Sorting Strategies in Spark (HTX xData)

Context

Large camera streaming datasets require efficient joins and ordering to support enrichment and alert ranking use-cases.

Dataset A (events) — millions per hour

Dataset B (locations) — ~10 K static records

Sorting Strategies

Sort Within Partition

- df.sortWithinPartitions("ts")

- Cheap; preserves locality; used for window ops inside micro-batch.

Global Sort (orderBy)

- Full shuffle.

- Only when absolutely needed.

- Often inefficient for streaming.

Range Partitioning + Sort

- repartitionByRange("location").sortWithinPartitions(...)

- Best for range window analytics or distributed ordering.

Sort-Merge Join (SMJ)

- Required when join keys are large and broadcast is impossible.

- Must sort both sides + shuffle — expensive but scalable.

Broadcast Hash Join (BHJ)

- Ideal for small dimension table (~10–100 k rows).

- Zero shuffle join path.

- Default choice for this case.

Strategy for This Project

- Broadcast dataset B to avoid shuffling camera events.
- If broadcast disabled (e.g., size growth), use SMJ on geographical_location_oid.

Additional Notes

- Use adaptive query execution (AQE) to auto-switch join strategy.
- Enable spark.sql.autoBroadcastJoinThreshold.
- Monitor skew; if hotspots appear, apply salting on hot OIDs.