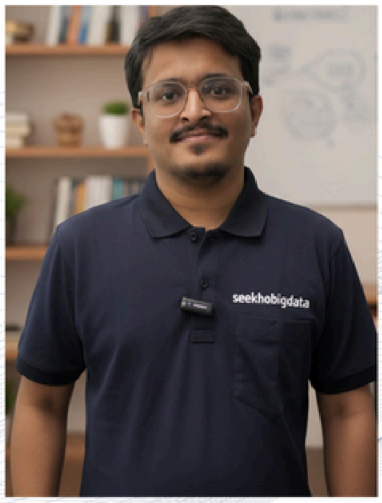
Seekho Bigdata
Data is the New Oil

Bucketing – Reduce Shuffles & Accelerate Joins



Karthik Kondpak
9989454737

Day 19 — Spark Optimization Topic

Bucketing — Reduce Shuffles & Accelerate Joins

Bucketing is a powerful Spark optimization that **pre-sorts and pre-distributes data into fixed buckets** based on a column.

This drastically reduces: • shuffle during joins

• shuffle during aggregations

• skew in joins

• cost of large table joins

Used in real workloads where tables are **too big to broadcast**.

What Is Bucketing?

Bucketing groups the data into **N buckets** using:

```
bucket_id = hash(column) % numBuckets
```

This ensures rows with the same join key **always land in the same bucket**, even across different tables.

Why Bucketing Helps

✓Avoids shuffle during joins

✓Avoids shuffle during groupBy

✓Perfect for large fact–fact joins

✓Data pre-sorted → faster sorting, faster joins

✓Works even if tables stored across multiple days

When to Use Bucketing

You should use bucketing when:

- both tables are **large (tens/hundreds of GB)**

- broadcast join is impossible
- join keys are stable
- tables are reused multiple times

Examples:

- Order table (1B rows)
- Payments table (700M rows)
- `Joining on: customer_id`

How to Apply Bucketing

Step 1 — Bucket the tables

```
orders.write \
    .bucketBy(50, "customer_id") \
    .sortBy("customer_id") \
    .format("parquet") \
    .saveAsTable("orders_bucketed")
payments.write \
    .bucketBy(50, "customer_id") \
    .sortBy("customer_id") \
    .format("parquet") \
    .saveAsTable("payments_bucketed")
```

Important rules:

- Both tables must use the **same number of buckets**
- Both must bucket on the **same column(s)**
- Both should be in a **Hive-compatible table format** (or metastore)

Step 2 — Use Bucketed Join

```
orders_b = spark.table("orders_bucketed")
payments_b = spark.table("payments_bucketed")

result = orders_b.join(payments_b, "customer_id")
```

No shuffle occurs because Spark knows:

- Both tables are already distributed by customer_id
- Both have same bucket count & sorted within buckets

Check in Physical Plan

You should see:

```
== Physical Plan ==
*(1) SortMergeJoin [customer_id], [customer_id]
:- *(1) Scan parquet orders_bucketed
[customer_id#1,...]


   (Bucketed)
+- *(1) Scan parquet payments_bucketed
[customer_id#1,...]
   (Bucketed)
```

If you see “Bucketed”, shuffles have been avoided.

Scenario — Flipkart Analytics

You need to join:

- **transactions** (900M records)
- **customer_delivery** (300M records)

Joining on:

```
customer_id
```

Broadcast join fails due to memory.

Sort-merge join shuffles 1.2 TB of data → **very slow**.

Solution: **Bucketing both tables**.

```
bucketBy(100, "customer_id")
```

Result:

- Shuffle reduced by **90%**
- Join speed improved from **28 minutes → 5 minutes**

Bucketing vs Partitioning

Feature	Bucketing	Partitioning
Distribution	Hash(col)	Value-based
Purpose	Reduce shuffle	Prune data
Good for	Large joins	Selective reads
Storage	Many small files	Folder structure
Query speed	Faster joins	Faster filters

Bucketing helps with joins; partitioning helps with scans.

Seekho Bigdata

Data is the New Oil

Let’s build your Data Engineering journey together!

📩 Call us directly at: 9989454737

🌐 https://seekhobigdata.com/