Seekho Bigdata
Data is the New Oil

Joins – Choosing the Right Join for Speed

Karthik Kondpak
9989454737

Day 20: Joins — Choosing the Right Join for Speed (with Physical Plan)

Master this and your Spark jobs instantly become faster + cheaper.

Why Join Strategy Matters

A join is one of the **costliest** operations in Spark because it often requires:

- Shuffle
- Sorting
- Data movement across nodes
- Large memory usage

Choosing the wrong join = **slow job + high cluster cost + OOM errors**

Choosing the right join = **10× faster pipeline**

Types of Physical Join Strategies in Spark

Spark has **3 physical join algorithms**:

Broadcast Hash Join (BHJ)

Fastest join when one dataset is small

✓ When to Use

- One side is **< 10–20% of cluster memory**
- Typically **< 500 MB**
- You can broadcast dimension tables

✓ Advantages

- **No shuffle**

Data is sent to all executors

Very fast

When to Avoid

✗

- Large dataset → causes executor OOM
- Huge dimension table

🔧 Execution

- Spark broadcasts small table
- Large table scans locally
- Hash tables created on executors

🧬 Physical Plan Output

```
== Physical Plan ==
*(1) BroadcastHashJoin [id], [id], Inner, BuildRight
:- *(1) FileScan parquet (big_table)
+- BroadcastExchange HashedRelationBroadcastMode
   +- *(2) FileScan parquet (small_table)
```

Shuffle Hash Join

Used when both tables are small enough but not broadcastable

✔ When to Use

- Both tables **fit into executor memory**
- But not small enough for broadcast

❌ When to Avoid

- Large tables
- Skewed keys

🧬 Physical Plan

```
== Physical Plan == HashJoin [id], [id], Inner, BuildRight :- Exchange hashpartitioning(id) +- Exchange hashpartitioning(id)
```

✔Pros

- No sorting needed
- Faster than sort-merge when values are small

❌ Cons

- Still causes shuffle
- Memory-intensive

Sort-Merge Join (SMJ)

Most common join for large datasets

✔ When to Use

- **Both tables large**
- Join key must be sorted
- Recommended for big fact tables

❌ When to Avoid

- Very small tables
- Skewed partitions

🧬 Physical Plan

```
== Physical Plan ==
SortMergeJoin [id], [id], Inner
:- Sort [id ASC NULLS FIRST]
: +- Exchange hashpartitioning(id)
+- Sort [id ASC NULLS FIRST]
   +- Exchange hashpartitioning(id)
```

✓Pros

- Handles very large datasets
- Stable plan

✗ Cons

- Requires **sorting** → expensive
- Heavy shuffle

How Spark Decides Join Strategy

Condition	Spark Choice
One table ≤ broadcast threshold	**Broadcast Hash Join**
Both small but not broadcastable	**Shuffle Hash Join**
Both large	**Sort-Merge Join**
Hint used	Your hint overrides Spark (e.g. broadcast)

PySpark Examples

Broadcast Join

```
from pyspark.sql.functions import broadcast
```

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

Sort Merge Join

```
spark.conf.set("spark.sql.join.preferSortMergeJoin",
True)

df = df1.join(df2, "id")
```

Shuffle Hash Join

```
spark.conf.set("spark.sql.join.preferSortMergeJoin",
False)

df = df1.join(df2, "id")
```

🔥 Which Join Should YOU Use?

✅ **When small table < 500 MB**

Broadcast Join

✅ **When both mid-size but fit in memory**

Shuffle Hash Join

✅ **When fact table is huge**

Sort-Merge Join

Seekho Bigdata

Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/