TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Macroarea di Ingegneria
Dipartimento di Ingegneria Civile e Ingegneria Informatica

SABD Project #1

A comparative Analysis of Carbon Intensity and Carbon-Free Energy in Italy and Sweden (2021- 2024) Using Apache Spark

Corso di Sistemi e Architetture per Big Data
Laurea Magistrale in Ingegneria Informatica

A.A. 2024/25
Matteo Basili, Adriano Trani

Outline

- Goals
- Introduction & Background
- Overview
- Data Acquisition and Ingestion (NiFi)
- Implementation of Queries (Spark)
- Export of the Results (HDFS → Redis)
- Charts Creation (Grafana)
- Results, Performance Analysis and Discussion
- Demo

Goals

- Analyze carbon intensity and carbon-free energy share in Italy and Sweden (2021–2024) using ElectricityMaps data
- Develop a distributed Big Data pipeline to:
  - Retrieve, process, and analyze data
  - Answer specific queries
  - Export results and generate charts
- Measure execution time for each query in a controlled environment
- Present and discuss the results of the analysis

Queries

- **Q1** – Yearly Aggregation by Country
  - Compute the average, minimum, and maximum values of:
    - Carbon intensity
    - Carbon-free energy share (CFE%)
  - Comparison between Italy and Sweden with visual plots
- **Q2** – Monthly Aggregation for Italy
  - Identify the top 5 and bottom 5 months for each metric (carbon intensity and CFE%)
  - Display monthly trends through visualizations
- **Q3** – 24-Hour Aggregation
  - For both countries, compute:
    - Minimum, 25th percentile, median (50th), 75th percentile, and maximum for carbon intensity and CFE%
  - Show hourly comparisons using charts

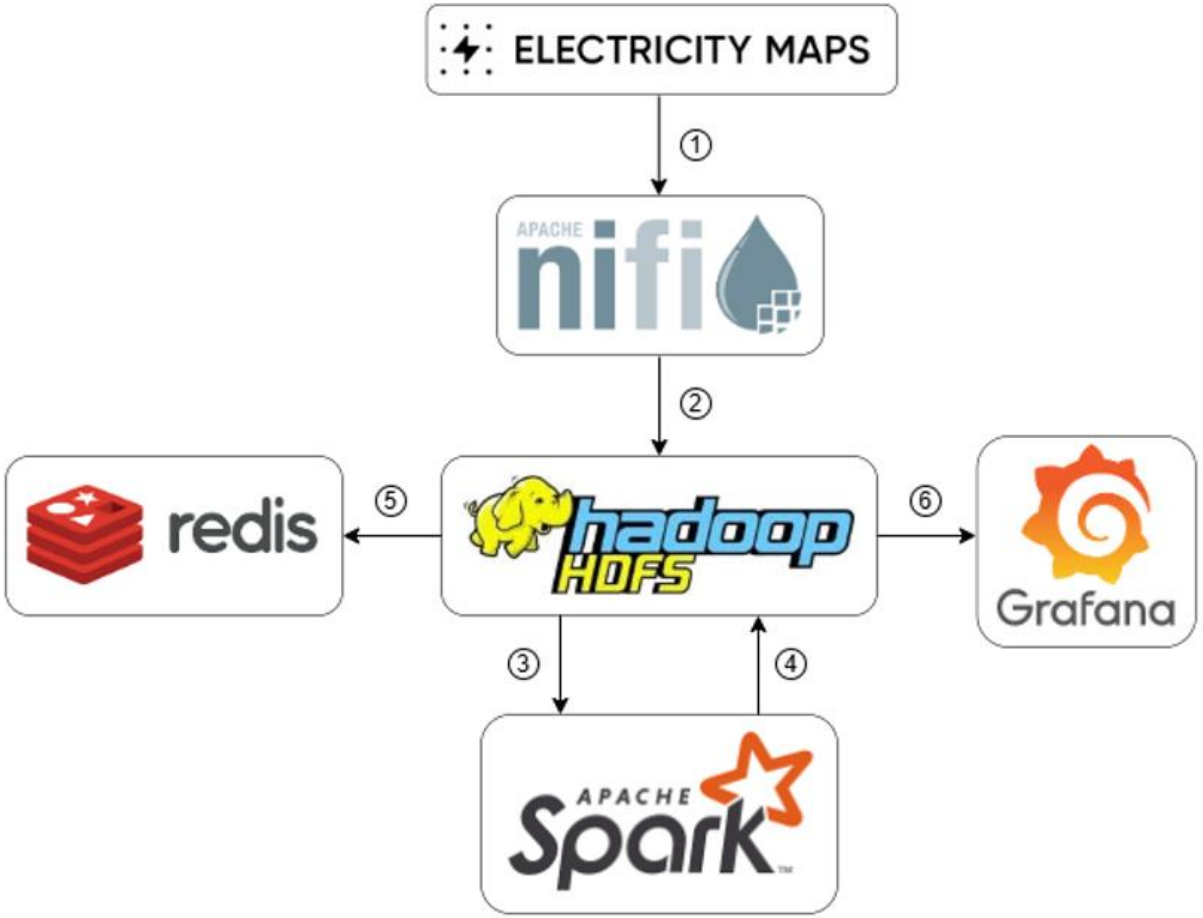
Introduction & Background

- The term *carbon intensity* describes the amount of greenhouse gases emitted per unit of electricity load
  - It's expressed as grams of CO_2 equivalent per kilowatt hour (gCO_2eq/kWh)
- The term *carbon-free energy* represents the percentage of electricity available on a grid from low or zero carbon emission sources
  - Wind
  - Hydro
  - Geothermal
  - Biomass
  - Nuclear energy

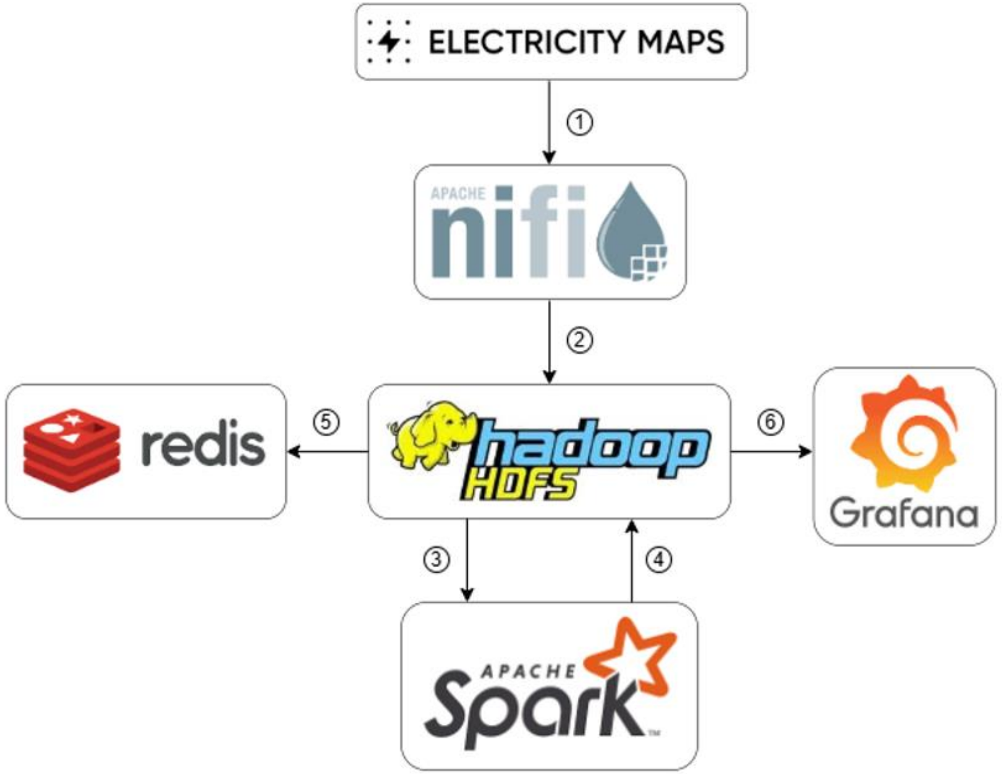
Overview - Technology Stack

- Docker & Docker Compose: Containerization of the entire system
- Apache Spark: Processing the data
- HDFS: Distributed data storage
- Apache NiFi: Data acquisition and ingestion
- Redis: Exporting the results of the queries
- Grafana: Charts maker
- Python: Main programming language

Overview - Architecture



Overview - DataFlow

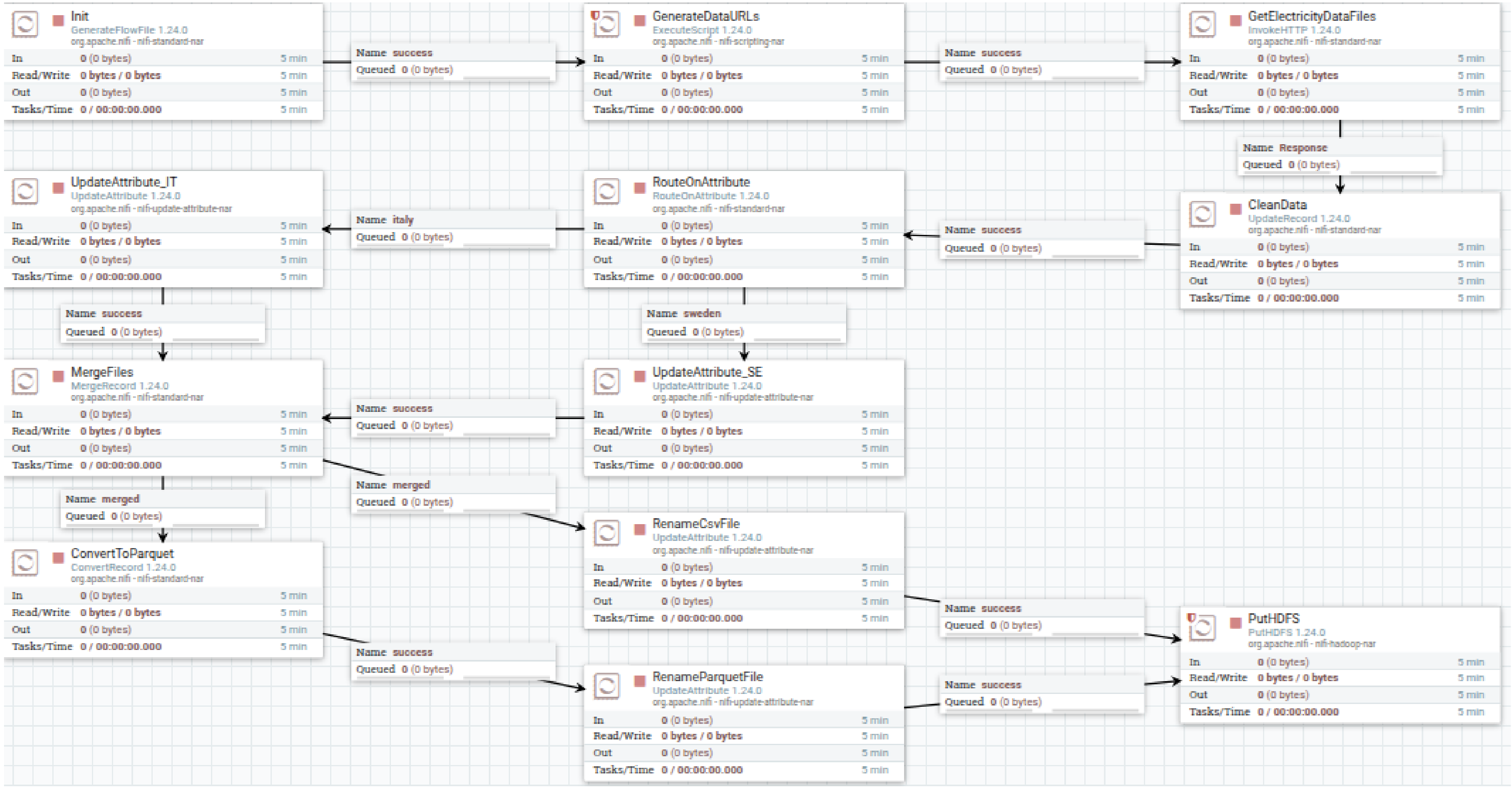


1) Hourly CSV files are retrieved from Electricity Maps via API
2) NiFi preprocesses the data and loads both CSV and Parquet formats into HDFS
3) Spark reads the data from HDFS and executes processing
4) Query results are written back to HDFS in CSV format
5) Aggregated results are exported to Redis for fast access
6) Grafana reads the data (CSV format) and creates charts

Overview - Docker Containers

- The system is fully containerized with Docker and orchestrated using Docker Compose
  - HDFS: namenode, datanode1, datanode2
  - NiFi: nifi
  - Spark: spark-master, spark-worker-1, spark-worker-2
  - Redis: redis, results_exporter
  - Grafana: grafana, grafana-image-render

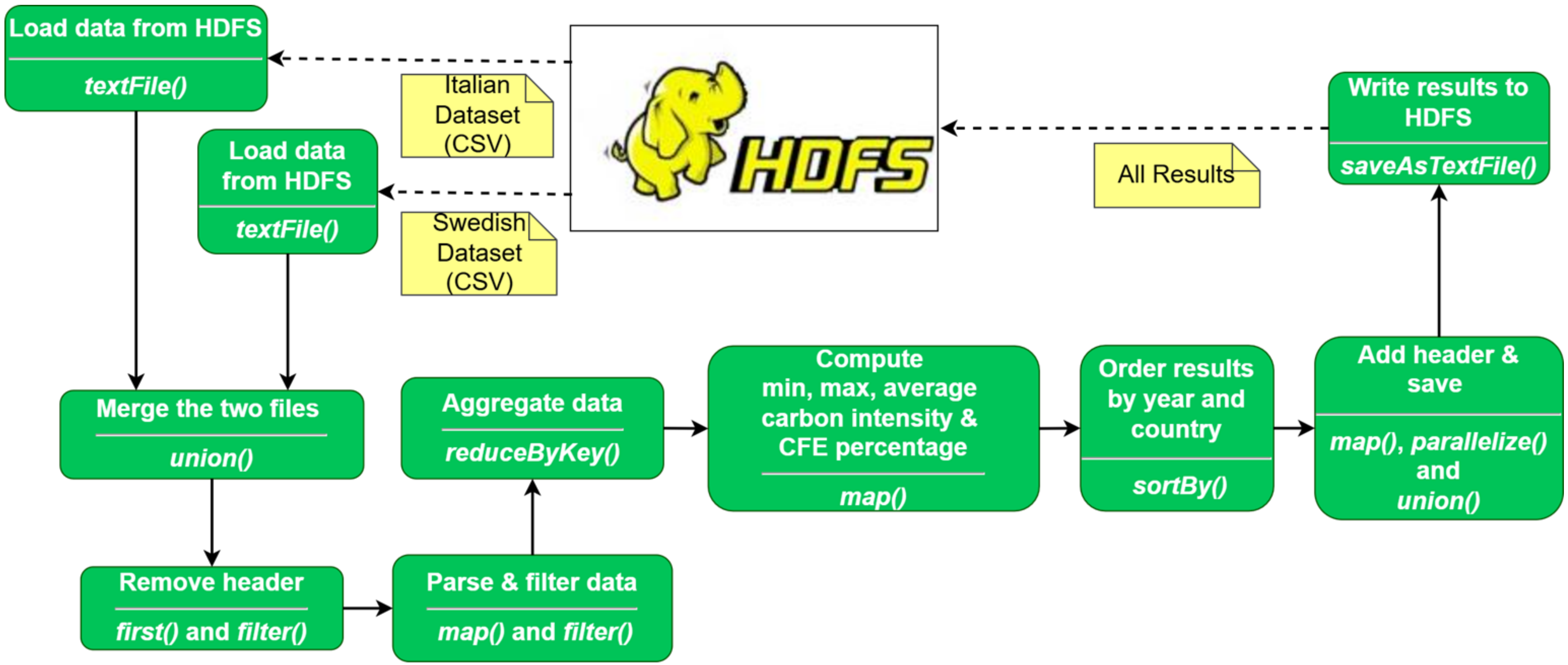
NiFi - Data Acquisition and Ingestion



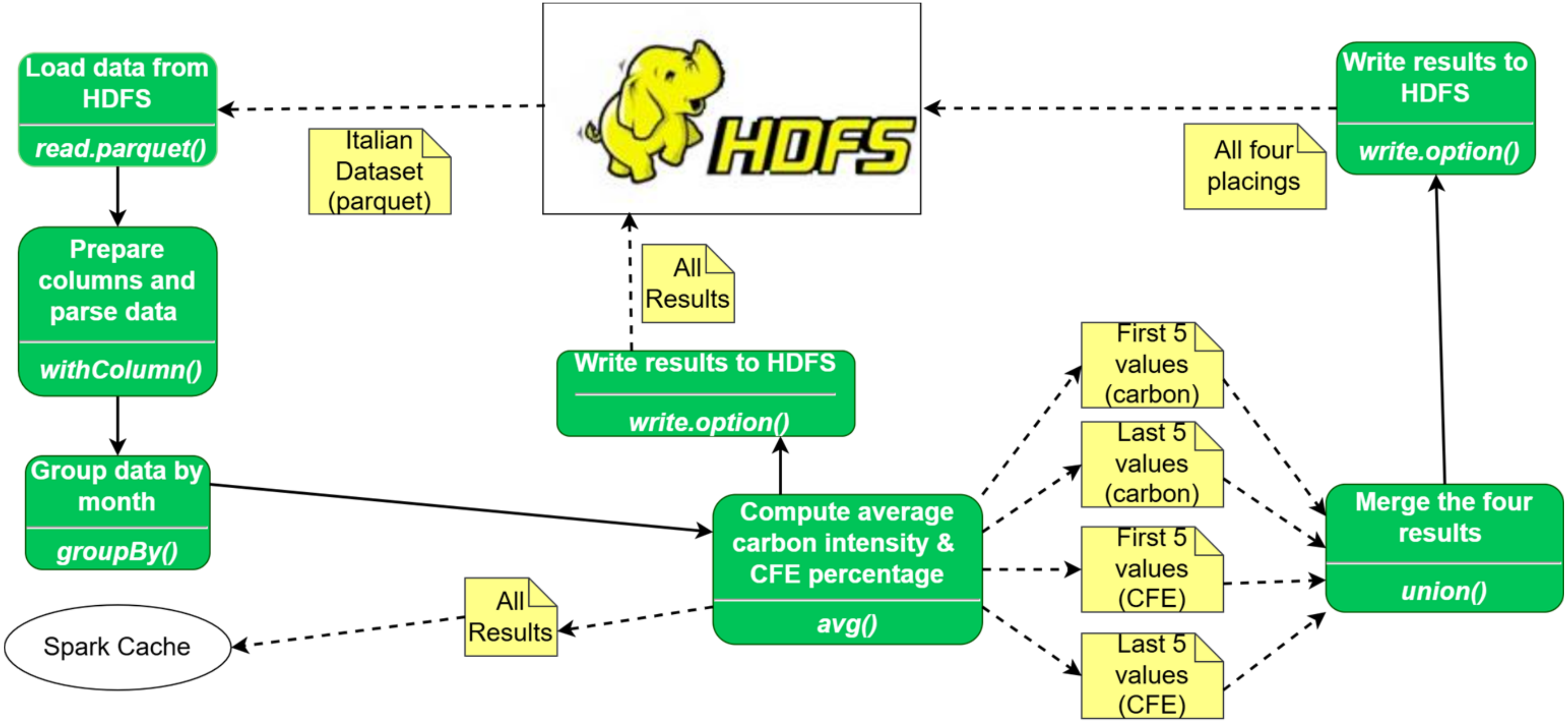
Implementation

- The analytical queries have been implemented using three different Spark programming interfaces
  - RDD API
    - CSV input format
  - DataFrame API
    - Parquet input format
  - SQL
    - Parquet input format

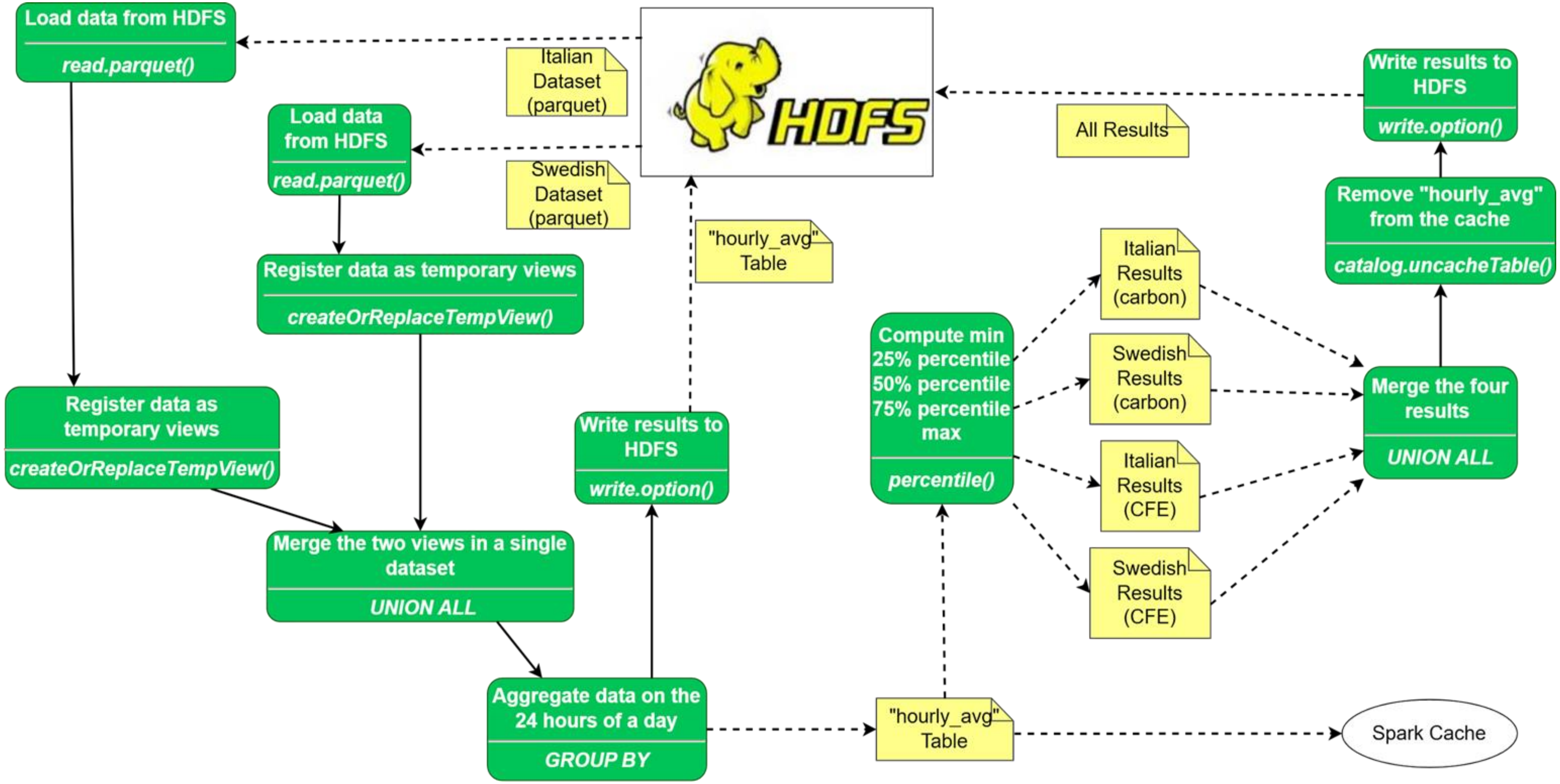
Implementation – Q1 (RDD Version)



Implementation – Q2 (DataFrame version)



Implementation – Q3 (SQL version)



Export – HDFS & Redis

- Results are written directly to HDFS
  - Each output is saved under the */output/* directory with a unique timestamp to distinguish different runs
  - The data is written as a CSV file that includes a header row
  - Spark automatically handles partitioning internally
- Output files are filtered and read with streaming
- Every line is parsed
  - Headers ignored, rows splitted, fields validated, keys builded
    - Q1 - key: *Q1:<country>:<year>*
    - Q2 - key: *Q2:IT<order_type>:<rank>*
    - Q3 - key: *Q3:<country>:<metric>*
  - Use a non-transactional pipeline for batch writing to Redis
- ***export_q*_hdfs_to_redis.py*** scripts

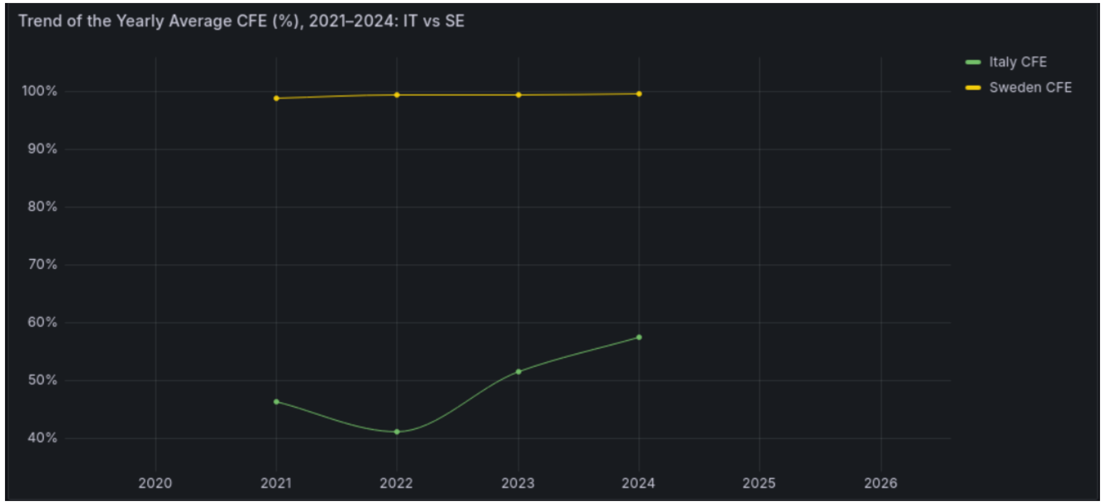
Grafana

- ***create_q*_plots.py*** scripts
  - Create the CSV Data Source
    - Grafana API and marcusolsson-csv-datasource plugin
  - Create the Dashboard and Panel using the API
  - Save the dashboard using Selenium
  - Use Grafana's render service to export panel images as PNGs
    - Save them in the *Results/images* directory

Results

- For example

year	country	carbon-mean	carbon-min	carbon-max	cfe-mean	cfe-min	cfe-max
2021	IT	280.084245	121.24	439.06	46.305932	15.41	77.02
2022	IT	321.617976	121.38	447.33	41.244127	13.93	77.44
2023	IT	251.819465	74.44	429.93	51.596057	20.39	85.02
2024	IT	207.299189	50.18	345.65	57.431828	20.9	90.26
2021	SE	5.946325	1.5	55.07	98.962411	92.8	99.65
2022	SE	3.875823	0.54	50.58	99.551723	94.16	99.97
2023	SE	3.903308	0.31	51.57	99.525599	94.12	99.98
2024	SE	3.726149	0.25	45.61	99.557928	94.27	99.98



Performance Analysis – Setup

- Spark Cluster (default) Configuration:
  - Executors: 2 (1 per worker node)
  - Cores per Executor: 2 (total 4 vCPUs)
  - Memory: ~1 GB per executor and driver
  - Partitioning: Based on HDFS block size (128 MB)
  - Deploy Mode: Client mode (driver on spark-master)

Performance Analysis – Execution Logic

- ***run_query_isolated.py*** script
  - 10 runs
    - Ensures cold-start conditions (no cache interference)
  - Execution time measured before and after job
  - Compute mean & standard deviation
  - Results saved to *Results/analysis/* folder

Performance Analysis – Results

Q1	Average	Standard Deviation
RDD	42.12 seconds	3.05 seconds
DataFrame	52.45 seconds	11.07 seconds
SQL	49.04 seconds	0.73 seconds

Q2	Average	Standard Deviation
RDD	44.04 seconds	0.59 seconds
DataFrame	71.80 seconds	1.12 seconds
SQL	52.90 seconds	1.43 seconds

Q3	Average	Standard Deviation
RDD	47.93 seconds	1.13 seconds
DataFrame	93.35 seconds	1.60 seconds
SQL	79.46 seconds	0.81 seconds

Discussion – Query Results

- Annual Trend (Q1)
  - Italy: Between 2022 and 2024, –35% carbon intensity (~322 → 207 gCO_2eq/kWh); CFE ↑ from 41% to 57%
    - Driven by renewables, hydropower recovery, low-carbon imports
  - Sweden: Stable near-zero emissions (3–6 gCO_2eq/kWh) and ~99% CFE
    - Consistent clean mix: hydro, nuclear, wind
- Monthly Variability – Italy (Q2)
  - Peak: Dec 2022 (~360 gCO_2eq/kWh), Best: May 2024 (~158 gCO_2eq/kWh)
  - CFE % inversely correlated with emissions
  - 2024 shows structural progress
- Hourly Distribution (Q3)
  - Italy: Emissions ↓ at midday (~220 gCO_2eq/kWh), ↑ at night (~297 gCO_2eq/kWh); CFE peaks at ~57% (day), drops to ~42% (night)
  - Sweden: Flat profile <6 gCO_2eq/kWh & ~99.5% CFE → steady supply

Discussion – Performance Analysis

- RDDs with CSV perform better than Dataframe/SQL with Parquet
  - Initial Overhead: DataFrame + Parquet
    - DataFrames trigger Catalyst optimizer:
      - Logical plan → Physical plan → Extra planning time
    - Parquet needs:
      - Metadata parsing, schema resolution, deserialization
  - RDD + CSV is direct and low-level
    - RDDs skip Catalyst, schema parsing, and logical planning
    - CSV is simple to parse (line-by-line, no metadata)
  - Cold Start impacts DataFrame + Parquet more
    - Spark must initialize driver, executors, and plan queries
    - Parquet files need to be scanned and interpreted

References

- https://github.com/MatteoBasili/sabd-progetto1-2024_25

Demo

- Open terminal
- Enter the root directory of the project
- Launch containers
  - $ docker compose up -d
- Execute the full pipeline for one of the queries
  - $ python3 ./scripts/run_full_pipeline.py [q1|q2|q3] [rdd|df|sql]
- Check the results (tables and/or images) in the directory Results