DATE
23/12/2024

WALMART SALES REAL-TIME PROCESSING

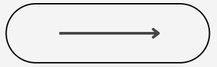
Navadeep Vedantham
navadeep_vedantham@apple.com

Aim:

To design and implement a high-performance data pipeline for processing, aggregating, and storing Walmart sales data in real-time using Apache Spark, Kafka, and Google Cloud Storage (GCS). Focus on optimization through caching, persisting, and broadcasting techniques while ensuring data validation, enrichment, and efficient storage.

Objectives

- Real-Time Data Ingestion: Implement a streaming pipeline to consume weekly sales data from Kafka.
- Data Validation: Clean the incoming sales data by removing negative values and handling missing or invalid values.
- Data Enrichment: Enhance the dataset by joining it with additional metadata from features.csv and stores.csv.
- Aggregation: Calculate store-level and department-level sales metrics.
- Optimized Storage: Store the processed data efficiently in GCS using Parquet and JSON formats.
- Performance Optimization: Use caching, persisting, and broadcasting techniques.
- Real-Time Simulation: Simulate continuous real-time data processing and aggregation.
- Visualization: Visualize key metrics at various stages.



DETAILED WORKFLOW

Step 1: Environment Setup

- Configure Spark for local/cluster execution.
- Set up connections with Kafka and GCS.

Step 2: Data Loading

- Load metadata from features.csv and stores.csv.
- Stream sales data from Kafka topic.

Step 3: Data Validation and Enrichment

- Remove invalid sales values.
- Enrich sales data with metadata using joins and broadcasting.
- Cache frequently used DataFrames for optimization.

Step 4: Data Aggregation

- Compute store-level and department-level metrics:
  - Store Metrics: Total sales, average sales, and top store sales.
  - Department Metrics: Total and average sales by department.

DETAILED WORKFLOW

Step 5: Data Persistence

- Save enriched and aggregated data to GCS in Parquet and JSON formats.
- Utilize checkpointing for fault tolerance.

Step 6: Real-Time Processing

- Implement sliding window aggregations (e.g., 10-minute windows with 5-minute sliding).

Step 7: Visualization

- Display Kafka data, aggregated metrics, and windowed trends in real-time.

TECHNOLOGIES USED AND WHY

- Apache Spark (Scala):
- Why: Fast in-memory processing and rich API for data transformations and analytics.
- Google Cloud Storage (GCS):
- Why: Reliable, scalable cloud storage for output datasets in various formats.
- Spark SQL & DataFrame API:
- Why: Simplifies data querying and processing.
- Local Environment with Spark:
- Why: Easy to configure for development and testing.

FIGURES AND REPORTS

- Kafka Producer and Consumer Visualization
  - Real-time sales data production and consumption.
- Aggregated Metrics Visualisation
  - Metrics displayed for store-level and department-level analysis.
- Windowed Data Trends
  - Real-time visualisation of sliding window aggregations.

FIGURES AND REPORTS

```
/Users/navadeep/Library/Java/JavaVirtualMachines/corretto-11.0.25/Contents/Home/bin/java ...
Produced: {"Store": 99, "Department": 9, "Weekly_Sales": 52298, "IsHoliday": 1}
Produced: {"Store": 75, "Department": 9, "Weekly_Sales": 9320, "IsHoliday": 1}
Produced: {"Store": 25, "Department": 4, "Weekly_Sales": 41863, "IsHoliday": 0}
Produced: {"Store": 73, "Department": 4, "Weekly_Sales": 97197, "IsHoliday": 0}
Produced: {"Store": 53, "Department": 3, "Weekly_Sales": 3421, "IsHoliday": 0}
Produced: {"Store": 22, "Department": 9, "Weekly_Sales": 81013, "IsHoliday": 1}
Produced: {"Store": 45, "Department": 7, "Weekly_Sales": 5143, "IsHoliday": 0}
Produced: {"Store": 88, "Department": 18, "Weekly_Sales": 10847, "IsHoliday": 0}
Produced: {"Store": 25, "Department": 0, "Weekly_Sales": 47187, "IsHoliday": 1}
Produced: {"Store": 15, "Department": 13, "Weekly_Sales": 93016, "IsHoliday": 0}
Produced: {"Store": 66, "Department": 10, "Weekly_Sales": 85889, "IsHoliday": 1}
```

Fig: Data from producer being produced.

```
+--------------------------------------------------------------------------+
|json_string                                                               |
+--------------------------------------------------------------------------+
|{"Store": 20, "Department": 0, "Weekly_Sales": 908, "IsHoliday": 0}   |
|{"Store": 59, "Department": 12, "Weekly_Sales": 49147, "IsHoliday": 0}|
|{"Store": 18, "Department": 1, "Weekly_Sales": 6489, "IsHoliday": 1}  |
|{"Store": 4, "Department": 5, "Weekly_Sales": 65223, "IsHoliday": 0}  |
|{"Store": 89, "Department": 17, "Weekly_Sales": 81438, "IsHoliday": 0}|
|{"Store": 72, "Department": 7, "Weekly_Sales": 15564, "IsHoliday": 1} |
|{"Store": 23, "Department": 11, "Weekly_Sales": 7484, "IsHoliday": 0} |
|{"Store": 88, "Department": 14, "Weekly_Sales": 96948, "IsHoliday": 1}|
|{"Store": 14, "Department": 15, "Weekly_Sales": 93048, "IsHoliday": 1}|
|{"Store": 78, "Department": 18, "Weekly_Sales": 2860, "IsHoliday": 1} |
|{"Store": 48, "Department": 18, "Weekly_Sales": 29238, "IsHoliday": 1}|
|{"Store": 82, "Department": 19, "Weekly_Sales": 16642, "IsHoliday": 1}|
|{"Store": 7, "Department": 17, "Weekly_Sales": 71999, "IsHoliday": 1} |
|{"Store": 2, "Department": 16, "Weekly_Sales": 34726, "IsHoliday": 0} |
|{"Store": 11, "Department": 12, "Weekly_Sales": 66733, "IsHoliday": 1}|
|{"Store": 95, "Department": 3, "Weekly_Sales": 20296, "IsHoliday": 1} |
|{"Store": 72, "Department": 18, "Weekly_Sales": 5776, "IsHoliday": 0} |
|{"Store": 50, "Department": 0, "Weekly_Sales": 87388, "IsHoliday": 1} |
|{"Store": 12, "Department": 7, "Weekly_Sales": 84760, "IsHoliday": 0} |
|{"Store": 44, "Department": 15, "Weekly_Sales": 43714, "IsHoliday": 0}|
+--------------------------------------------------------------------------+
only showing top 20 rows
```

```
-------------------------------------------
Batch: 26
-------------------------------------------
+-----+----------+------------+---------------+----+-----------------+-----------+---------------------+
|Store|Department|Weekly_Sales|Kafka_IsHoliday|Type|Features_IsHoliday|Stream_Date|Stream_Date_Timestamp|
+-----+----------+------------+---------------+----+-----------------+-----------+---------------------+
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
|29   |11        |98872       |1              |B   |false            |NULL       |NULL                 |
+-----+----------+------------+---------------+----+-----------------+-----------+---------------------+
only showing top 20 rows
```

Fig: Data from consumer

CODE HIGHLIGHTS

Kafka Stream Setup

```
val kafkaStreamDF = spark.readStream
  .format("kafka")
  .option("subscribe", topic)
  .load()
```

Data Validation

```
val validDF = kafkaStreamDF.filter(F.col("Weekly_Sales").g
```

Data Enrichment

```
val enrichedDF = validDF
  .join(F.broadcast(cachedFeaturesDF), "Store", "left_oute
  .join(F.broadcast(cachedStoresDF), "Store", "left_outer"
```

Windowed Aggregation

```
val windowedMetricsDF = enrichedWithWatermarkDF
  .groupBy(
    F.window(F.col("Stream_Date_Timestamp"), "10 minutes",
    F.col("Store")
  )
  .agg(
    F.sum("Weekly_Sales").alias("Total_Weekly_Sales"),
    F.avg("Weekly_Sales").alias("Avg_Weekly_Sales"),
    F.max("Weekly_Sales").alias("Top_Store_Sales")
  )
```

CONCLUSION

- Impact: Real-time insights for improved decision-making.
- Performance: Enhanced efficiency with optimized Spark techniques.
- Extensibility: Scalable for new Kafka topics and datasets.

Thank You