Seekho Bigdata
Data is the New Oil

Spark Optimization Topic

Karthik Kondpak
9989454737

Day 10 — Spark Optimization Topic

Broadcast Join — When It Saves You, and When It Breaks Your Job

Broadcast join is one of the **fastest** join strategies in Spark.

But if used incorrectly, it can **slow down your cluster** or even **crash executors**.

Today you'll learn exactly:

- When broadcast join is ideal
- When it is dangerous
- How Spark decides to broadcast
- How to force or prevent broadcast
- Real-world examples
- Mistakes to avoid

🎯 What is a Broadcast Join?

Broadcast join means:

➡ Spark sends (**broadcasts**) a **small DataFrame** to **all executors**

➡ Large DataFrame doesn't need shuffle

➡ Join becomes very fast (O(n) complexity)

Ideal when:

One table is small (typically < 10–20 MB)

Why Broadcast Join Is Fast

Normal join requires:

- Shuffle both tables
- Sort
- Exchange data across executors
- Build partitions

Broadcast join avoids all this.

✓ **No shuffle**

✓ **No data movement**

✓ **No sort**

Only the small table is copied to each executor.

Default Broadcast Size

Spark automatically broadcasts tables smaller than:
`spark.sql.autoBroadcastJoinThreshold`

Default = **10 MB**

Check or update:

```
spark.conf.get("spark.sql.autoBroadcastJoinThreshold")
spark.conf.set("spark.sql.autoBroadcastJoinThreshold",
"50MB")
```

When Broadcast Join SAVES You

When one table is very small (10–20 MB)

Example:

- Orders = 400M rows
- State lookup table = 300 rows

Broadcasting the small lookup table massively speeds up the join.

When you want to AVOID shuffle on a large dataset

Shuffling a billion-row dataset is costly.

Broadcast join turns this:

Shuffle → Sort → Exchange → Merge

into

✔ Simple map-side hash join

When you're repeatedly joining the same small dimension table

Broadcast improves:

- Star schema joins
- Fact → dimension joins
- Lookup joins

Example:

Customer dimension (2 MB) joins with many fact tables.

When running on clusters with many executors

Idea:

"More executors = better broadcast utilisation"

Each executor gets its own in-memory copy of the small table.

When Broadcast Join BREAKS Your Job

Broadcast join is dangerous when the “small” table is actually *not small*.

❌ 1. When the table is too large to fit in executor memory

If Spark broadcasts:

- 200 MB table
- On executors with 4–8 GB memory

Executors may crash with:

```
OutOfMemoryError: Java heap space
```

❌ 2. When the table looks small but contains complex data

Example:

- A table with 8 MB compressed JSON may expand to 500 MB in-memory

- Structs, arrays, maps → huge in-memory expansion

Spark estimates COMPRESSED size, not actual memory size.

❌ 3. When joining multiple broadcast tables

If Spark broadcasts 4–5 small dimensions:

- Each executor stores 4–5 copies
- Memory pressure increases
- GC overhead increases

❌ 4. When small table has high cardinality keys

If keys are not evenly distributed, CPU time increases:

- Hash collisions
- Large hash map
- Slower join

Broadcasting does not solve skew.

Example — Swiggy / Zomato Analysis

Tables:

- orders: 500M rows
- `restaurants_dim: 15 MB`
- `cities_dim: 5 MB`

Broadcast join helps:

```
from pyspark.sql.functions import broadcast

df = orders.join(broadcast(restaurants_dim),
"restaurant_id", "inner")
```

But if restaurants_dim grows to 120 MB in Delta (due to history),

broadcast join will start failing.

How to Force Broadcast Join

```
df = large_df.join(broadcast(small_df), "id")
```

Or:

```
spark.conf.set("spark.sql.autoBroadcastJoinThreshold",
-1)
```

🛑 How to Disable Broadcast Join

Disable completely:

```
spark.conf.set("spark.sql.autoBroadcastJoinThreshold",
"-1")
```

Disable for specific table:

```
small_df.hint("no_broadcast")
```

How to Check in Spark UI

Under **SQL → Physical Plan**, look for:

```
BroadcastHashJoin
BroadcastExchange
```

This means Spark used a broadcast join.

Let's build your Data Engineering journey together!

Call us directly at: 9989454737

https://seekhobigdata.com/