Seekho Bigdata
Data is the New Oil

Spark Optimization Topic

Karthik Kondpak
9989454737

Day 14 — Spark Optimization Topic

Joins — Choosing the Right Join for Speed

(with Physical Plan Explained)

In PySpark, JOINs are the **most expensive operations** because they often trigger:

✔ Shuffle

✔ Sort

✔ Network transfer

✔ Data skew

✔ High memory usage

Choosing the **right join strategy** can reduce runtime from **minutes → seconds**.

6 Join Types Every Data Engineer Must Know

Spark uses different physical join strategies depending on data size and configuration.

Broadcast Hash Join (BHJ)

Fastest join when one table is small.

✓ **When to use**

When one table is ≤ **10 MB** (default broadcast threshold).

```
df1.join(broadcast(df2), "id")
```

✓ **Physical Plan**

```
BroadcastHashJoin
  [*BroadcastExchange*]
```

✓ Why it's fast

- No shuffle
- Small table copied to executors
- Big table scanned locally

Shuffle Hash Join (SHJ)

Used when both tables are medium-sized and hashable.

✓Physical Plan

```
HashExchange
HashExchange
ShuffledHashJoin
```

✓ When to use

- Tables fit in memory
- Join key is well distributed
- No skew

⚠ If skew exists: performance drops sharply.

Sort Merge Join (SMJ)

Default join for large datasets.

✓ **When to use**

- Both tables huge
- Join key cannot be broadcast
- Sorting and merging required

✓ **Physical Plan**

```
Exchange (HashPartitioning)
Sort
Exchange (HashPartitioning)
Sort
SortMergeJoin
```

✓ **Why expensive**

- Heavy shuffle
- Heavy sort
- Large data movement

Cartesian Join (CROSS JOIN)

Produces N × M rows — extremely expensive.

Use only when **logically required**.

✓**Physical Plan**

```
CartesianProduct
```

⚠Avoid at all cost unless business logic demands it.

Skew Join Optimization (AQE)

When data is skewed:

- One key has too many rows
- Executors take long
- Entire job slows

AQE fixes it by:

✓ Splitting skewed partitions

✓Broadcasting smaller side

✓Using hybrid join strategy

✓**Physical Plan**

```
AdaptiveSparkPlan
  SplitSkewedPartition
  BroadcastHashJoin
```

Bucketed Sort Merge Join (Ultra-fast for large data)

If two tables are **bucketed on the same key** AND **same bucket count**, Spark avoids shuffle.

✓**Physical Plan**

```
Scan (Bucketed)
SortMergeJoin
```

✓No Exchange

✓No repartitioning

✓Much faster

Join Selection Matrix — Which Join Should You Use?

Table Sizes	Best Join	Why
Small + Large	Broadcast Hash Join	No shuffle
Medium + Medium	Shuffle Hash Join	Faster than SMJ
Large + Large	Sort Merge Join	Handles big data
Pre-bucketed tables	Bucketed SMJ	Zero shuffle
Skewed keys	AQE + Skew Join	Splits heavy partitions

Physical Plan Comparison

Broadcast Join

```
BroadcastHashJoin
  +- BroadcastExchange
  +- Project / Filter / Scan
```

Sort Merge Join

```
SortMergeJoin
  +- Sort
  +- Exchange
  +- Scan
```

Shuffle Hash Join

```
ShuffledHashJoin
  +-  Exchange
  +- Exchange
```

Scenario Example :

You work at **Swiggy India**, processing daily restaurant orders.

Tables:

- `orders` →900M rows
- `restaurants` → 40 MB

Best join strategy: Broadcast Join

```
df_orders.join(broadcast(df_restaurants),
"restaurant_id")
```

✓Saves shuffle

✓Saves sort

Saves network cost

✓ Ideal for daily pipelines

✓

Expert Tip

Before joining ALWAYS check size:

```
df.count()
spark.c onf.ge t("sp ark.s ql.aut oBroadc astJoi nThre shold
)
```

And check for skew:

```
df.groupBy("join_key").count().orderBy(desc("count"))
```

Seekho Bigdata

Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/