Seekho Bigdata
Data is the New Oil

Spark Optimization Topic

Bucketing — How It Reduces Shuffles and Speeds Up Joins

Karthik Kondpal
9989454737

Day 13 — Spark Optimization Topic

🔥 Bucketing — How It Reduces Shuffles and Speeds Up Joins

Bucketing is one of Spark's most **powerful shuffle-reduction techniques**, but also one of the **least understood**.

If you struggle with large JOINs, slow aggregations, or skewed data —

Bucketing can save your job.

What Is Bucketing?

Bucketing **divides data into fixed number of buckets** using a **hash function** on a column.

Example:

```
BUCKET 4 ON customer_id
```

means Spark will hash the customer_id values → assign rows into 4 fixed buckets.

Unlike partitioning:

✔ Buckets are **pre-distributed**

✔ Every bucket is same size (hash-based)

✔ Useful for JOIN and GROUP BY

✔ Works across multiple tables

📌 Why Bucketing Matters

✔ **Joins become faster**

If two tables are bucketed on the same key, Spark **avoids shuffles**.

✔ **GroupBy becomes faster**

Spark can skip repartitioning and reuse existing buckets.

✔ **Reduces memory pressure**

Less shuffle → fewer shuffle partitions → fewer spills.

✓ **Works on massive datasets**

Used heavily in:

- E-commerce (Flipkart, Amazon India)
- Telecom
- Banking
- Ad-tech

Bucketing Example (Syntax)

Creating a bucketed table

```
df.write \
  .bucketBy(8, "customer_id") \
  .sortBy("customer_id") \
  .format("parquet") \
  .saveAsTable("bucketed_customers")
```

Reading it

```
df = spark.table("bucketed_customers")
```

Why SORT BY?

Sorting inside the bucket:

✓ speeds up merges

✓ reduces downstream sort work

✓ helps in window functions, joins, and groupBy

Scenario

You work at Flipkart India, handling customer transactions.
Two tables:

1. customers (100M rows)
2. orders (1.2B rows)

You frequently join:

```
SELECT *

FROM customers c

JOIN orders o
```

```
ON c.customer_id = o.customer_id
```

If both tables are bucketed on customer_id with same number of buckets, Spark:

✓Avoids shuffle

✓Reads matching buckets directly

Pipeline runs 3–5× faster

✓ Reduces cluster cost

✓

How Bucketing Removes Shuffle

Normal join physical plan:

```
Exchange  (HashPartitioning)
Sort Join
```

With bucketing:

```
Scan (Bucketed)
SortMergeJoin
```

Notice:

No Exchange → No shuffle

✔ Already bucketed → Already partitioned

✔ Spark just sorts inside bucket → Joins → Done

When to Use Bucketing

Use bucketing when:

✓ You join the **same two big tables repeatedly**

✓ Same join key is used often (customer_id, product_id, mobile_no)

✓ Data is stable (not updated very frequently)

✓ You want predictable distribution

⚠ When NOT to Use Bucketing

Bucketing is NOT ideal if:

✗ Data is highly dynamic

✗ You don’t manage tables (external sources)

✗ You don’t know join keys ahead of time

✗ You rarely join those tables

Seekho Bigdata

Data is the New Oil

Let's build your Data Engineering journey together!

📩 Call us directly at: 9989454737

https://seekhobigdata.com/