AWS Glue vs Spark on EMR for UPSERT Operations

Technical Architecture Decision Document

Nathan Lunceford

2025-01-01

Table of contents

1 Executive Summary

This document compares two AWS-based Apache Spark solutions for handling UPSERT (merge) operations from S3 to multiple target databases including MSSQL, Apache Iceberg, and potentially Clickhouse:

1. AWS Glue: A managed Apache Spark service
2. Apache Spark on Amazon EMR: A more configurable Spark deployment

> **Note**
>
> Both solutions use Apache Spark as their processing engine. The key differences lie in management, configuration, and operational aspects rather than core processing capabilities.

2 Understanding the Options

AWS Glue

AWS Glue provides a managed Apache Spark environment with:

- Built-in Apache Spark engine (same as EMR)
- AWS-specific optimizations and tooling
- Both Spark SQL and PySpark interfaces
- Additional features like DynamicFrames

- Managed infrastructure and scaling

Spark on EMR

Amazon EMR provides a more traditional Spark deployment with:

- Full Apache Spark ecosystem
- Complete configuration control
- Custom cluster management
- Direct access to Spark internals
- Infrastructure flexibility

3 Cost Analysis

3.1 AWS Glue Costs

3.1.1 Pricing Structure

- $0.44 per DPU-Hour (1 DPU = 4 vCPU, 16GB memory)
- Minimum 10-minute billing
- Development endpoints additional cost

3.1.2 Hidden Savings

- No cluster management costs
- Includes Spark optimization
- Less operational overhead

3.1.3 Considerations

- More expensive per compute hour
- Less granular scaling
- Simplified cost model

  :::

3.2 EMR Costs

3.2.1 Direct Costs

- EC2 instance costs
- EMR service charges
- Storage and data transfer

3.2.2 Optimization Options

- Spot instance usage
- More granular scaling
- Resource optimization

3.2.3 Hidden Costs

- Operational overhead
- Management complexity
- Required expertise

  :::

4 Performance Comparison

4.1 AWS Glue Performance

```
# Example Glue Spark UPSERT implementation
from awsglue.context import GlueContext
from pyspark.context import SparkContext

# Initialize Glue Spark context
glueContext = GlueContext(SparkContext.getOrCreate())
spark = glueContext.spark_session

# Read from S3 (using standard Spark)
source_df = spark.read.parquet("s3://bucket/path")

# MSSQL UPSERT
def perform_mssql_upsert(df):
```

```python
# Write to staging table using Spark JDBC
df.write \
    .format("jdbc") \
    .option("url", jdbc_url) \
    .option("dbtable", "staging_table") \
    .mode("overwrite") \
    .save()

# Execute MERGE using Spark SQL
spark.sql("""
MERGE INTO target_table t
USING staging_table s
ON t.key = s.key
WHEN MATCHED THEN UPDATE...
WHEN NOT MATCHED THEN INSERT...
""")
```

Glue Performance Strengths

- Pre-configured Spark optimizations
- AWS service-specific tuning
- Auto-scaling built in
- Warm pools reduce startup time

  :::