# Case Study 4: Optimized Real-Time Data Processing with Caching, Persisting, and Broadcasting

## Problem Statement

The goal is to build a high-performance data pipeline for processing, aggregating, and storing sales data using the Walmart Recruiting Dataset. The focus is on optimizing the performance of Spark jobs by leveraging caching, persisting, and broadcasting while handling large-scale data efficiently. The pipeline should support data validation, enrichment with metadata, and efficient storage in Google Cloud Storage (GCS) using JSON and Parquet formats. Additionally, caching should be strategically used to improve performance, especially when data is accessed repeatedly across different stages of the pipeline.

## Project Structure

The project is organized as follows:

```
Case Study 4/
├── SalesRecord/
│   ├── SalesRecord.scala
│   └── SalesRecordProto.scala
├── StaticDataProcessing.scala
└── README.md
```

## Tasks

### 1. Data Preparation

Load and preprocess the Walmart Recruiting Dataset. The dataset includes:

- `train.csv`: Contains historical weekly sales data for various stores and departments.

- `features.csv`: Contains additional data about stores and regions.
- `stores.csv`: Contains store metadata.

## 2. Data Validation and Enrichment

Validate the sales data and enrich it with metadata by performing the following steps:

- Ensure no missing or invalid values in critical columns.
- Filter out records where `Weekly_Sales` is negative.
- Perform joins with `features.csv` and `stores.csv` on relevant keys.

**Caching Scenario:** Cache the `features.csv` and `stores.csv` datasets after cleaning, as they will be used repeatedly in multiple join operations.

## 3. Aggregation

Compute key metrics to analyze sales performance:

- **Store-Level Metrics:** Total weekly sales, average weekly sales, and top-performing stores.
- **Department-Level Metrics:** Total sales, weekly trends, and holiday vs. non-holiday sales.

**Caching Scenario:** Cache intermediate aggregation results (e.g., store-level or department-level metrics) to avoid recomputation when deriving further insights.

## 4. Storage Optimization

Optimize the storage of processed and aggregated data by:

- Storing enriched datasets in Parquet format for compact storage and efficient querying.
- Partitioning the data by `Store` and `Date`.
- Saving aggregated metrics in JSON format for lightweight storage.

**Caching Scenario:** Cache the partitioned enriched dataset before writing to storage to ensure the partitioning logic is reused in downstream processes.

### 5. Real-Time Simulation

Simulate real-time ingestion and updates of weekly sales data by:

- Ingesting weekly sales updates from a Kafka topic.
- Updating aggregated metrics in real time to reflect new data.

**Persisting Scenario:** Persist the streaming results to ensure fault tolerance in case of node failures.

### 6. Performance Optimization

Optimize the performance of the pipeline by:

- Using caching for intermediate datasets that are accessed repeatedly.
- Using broadcasting for small, static datasets like `stores.csv`.
- Persisting aggregated metrics with a suitable storage level for downstream access.

## Code Overview

### `SalesRecord/SalesRecord.scala`

This file contains the case class and companion object for the `SalesRecord` protocol buffer message. It includes methods for parsing and serializing the message, as well as lenses for accessing and modifying fields.

### `SalesRecord/SalesRecordProto.scala`

This file contains the generated code for the `SalesRecord` protocol buffer message. It includes the Scala and Java descriptors for the message, as well as methods for accessing the descriptors.

### `StaticDataProcessing.scala`

This file contains the main Spark application for processing the sales data. It includes the following steps:

- Initializing the Spark session with GCS configurations.
- Loading the datasets from GCS.

- Validating and enriching the data.
- Computing store-level and department-level metrics.
- Storing the enriched data and aggregated metrics in GCS.
- Simulating real-time ingestion and updates of weekly sales data.
- Optimizing the performance of the pipeline using caching, persisting, and broadcasting.

## Steps and Results:

### Load the features dataset:

| Store | Date | Temperature | Fuel_Price | MarkDown1 | MarkDown2 | MarkDown3 | MarkDown4 | MarkDown5 | CPI | Unemployment | IsHoliday |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2010-02-05 | 42.31 | 2.572 | NA | NA | NA | NA | NA | 211.0963582 | 8.106 | false |
| 1 | 2010-02-12 | 38.51 | 2.548 | NA | NA | NA | NA | NA | 211.2421698 | 8.106 | true |
| 1 | 2010-02-19 | 39.93 | 2.514 | NA | NA | NA | NA | NA | 211.2891429 | 8.106 | false |
| 1 | 2010-02-26 | 46.63 | 2.561 | NA | NA | NA | NA | NA | 211.3196429 | 8.106 | false |
| 1 | 2010-03-05 | 46.5 | 2.625 | NA | NA | NA | NA | NA | 211.3501429 | 8.106 | false |
| 1 | 2010-03-12 | 57.79 | 2.667 | NA | NA | NA | NA | NA | 211.3806429 | 8.106 | false |
| 1 | 2010-03-19 | 54.58 | 2.72 | NA | NA | NA | NA | NA | 211.215635 | 8.106 | false |
| 1 | 2010-03-26 | 51.45 | 2.732 | NA | NA | NA | NA | NA | 211.0180424 | 8.106 | false |
| 1 | 2010-04-02 | 62.27 | 2.719 | NA | NA | NA | NA | NA | 210.8204499 | 7.808 | false |
| 1 | 2010-04-09 | 65.86 | 2.77 | NA | NA | NA | NA | NA | 210.6228574 | 7.808 | false |

only showing top 10 rows

### Load the train dataset:

```
+-----+----+----------+------------+---------+
|Store|Dept|      Date|Weekly_Sales|IsHoliday|
+-----+----+----------+------------+---------+
|    1|   1|2010-02-05|     24924.5|    false|
|    1|   1|2010-02-12|    46039.49|     true|
|    1|   1|2010-02-19|    41595.55|    false|
|    1|   1|2010-02-26|    19403.54|    false|
|    1|   1|2010-03-05|     21827.9|    false|
|    1|   1|2010-03-12|    21043.39|    false|
|    1|   1|2010-03-19|    22136.64|    false|
|    1|   1|2010-03-26|    26229.21|    false|
|    1|   1|2010-04-02|    57258.43|    false|
|    1|   1|2010-04-09|    42960.91|    false|
+-----+----+----------+------------+---------+
only showing top 10 rows
```

### Load the Stores Dataset:

```
+-----+----+------+
|Store|Type|  Size|
+-----+----+------+
|    1|   A|151315|
|    2|   A|202307|
|    3|   B| 37392|
|    4|   A|205863|
|    5|   B| 34875|
|    6|   A|202505|
|    7|   B| 70713|
|    8|   A|155078|
|    9|   B|125833|
|   10|   B|126512|
+-----+----+------+
only showing top 10 rows
```

**Enriched dataset formed after joins:**

| Store | Date | Dept | Weekly_Sales | IsHoliday | Temperature | Fuel_Price | MarkDown1 | MarkDown2 | MarkDown3 | MarkDown4 | MarkDown5 | CPI | Unemployment | IsHoliday | Type | Size |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2010-02-05 | 1 | 24924.5 | false | 42.31 | 2.572 | NA | NA | NA | NA | NA | 211.0963582 | 8.106 | false | A | 151315 |
| 1 | 2010-02-12 | 1 | 46039.49 | true | 38.51 | 2.548 | NA | NA | NA | NA | NA | 211.2421698 | 8.106 | true | A | 151315 |
| 1 | 2010-02-19 | 1 | 41595.55 | false | 39.93 | 2.514 | NA | NA | NA | NA | NA | 211.2891429 | 8.106 | false | A | 151315 |
| 1 | 2010-02-26 | 1 | 19403.54 | false | 46.63 | 2.561 | NA | NA | NA | NA | NA | 211.3196429 | 8.106 | false | A | 151315 |
| 1 | 2010-03-05 | 1 | 21827.9 | false | 46.5 | 2.625 | NA | NA | NA | NA | NA | 211.3501429 | 8.106 | false | A | 151315 |
| 1 | 2010-03-12 | 1 | 21043.39 | false | 57.79 | 2.667 | NA | NA | NA | NA | NA | 211.3806429 | 8.106 | false | A | 151315 |
| 1 | 2010-03-19 | 1 | 22136.64 | false | 54.58 | 2.72 | NA | NA | NA | NA | NA | 211.215635 | 8.106 | false | A | 151315 |
| 1 | 2010-03-26 | 1 | 26229.21 | false | 51.45 | 2.732 | NA | NA | NA | NA | NA | 211.0180424 | 8.106 | false | A | 151315 |
| 1 | 2010-04-02 | 1 | 57258.43 | false | 62.27 | 2.719 | NA | NA | NA | NA | NA | 210.8204499 | 7.808 | false | A | 151315 |
| 1 | 2010-04-09 | 1 | 42960.91 | false | 65.86 | 2.77 | NA | NA | NA | NA | NA | 210.6228574 | 7.808 | false | A | 151315 |

only showing top 10 rows

### Store wise Metrics formed after grouping by store:

| Store | Total_Weekly_Sales | Average_Weekly_Sales | Data_Count |
|---|---|---|---|
| 31 | 1.9961493036000007E8 | 19757.98578244087 | 10103 |
| 34 | 1.382526272E8 | 13546.21077797374 | 10206 |
| 28 | 1.8927150802999988E8 | 18739.753270297017 | 10100 |
| 26 | 1.4341661982000002E8 | 14565.978043875688 | 9846 |
| 27 | 2.5385718997000003E8 | 24892.84075014709 | 10198 |
| 44 | 4.329367155000003E7 | 6060.992797144061 | 7143 |
| 12 | 1.4429114655999997E8 | 14924.611766652873 | 9668 |
| 22 | 1.4707677093E8 | 15245.855802840262 | 9647 |
| 1 | 2.2240676677000004E8 | 21742.767305699486 | 10229 |
| 13 | 2.8651795036000013E8 | 27394.392423749894 | 10459 |

### Top Performing Stores:

```
Top-Performing Stores:
+-----+--------------------+--------------------+----------+
|Store|  Total_Weekly_Sales|Average_Weekly_Sales|Data_Count|
+-----+--------------------+--------------------+----------+
|   20|3.0140138144999987E8|   29618.84644752357|     10176|
|    4|2.9954526929999995E8|   29175.54001168793|     10267|
|   14| 2.890018644399999E8|  28877.084776179043|     10008|
|   13|2.8651795036000013E8|  27394.392423749894|     10459|
|    2|       2.753871555E8|  26956.456098277213|     10216|
+-----+--------------------+--------------------+----------+
```

**Department wise Metrics formed after grouping by department:**

```
+-----+----+------------------+--------------------+----------+
|Store|Dept|Total_Weekly_Sales|Average_Weekly_Sales|Data_Count|
+-----+----+------------------+--------------------+----------+
|    2|  80| 3723902.120000002|   26041.27356643358|       143|
|    8|  52| 278165.8300000001|  1945.2155944055949|       143|
|   15|  14|        1828384.11|  12785.902867132867|       143|
|   15|  26|         708668.85|   4955.726223776223|       143|
|   18|  95| 8246559.999999999|   57668.25174825174|       143|
|   32|  79|        2211345.25|  15463.952797202797|       143|
|   42|  96|        2171236.81|  15183.474195804196|       143|
|   43|   7|          73898.52|   516.7728671328672|       143|
|    3|  22|443553.09000000014|   3101.769860139861|       143|
|   28|  16|1079739.5700000005|   7550.626363636367|       143|
+-----+----+------------------+--------------------+----------+
```

## Conclusion

This project demonstrates how to build a high-performance data pipeline for processing, aggregating, and storing sales data using Spark. By leveraging caching, persisting, and broadcasting, the pipeline can handle large-scale data efficiently and provide real-time insights into sales performance.