Module 4 - Property Optmization

March 7, 2025

0.0.1 Key Considerations for Your Dataproc Cluster

1. **Cluster Resources:**
   - **Master:** `n2-standard-4` (4 vCPUs, 16 GB RAM, 32GB disk)
   - **Workers (2x):** `n2-standard-4` (4 vCPUs, 16 GB RAM, 64GB disk each)
   - **Total:** 8 worker vCPUs, ~32 GB RAM (excluding master node)
2. **Dataproc Features Disabled:**
   - No **autoscaling**, **Metastore**, **advanced execution layer**, **advanced optimizations**
   - **Storage:** `pd-balanced` (no SSDs, so I/O optimization is crucial)
   - **Networking:** Internal IP **enabled**
3. **Optimization Strategy:**
   - Tune **shuffle partitions**, **broadcast join threshold**, and **storage persistence**
   - Adjust **parallelism** based on **2 workers x 4 cores**
   - Avoid **excessive caching** due to **disk-based storage**

[2]:
```python
# https://spark.apache.org/docs/latest/configuration.html
```

[1]:
```python
from pyspark.sql import SparkSession
```

[8]:
```python
spark = SparkSession.builder \
.appName('Olist Ecommerce Performance Optmization') \
.config('spark.executor.memory','6g') \
.config('spark.executor.cores','4') \
.config('spark.executor.instances','2') \
.config('spark.driver.memory','4g') \
.config('spark.driver.maxResultSize','2g') \
.config('spark.sql.shuffle.partitions','64') \
.config('spark.default.parallelism','64') \
.config('spark.sql.adaptive.enabled','true') \
.config('spark.sql.adaptive.coalescePartition.enabled','true') \
.config('spark.sql.autoBroadcastJoinThreshold',20*1024*1024) \
.config('spark.sql.files.maxPartitionBytes','64MB') \
.config('spark.sql.files.openCostInBytes','2MB') \
.config('spark.memory.fraction',0.8) \
.config('spark.memory.storageFraction',0.2) \
.getOrCreate()
```

```
25/02/28 14:37:43 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
```

[ ]:

```

```

[9]:

```python
hdfs_path = '/olist/'
```

[10]:

```python
customers_df = spark.read.csv(hdfs_path + 'olist_customers_dataset.
 ↪csv',header=True,inferSchema=True)
orders_df = spark.read.csv(hdfs_path + 'olist_orders_dataset.
 ↪csv',header=True,inferSchema=True)
order_item_df = spark.read.csv(hdfs_path + 'olist_order_items_dataset.
 ↪csv',header=True,inferSchema=True)
payments_df = spark.read.csv(hdfs_path + 'olist_order_payments_dataset.
 ↪csv',header=True,inferSchema=True)
reviews_df = spark.read.csv(hdfs_path + 'olist_order_reviews_dataset.
 ↪csv',header=True,inferSchema=True)
products_df = spark.read.csv(hdfs_path + 'olist_products_dataset.
 ↪csv',header=True,inferSchema=True)
sellers_df = spark.read.csv(hdfs_path + 'olist_sellers_dataset.
 ↪csv',header=True,inferSchema=True)
geolocation_df = spark.read.csv(hdfs_path + 'olist_geolocation_dataset.
 ↪csv',header=True,inferSchema=True)
category_translation_df = spark.read.csv(hdfs_path +␣
 ↪'product_category_name_translation.csv',header=True,inferSchema=True)
```

[11]:

```python
full_orders_df = spark.read.parquet('/olist/processed/')
```

[12]:

```python
full_orders_df.printSchema()
```

```
root
 |-- customer_id: string (nullable = true)
 |-- order_id: string (nullable = true)
 |-- seller_id: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- order_status: string (nullable = true)
 |-- order_purchase_timestamp: timestamp (nullable = true)
 |-- order_approved_at: timestamp (nullable = true)
 |-- order_delivered_carrier_date: timestamp (nullable = true)
 |-- order_delivered_customer_date: timestamp (nullable = true)
 |-- order_estimated_delivery_date: timestamp (nullable = true)
 |-- order_item_id: integer (nullable = true)
 |-- shipping_limit_date: timestamp (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- product_category_name: string (nullable = true)
|-- product_name_lenght: integer (nullable = true)
|-- product_description_lenght: integer (nullable = true)
|-- product_photos_qty: integer (nullable = true)
|-- product_weight_g: integer (nullable = true)
|-- product_length_cm: integer (nullable = true)
|-- product_height_cm: integer (nullable = true)
|-- product_width_cm: integer (nullable = true)
|-- seller_zip_code_prefix: integer (nullable = true)
|-- seller_city: string (nullable = true)
|-- seller_state: string (nullable = true)
|-- customer_unique_id: string (nullable = true)
|-- customer_zip_code_prefix: integer (nullable = true)
|-- customer_city: string (nullable = true)
|-- customer_state: string (nullable = true)
|-- geolocation_zip_code_prefix: integer (nullable = true)
|-- geolocation_lat: double (nullable = true)
|-- geolocation_lng: double (nullable = true)
|-- geolocation_city: string (nullable = true)
|-- geolocation_state: string (nullable = true)
|-- review_id: string (nullable = true)
|-- review_score: string (nullable = true)
|-- review_comment_title: string (nullable = true)
|-- review_comment_message: string (nullable = true)
|-- review_creation_date: string (nullable = true)
|-- review_answer_timestamp: string (nullable = true)
|-- payment_sequential: integer (nullable = true)
|-- payment_type: string (nullable = true)
|-- payment_installments: integer (nullable = true)
|-- payment_value: double (nullable = true)
|-- is_delivered: integer (nullable = true)
|-- is_canceled: integer (nullable = true)
|-- order_revenue: double (nullable = true)
|-- customer_segment: string (nullable = true)
|-- hour_of_day: integer (nullable = true)
|-- order_day_type: string (nullable = true)
```

1 Optimized Join Stragies

[ ]:
```
# Broadcast

customers_broadcast_df = broadcast(customers_df)
optimized_broadcast_join = full_orders_df.
 ↪join(customers_brodcast_df,'customer_id')
```

[ ]:
```
# Sort and Merge join

sorted_customers_df = customers_df.sortWithinPartitions('customer_id')
sorted_orders_df = full_orders_df.sortWithinPartitions('customer_id')


optimized_merge_full_orders_df = sorted_orders_df.
 ↪join(sorted_customers_df,'customer_id')
```

[ ]:
```

```

[13]:
```
# Bucket join



bucketed_customers_df = customers_df.repartition(10,'customer_id')
bucketed_orders_df = full_orders_df.repartition(10,'customer_id')

bucket_join_df = bucketed_orders_df.join(bucketed_customers_df,'customer_id')
```

[ ]:
```

```

[16]:
```
# Skew Join handling

skew_handled_join = full_orders_df.join(customers_df,'customer_id')
```

```
25/02/28 15:05:56 WARN HintErrorLogger: Unrecognized hint: skew(customer_id)
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```
# Caching -
```