Optimizing PySpark Jobs in AWS Glue

Configuration, Tuning, and Performance

Mastering PySpark Configuration in AWS Glue for Large Datasets

Buddhadeb Mondal

Agenda

- **Understanding PySpark and Glue Environment**
- **Configuring SparkSession in Glue Jobs**
- **Glue Job Configuration Parameters**
- **Glue Memory and Compute Configuration**
- **Tuning Glue Job Performance**
- **Glue DynamicFrames vs DataFrames**
- **Glue Spark Configuration Parameters**
- **Handling Large Datasets in Glue**
- **Glue Job Log Analysis and Monitoring**
- **Parallelism and Partitioning in Glue**

Agenda

- **Glue Spark Data Processing Best Practices**
- **Best Glue Configuration for Large Data Jobs**
- **Advanced PySpark Configurations in Glue**
- **Introduction & Accessing Spark UI**
- **Understanding Spark UI Pages & Metrics**
- **Debugging, Optimization & Case Studies from Spark UI**

Understanding PySpark and Glue Environment

1.1 What is PySpark?

1.2 What is AWS Glue?

1.3 Key Differences Between Glue and Spark on EMR

1.4 Glue vs PySpark Standalone: Performance & Cost Implications

1.5 Infrastructure Components in AWS Glue Glue Jobs
- Glue Crawlers
- Glue Data Catalog
- Glue Triggers

Configuring SparkSession in Glue Jobs

```
spark.sparkContext.getConf().getAll()

conf.set("spark.sql.adaptive.enabled", "true")  # Enable AQE (Adaptive Query Execution)

conf.set("spark.sql.adaptive.coalescePartitions.enabled", "true")

conf.set("spark.sql.adaptive.coalescePartitions.minPartitionSize", "64MB")
```

Glue Memory and Compute Configuration

1. **Glue Job Execution and Worker Types**
2. **Worker Count and Scaling**
3. **Memory Allocation per Worker**
4. **Job Parameters Affecting Compute**
5. **Optimizing Glue Jobs for Performance**
6. **AWS Glue DPU (Data Processing Unit) Cost Considerations**

Glue DynamicFrames vs DataFrames

Feature	**DynamicFrame** (AWS Glue)	**DataFrame** (Apache Spark)
Definition	AWS Glue-specific data abstraction designed for ETL operations.	Standard Apache Spark DataFrame used for general data processing.
Schema Handling	Schema is **flexible** and inferred dynamically (schema-on-read).	Schema is **strict** and must be defined before transformations.
Use Case	Best for working with **semi-structured** or **evolving** schemas.	Best for structured data with a **fixed schema**.

Glue Spark Configuration Parameters

- # Memory Management
- "spark.executor.memory": f"{int(executor_memory)}g",
- "spark.driver.memory": f"{int(driver_memory)}g",
- "spark.executor.memoryOverhead": f"{max(1024, int(0.1 * (executor_memory * 1024)))}",

- # Shuffle Optimization
- "spark.sql.shuffle.partitions": shuffle_partitions,
- "spark.default.parallelism": default_parallelism,
- "spark.sql.files.maxPartitionBytes": "128MB",

Glue Spark Configuration Parameters

- # Parquet-Specific Settings
- "spark.sql.parquet.filterPushdown": "true",
- "spark.sql.parquet.enableVectorizedReader": "true",
- "spark.sql.parquet.mergeSchema": "false",

- # Adaptive Query Execution (AQE)
- "spark.sql.adaptive.enabled": "true",
- "spark.sql.adaptive.coalescePartitions.enabled": "true",
- "spark.sql.adaptive.shuffle.targetPostShuffleInputSize": "64MB",

Glue Spark Configuration Parameters

- # Optimizing Joins and Broadcasts
- "spark.sql.autoBroadcastJoinThreshold": "104857600", # 100MB

- # Parallelism and Partitioning
- "spark.sql.files.openCostInBytes": "134217728", # 128MB
- "spark.sql.files.minPartitionNum": "200",
- "spark.sql.files.ignoreCorruptFiles": "true",

Glue Spark Configuration Parameters

- # Data Storage Optimization
- "spark.sql.sources.partitionOverwriteMode": "dynamic",
- "spark.sql.hive.convertMetastoreParquet": "true",
- "spark.sql.orc.filterPushdown": "true",

- # Miscellaneous Performance Tuning
- "spark.sql.execution.arrow.enabled": "true",
- "spark.serializer": "org.apache.spark.serializer.KryoSerializer",
- "spark.rdd.compress": "true",
- "spark.shuffle.service.enabled": "true",
- "spark.dynamicAllocation.enabled": "true",