Seekho Bigdata
Data is the New Oil

Broadcast Join vs Sort-Merge Join

Karthik Kondpak
9989454737

Day 5 — Spark Optimization Topic

5. Broadcast Join vs Sort-Merge Join

Understanding this difference is **mandatory** for PySpark performance optimization.

1. Broadcast Join

Spark**copiesthesmallerDataFrame** to every executor and performs a join **locally**, avoiding shuffle.

✔How it works

- Small table (usually < 100–500 MB) is **broadcasted**
- Each executor gets a **local copy**
- Large table is scanned once
- Join is done **map-side** (no shuffle of big table)

✔Advantages

- Zero shuffle for the big table
- Very fast for small-to-big joins
- Ideal for dimension tables or lookups

✓ When to use

- Dimension table < threshold (default 10 MB, configurable)
- Star-schema joins
- Lookup tables (country codes, categories, products)
- When joining a large fact table with a small reference table

✓Example

```
from pyspark.sql.functions import broadcast

result = large_df.join(broadcast(small_df),
"customer_id")
```

2. Sort-Merge Join

The default join in Spark when both tables are **large** or when no broadcast is used.

✓How it works

- Both DataFrames are **shuffled** on join key
- Each side is **sorted**
- Merge happens on sorted partitions

✓Advantages

- Great for large-to-large joins

- Stable and scalable
- Works reliably at massive scale

✔ Disadvantages

- Shuffle + sort = expensive
- Needs large memory
- Can lead to skew

✔ Example

```
result = large_df.join(another_large_df, "txn_id")
```

Broadcast Join vs Sort-Merge Join — Side-by-Side

Feature	Broadcast Join	Sort-Merge Join
Shuffle	No (for small table)	Yes (both tables)
Best for	Small-to-big joins	Big-to-big joins
Speed	Very fast	Medium/Slow (depending on size)
Memory usage	More (stores copy on each executor)	Moderate
Requires sorting	No	Yes

Skew handling	Limited	More advanced mechanisms

Spark Choosing Strategy

Sparkautomaticallydecidesjoinstrategy based on:

1. Size of smaller DataFrame

If < `spark.sql.autoBroadcastJoinThreshold`

(default 10 MB):

→ **Broadcast Join**

2. If broadcast() keyword is manually used

→ **Forced Broadcast Join**

3. If both sides are large

→ **Sort-Merge Join**

Real Example: Indian E-commerce Dataset

customers_small (50,000 rows)

transactions (400 million rows)

Goal: Join both by **customer_id**

Best Approach:

```
df = transactions.join(broadcast(customers_small),
"customer_id")
```

Reason: customers_small fits into memory → avoids shuffle of 400M rows.

When Broadcast Join is Bad

Do **NOT** use broadcast when:

✗ The small table is actually large

Broadcasting a 1–2 GB table can crash executors.

✗ High concurrency

Multiple queries broadcasting big tables can cause memory pressure.

✗ Running on small cluster

Executors may not have memory for several broadcast copies.

When Sort-Merge Join is Better

✓Both tables are large

✓You need a stable join with sorting

✓After applying salting or skew-handling techniques

✓When broadcast threshold is exceeded

Seekho Bigdata
Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

https://seekhobigdata.com/