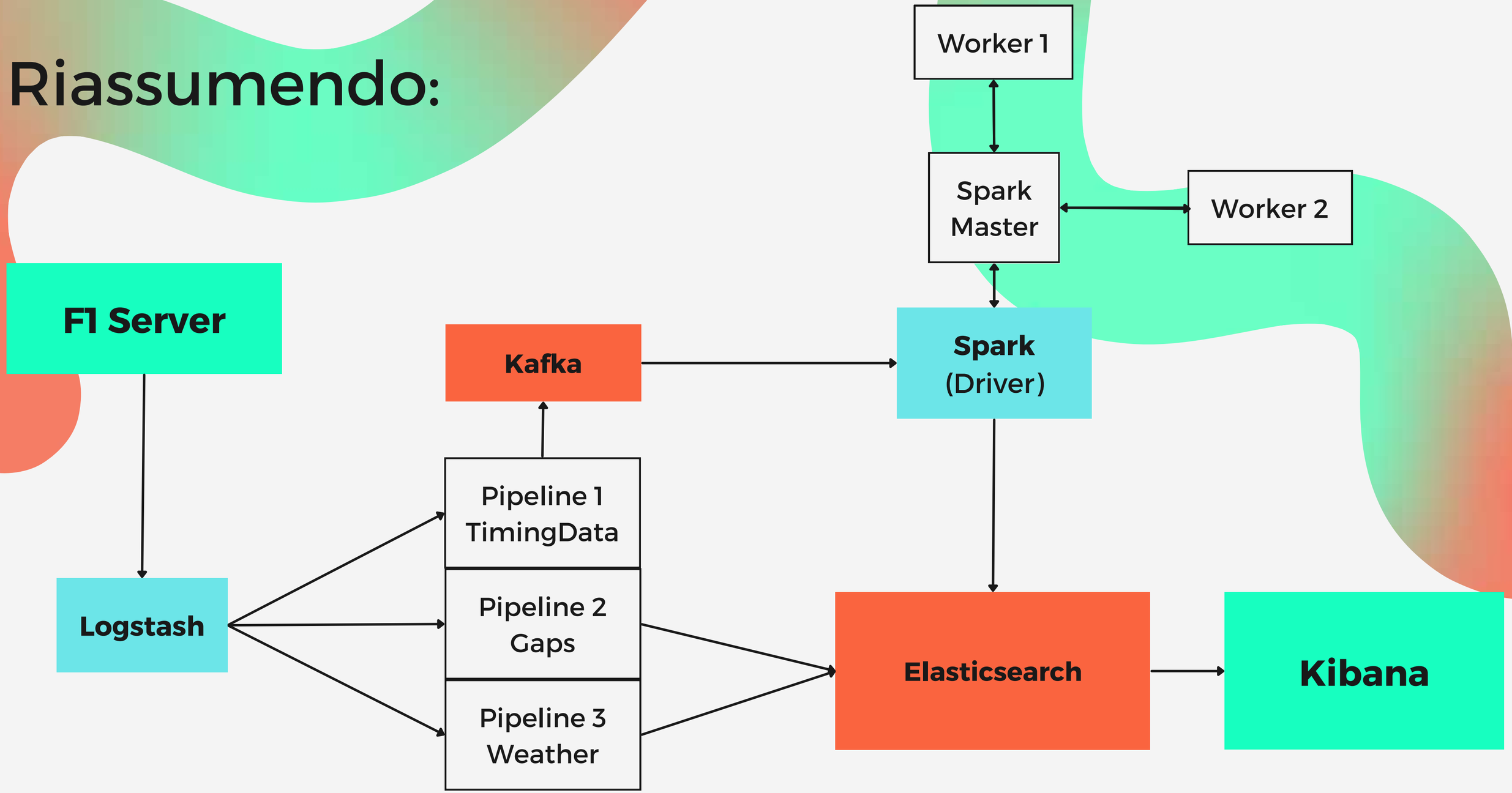
DAG al giro 10

Si presenta con piu di 10 fasi, perchè Structured Streaming le ripete tutte se vi è una operazione di shuffle che li lega (union). Ciò rallenta l'elaborazione ad ogni giro, occupando molta RAM.

DAG al giro 40 con codice ottimizzato

Risoluzione rallentamento slegando i microbatch tra di loro, gestione dello storico e shuffle affidata a Dataframe pandas. Elaborazione instantanea a qualsiasi giro e poca RAM usata.

Riassumendo:



Avvio dei container

Posizionarsi sulla cartella del compose e digitare:

docker compose up

Attendi l'avvio dei container

Per visualizzare la dashboard vai su:

http://localhost:5601