Analisi di dati energetici con Apache Spark

Gianluca Ronzello
Università degli Studi di Roma Tor Vergata
gianlucaronzello@gmail.com

***Abstract*—In questo report viene descritta la pipeline e l'implementazione di due query in Apache Spark su dati storici forniti da Electricity Maps sull'elettricità e sulle emissioni di CO2 per produrla.**

I. Introduzione

Il progetto riguardo lo svolgimento di due query tramite batch processing per l'analisi di dati energetici con Apache Spark. I dati vengono presi da Electricity Maps, lo studio si incentra sui dataset di Italia e Svezia nel intervallo di tempo 2021-2024, presi con granulità oraria.

II. Architettura pipeline

La pipeline (Figura 1) si articola in:

1) Vengono scaricati i dati da Electricity Maps tramite Nifi, aggregati in base al paese e caricati su HDFS
2) Si esegue batch processing su Apache Spark (tramite API RDD e Dataframe)
3) Si salvano i risultati su MongoDB e HDFS
4) Si caricano i dati da MongoDB su Grafana per creare dashboard interattive

Ogni componente dell'architettura è eseguito all'interno di un container Docker, con Docker Compose utilizzato come strumento di orchestrazione. Il sistema HDFS è composto da un NameNode e tre DataNode, mentre l'infrastruttura Spark prevede un master e tre worker.

Per la creazione dei container Docker vengono impiegate le immagini ufficiali, ad eccezione di quella relativa ad HDFS, per cui si è optato per un'immagine preconfigurata fornita dal professor Nardelli. È stato inoltre aggiunto un container dedicato alla gestione di MongoDB tramite una Web UI, accessibile sulla porta 8081.

Le interfacce web degli altri servizi sono esposte sulle porte predefinite: 9870 per l'HDFS NameNode, 8080 e 4040 per Spark, 8443 per NiFi e 3000 per Grafana.

III. Data ingestion

Per la fase di acquisizione dei dati è stato scelto Apache NiFi. Il flusso, illustrato nelle figure 2 e 3, scarica i dataset dal sito di Electricity Maps tramite richieste HTTP, aggrega i dati per paese e rimuove intestazioni ridondanti. I dati vengono successivamente convertiti nei formati CSV, Avro e Parquet, e salvati su HDFS.

Il flusso NiFi si compone dei seguenti processor:

- GenerateFlowFile: utilizzato per definire gli URL da cui scaricare i dati; sono stati impiegati due processor distinti, uno per ciascun paese considerato.
- UpdateAttribute: impiegato per assegnare un identificativo di gruppo basato sul paese, al fine di uniformare i flussi successivi.
- SplitJson e EvaluateJsonPath: utilizzati per estrarre correttamente gli URL dai dati JSON.
- InvokeHTTP: responsabile dell'invio delle richieste HTTP al sito di Electricity Maps per il recupero dei dati.
- MergeContent: aggrega i dataset suddivisi per anno in un unico file per ciascun paese.
- QueryRecord: consente di filtrare i dati ed eliminare elementi superflui risultanti dalla fusione dei dataset. La query utilizzata è SELECT * FROM FLOWFILE WHERE Country <> 'Country'
- ConvertRecord: converte i dati dal formato CSV ai formati Avro e Parquet.
- UpdateAttribute (aggiuntivi): utilizzati per assegnare nomi ai file generati, in base al formato di output.
- PutHDFS: salva i file finali all'interno del file system distribuito HDFS.

IV. Pre-Processing

Una volta acquisiti i dataset da HDFS, Spark esegue una fase di preprocessamento finalizzata all'ottimizzazione delle analisi successive. In questa fase vengono selezionate solo le colonne rilevanti per rispondere alle query di interesse, ed è effettuata una trasformazione della colonna "Datetime (UTC)", da cui vengono estratti e separati l'anno e il mese in due nuove colonne denominate "year" e "month". Il dataset risultante assume la forma riportata in Table1

V. Query

A. Query 1

La query 1 richiede di aggregare i dati su base annua, poi calcolare media, minimo e massimo per "Carbon intensity gCO2 eq/kWh (direct)" e "Carbon-free energy percentage (CFE%)" per ciascun anno.

1) RDD:

1) Si mappa ogni riga in una tupla chiave-valore con:
   - chiave = (country, year)
   - Valore = (carbon, cfe, carbon, cfe, carbon, cfe, 1)
2) Si esegue una reduceByKey, ottenendo una tupla: ((country, year),(min carbon, min cfe, max carbon, max cfe, somma carbon, somma cfe, count)

3) Si calcolano le medie utilizzando map, dividendo le somme per count

Il DAG generato si vede in figura 4

2) DataFrame:

1) Si raggruppa per (”country”,”year”) tramite goupBy
2) Si esegue un aggregazione, tramite agg, con funzione di minimo, massimo e media per ”carbon intensity” e ”cfe”
3) Si ordina tramite orderBy per (”country”,”year”)

B. Query 2

La Query 2, una volta aggregati i dati per la coppia (anno, mese) e calcolato la media per ”carbon intensity” e ”cfe”, richiedeva di calcolare la classifica delle prime 5 coppie per entrambe le metriche sia in ordine crescente che decrescente.

1) RDD:

1) Si mappa ogni riga in una tupla chiave-valore con:
   - chiave = (anno, mese)
   - Valore = (carbon, cfe, count)
2) Si utilizza una reduceByKey per aggregare per somma
3) Si utilizza la funzione map per calcolare la media
4) Si creano le classifiche usando la funzione sortBy e se ne prendono 5 elementi usando la funzione take

Il DAG generato si vede in figura 5

2) DataFrame:

1) Si raggruppa tramite groupBy per (anno, mese)
2) Si aggrega tramite la funzione media
3) Si creano le classifiche tramite la funzione orderBy e mettendo il limite a 5 elementi

VI. Salvataggio risultati

L’applicazione salva il risultato delle query su HDFS e sul data store NoSQL MongoDB. Il salvataggio su HDFS avviene nel formato CSV, con una copia salvata sul filesystem locale. Per quanto riguarda MongoDB la scrittura dei risultati viene effettuata da Spark tramite il connettore ufficiale sul database spark. Ogni query viene salvata nella propria collezione in cui ogni documento rappresenta una riga del CSV di output. Per caricare il connettore di MongoDB, è stato aggiunto il pacchetto org.mongodb.spark:mongo-spark-connector come package aggiuntivo del comando spark-submit

VII. Grafana

Per visualizzare i grafici richiesti dalle query si utilizza Grafana. Grafana è stato connesso a MongoDB tramite un plugin sviluppato dalla community. Una volta connessi sono state create 2 dashboard, una per query. La configurazione dei datasource e delle dashboard avviene in maniera automatica tramite il sistema di provisioning fornito da Grafana. Figura 6-7

VIII. Performance

Per le performance si è deciso di confrontare le performance dei differenti formati csv, avro, parquet. Questo è stato fatto sia per caricamento dati e preprocessamento (figura 8-9), che per il processamento sia con RDD che DataFrame (figura 10-11). Per i tempi di caricamento come evidenziato nel grafico il formato avro è il più veloce, mentre per i tempi di esecuzione non ci sono grandi discrepanze ne tra i vari formati ne cambiando da dataframe a RDD.

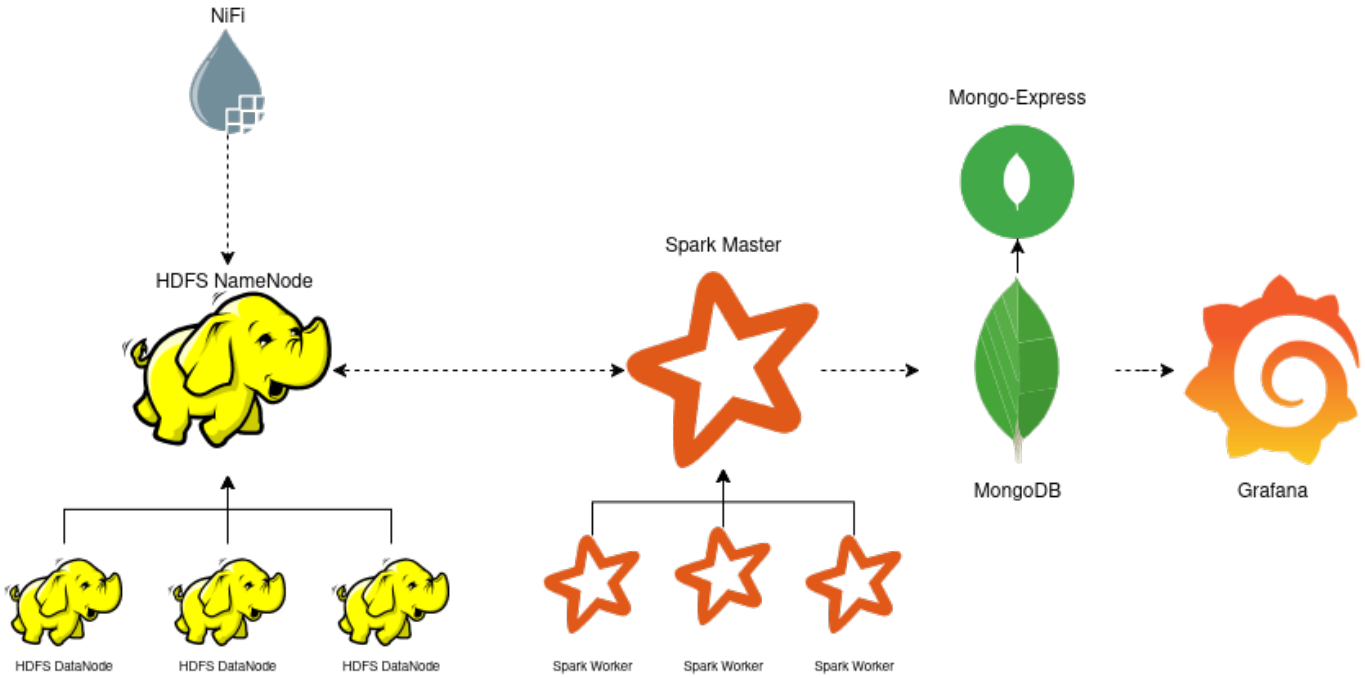

Fig. 1. Pipeline

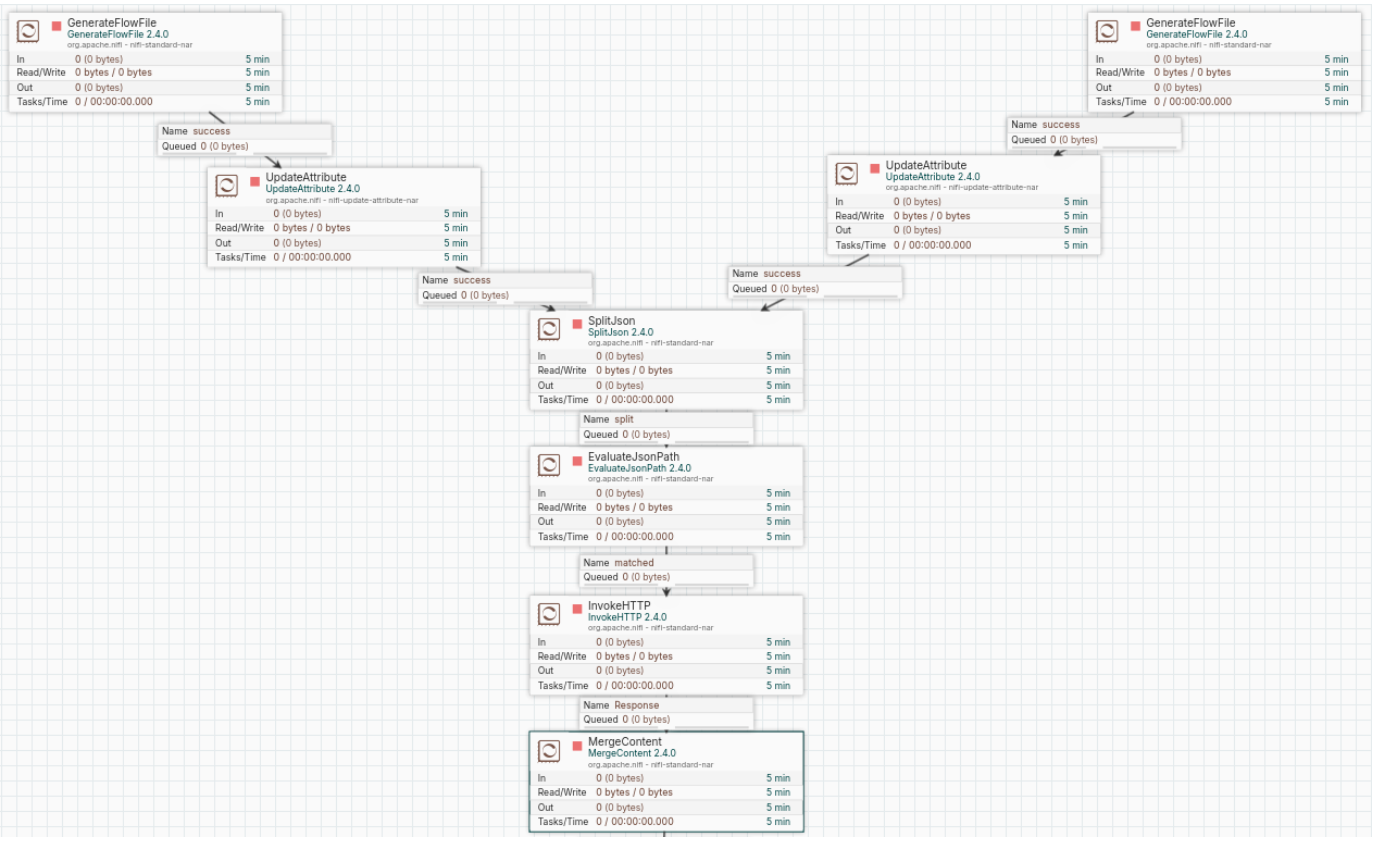

Fig. 2. flow parte 1

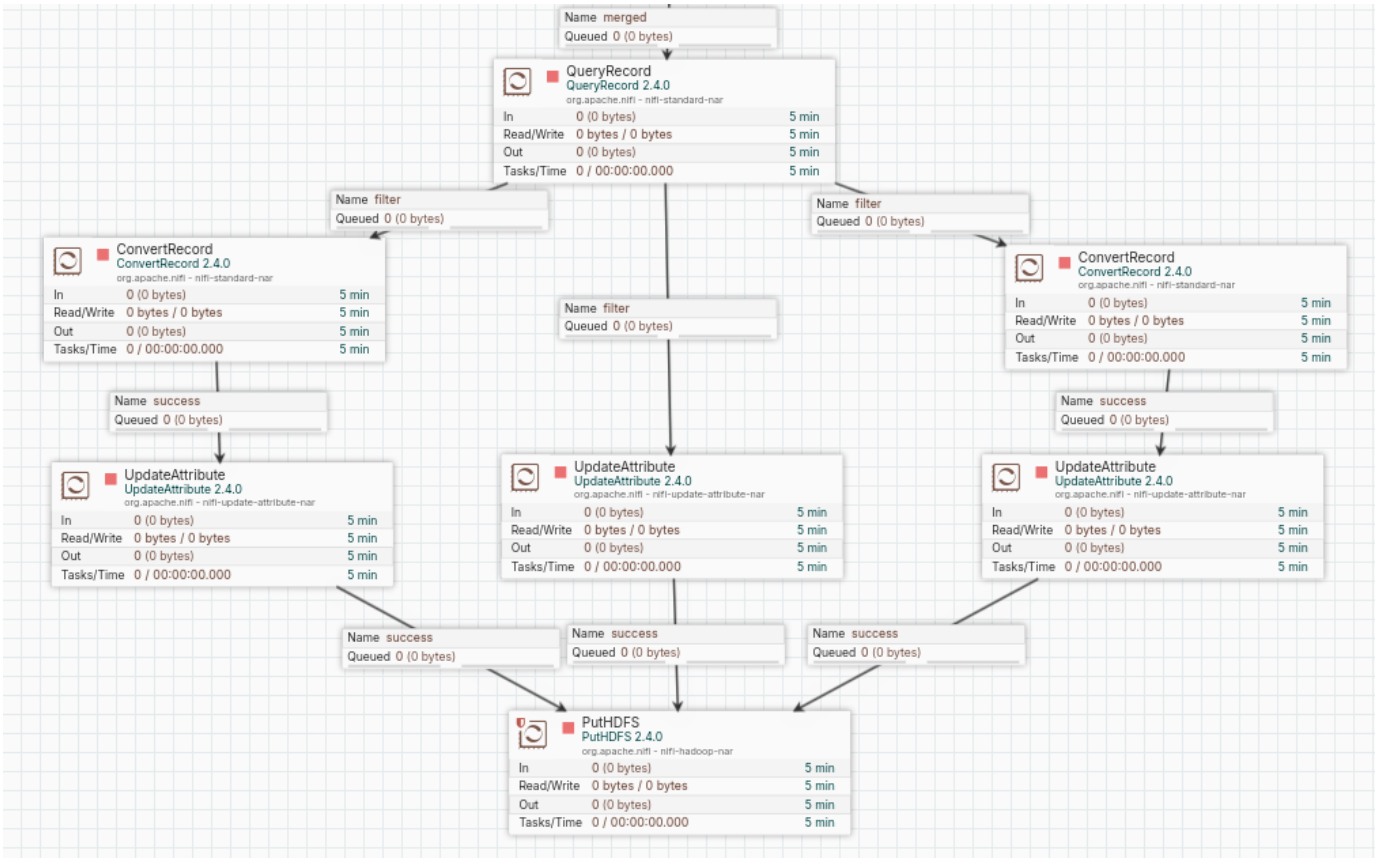

Fig. 3. flow parte 2

Datetime (UTC)	Country	Carbon intensity gCO$_2$eq/kWh (direct)	Carbon-free energy percentage (CFE%)	year	month

TABLE I
STRUTTURA DEL DATASET DOPO IL PREPROCESSAMENTO IN SPARK

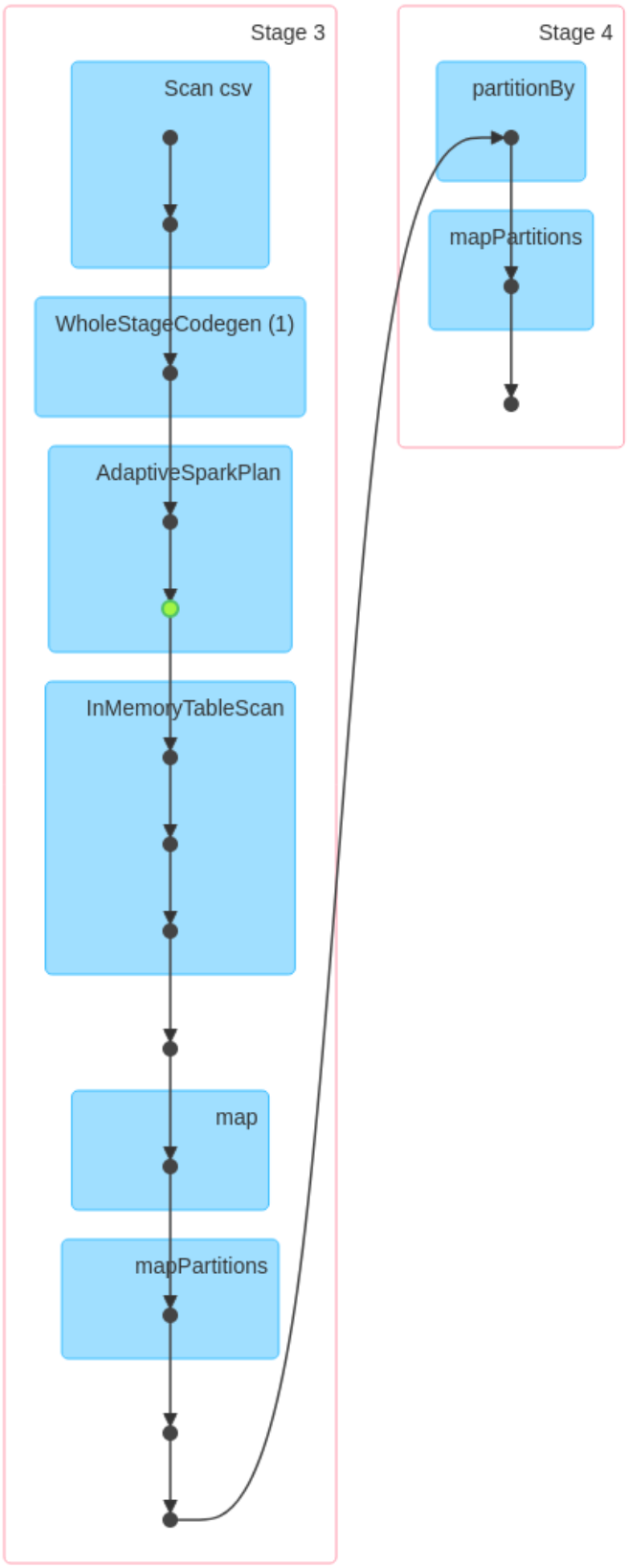

Fig. 4. DAG query 1

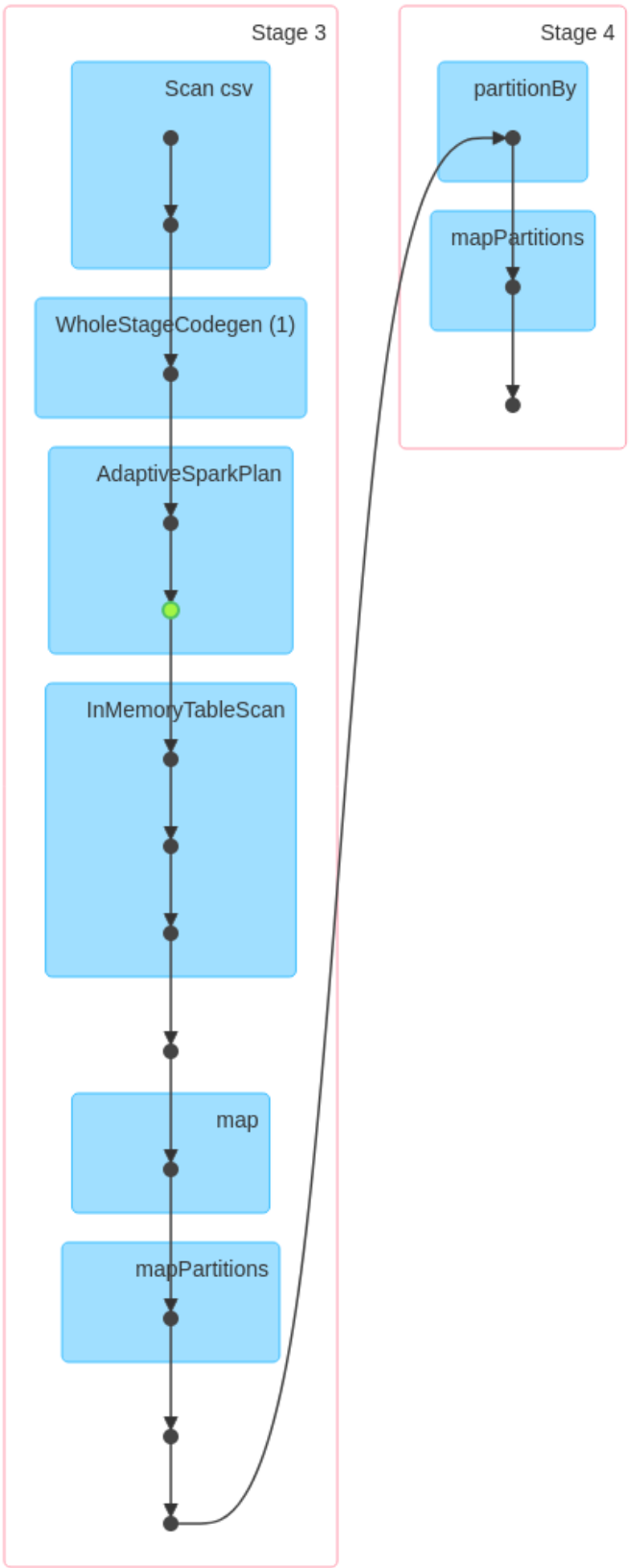

Fig. 5. DAG query 2

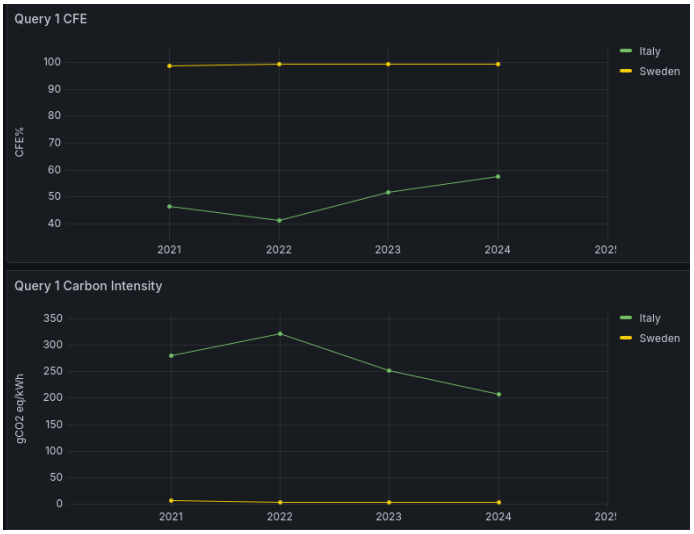

Fig. 6. Grafici query 1

Fig. 7. Grafici query 2

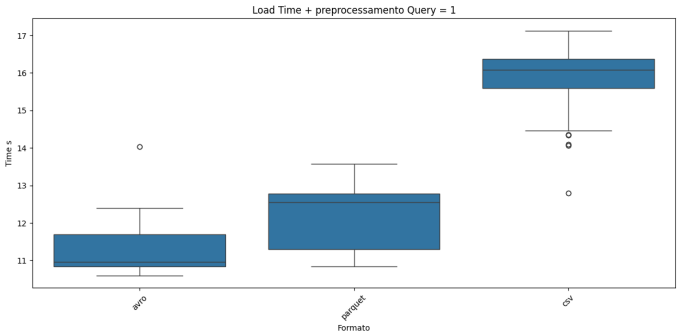

Fig. 8. Tempi caricamento e pre-processamento query 1

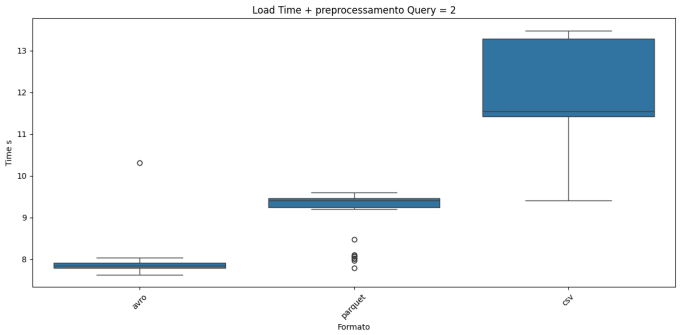

Fig. 9. Tempi caricamento e pre-processamento query 2

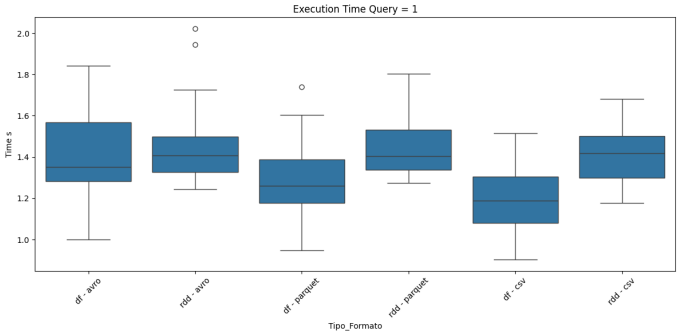

Fig. 10. Tempi esecuzione query 1

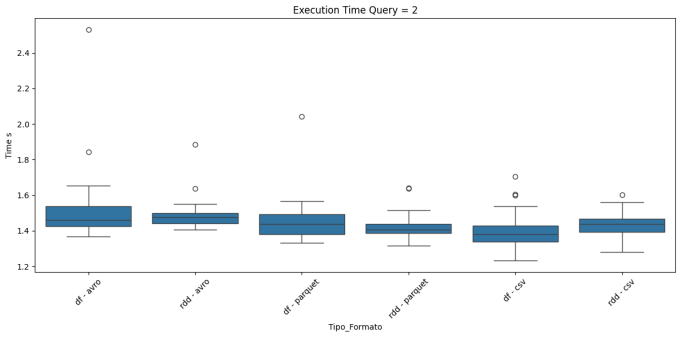

Fig. 11. Tempi esecuzione query 2