Catalyst Optimizer

Karthik Kondpak
9989454737

✅ Day 8 — Spark Optimization Topic

8. Catalyst Optimizer

How Spark Rewrites Your Query to Make It Faster

Catalyst Optimizer is the *brain* of Spark SQL.

It automatically **rewrites your query** to generate the *fastest possible execution plan*.
If you understand Catalyst, you understand **how Spark thinks**.

What Is the Catalyst Optimizer?

Catalyst isSpark's**queryoptimizationengine**.

It takes your SQL/PySpark code → rewrites → optimizes → builds a fast physical plan.
It works in four main phases:

1. **Analysis**
2. **Logical Optimization**
3. **Physical Planning**
4. **Code Generation (Whole-Stage Codegen)**

Think of Catalyst as:

> *“You write the query. Catalyst decides the fastest way to run it.”*

1. Analysis Phase

Catalyst checks:

- Are all columns valid?
- Do data types match?
- Does the table exist?
- Are expressions legal?

Example:

```
df = spark.read.parquet("/delta/sales")
df.filter(col("amount") > 500)
```

Catalyst verifies:

- amount column exists
- Type = integer
- Comparison operation is valid

If you mistype a column → error happens here.

2. Logical Optimization

CatalystrewritesyourqueryusingRULES to make it faster.

Top logical optimizations include:

✓**PredicatePushdown**

Move filters to the data source.

✓**Constant Folding**

Simplifies expressions like:

`amount > 100 + 200 →  amount > 300`

✓**ProjectionPruning(ColumnPruning)**

Read only required columns.

✓**FilterReordering**

Moves selective filters earlier to reduce data size.

✓**NULL Propagation**

If a condition is always false, skip evaluation.

Example of Logical Optimization

Yourcode:

```
df.select("name", "age").filter(col("age") > 30)
```

Catalyst rewrites it internally:

```
Filter(age > 30)
Project(name, age)
```

Order changed to **filter first → project later**,

so fewer rows are processed.

3. Physical Planning

Catalyst decides **which execution strategy is fastest**.

Spark has multiple physical operators:

- Hash Join
- Sort-Merge Join
- Broadcast Hash Join
- Shuffle Exchange
- Range Partitioning
- WholeStageCodegen

Catalyst evaluates all possibilities → picks the cheapest one.

Example: Join Strategy Decision

Yourcode:

```
df1.join(df2, "customer_id")
```

Catalyst checks:

- df2 is small? → Use broadcast join
- Keys sorted? → Use sort-merge join
- Hash distribution possible? → Use shuffle hash join

You don't decide the join type.

Catalyst decides the fastest one.

4. Whole-Stage Code Generation (WSCG)

Afterchoosing a plan,Sparkgenerates **optimizedJavabytecode** at runtime.

Benefits:

- Fewer function calls
- Faster CPU execution
- Better use of vectorized operations

Catalyst makes DataFrame operations **as fast as hand-written Java code**.

Checking Catalyst in Spark UI

Go to:

SQL Tab → Click a query → "Formatted Query Plan"

You will see:

- Parsed Logical Plan
- Analyzed Logical Plan
- Optimized Logical Plan
- Physical Plan

Example snippet:

```
== Optimized Logical Plan ==
Filter (age > 30)
Project [name, age]
```

Scenario — Flipkart Query Optimization

Query:

```
df.filter(col("state") == "Maharashtra") \
  .select("state", "amount")
```

Catalyst optimizes:

1. Push filter → Parquet reader
2. Read only required columns
3. Skip unnecessary metadata
4. Generate optimized execution code

Result:

✓ Less data read

✓ Faster processing

✓ Fewer shuffles

Let's build your Data Engineering journey together!

Call us directly at: 9989454737

https://seekhobigdata.com/