SPARK

SPARK Optimization Techniques for Data Engineers 🚀

Puneeth Kumar

@puneethkumar

1

1. Use Shared Variables

Tip: Utilize Broadcast and Accumulator variables.

Why: Efficiently share data across nodes and accumulate global values without the need for shuffling.

2

2. Prefer Hash Aggregate over Sort Aggregate

Tip: Use Hash Aggregate when dealing with large datasets.

Why: Sorting operations (nlogn) slow down performance. Hash Aggregate is faster, provided all columns except groupBy columns are Integers.

3

3. Cache Data Wisely

Tip: Use Cache or Persist to store data in memory when needed.

Why: Reduces computation time by reusing intermediate results, leading to faster execution.

4

4. Use Kryo Serializer

Tip: **Switch to Kryo Serializer.**

Why: **It offers faster serialization and deserialization compared to the default Java serializer, enhancing overall speed.**

5

5. Choose the Right File Format & APIs

Tip: Select optimal file formats (e.g., Parquet, ORC) and APIs (e.g., DataFrame API).

Why: Efficient file formats and APIs improve read/write performance and compression, leading to faster data processing.

6

6. Use Coalesce() Over Repartition()

Tip: Opt for Coalesce() to reduce the number of partitions.

Why: Coalesce() minimizes shuffling by reducing partitions, whereas Repartition() can cause extensive shuffling, slowing down performance.

7

7. Avoid Early Wide Transformations

Tip: Filter unnecessary data before wide transformations.

Why: Delaying shuffling and reducing the data volume early on minimizes performance bottlenecks.

8

8. Select Required Columns Only

Tip: Specify only the columns you need instead of using Select(*).

Why: Reduces the amount of data processed, leading to faster query execution.

9

9. Optimal Join Strategies

Tip: Choose the right join based on dataset sizes.

- Sort Merge Bucket Join: For large datasets.
- Broadcast Join: For joining a large dataset with a smaller one.

Why: Efficient joins improve performance by reducing shuffle operations.

10

10. Enable AQE (Adaptive Query Execution)

Tip: Turn on Adaptive Query Execution.

Why: AQE helps handle data skewness and allows dynamic optimization, improving query performance.

11

11. Proper Resource Allocation

Tip: Allocate resources appropriately.

Why: Even the best optimizations won't help if resources (memory, CPU) are not adequately allocated, leading to performance issues.

12

In Summary

Optimize Spark by using efficient data handling, strategic caching, advanced serialization, right file formats, minimizing shuffling, and ensuring proper resource allocation.

13

Puneeth Kumar

@puneethkumar

FOLLOW FOR MORE CONTENT