PySpark Performance Tuning — Complete Practical Guide

1. Diagnose First — Don't Tune Blind

• Use Spark UI (Jobs, Stages, Tasks, SQL tabs) to locate long stages, skewed tasks, and large shuffles.
• Inspect physical plans using df.explain("formatted").
• Check partition count, shuffle partitions, and data distribution.
• Validate input data skew and small file issues.

2. Fix by Symptom

- Data Skew → Use AQE skew join, broadcast small side, or salt keys.
- High Shuffle Time → Filter early, project required columns, adjust shuffle partitions.
- Driver/Executor OOM → Avoid collect(), cache strategically, enable Kryo serializer.
- Python UDFs Slow → Replace with built-in or pandas UDFs with Arrow enabled.
- Small Files → Coalesce or compact files, partition on low-cardinality columns.

3. Tuning Levers

• Query Structure: Project/filter early, join small to large, pre-aggregate.
• Spark SQL Optimizations: Enable AQE, coalesce partitions, dynamic partition pruning.
• I/O Formats: Prefer Parquet/Delta, leverage predicate pushdown.
• Partitions: Tune shuffle partitions (~2-4 tasks/core).
• Cluster: Moderate executors (4–8 cores), dynamic allocation on.

4. Config Cheat-Sheet

```
spark.sql.adaptive.enabled = true
spark.sql.adaptive.skewJoin.enabled = true
spark.sql.shuffle.partitions = 400
spark.serializer = org.apache.spark.serializer.KryoSerializer
spark.sql.execution.arrow.pyspark.enabled = true
```

5. Delta/Databricks Specifics

• OPTIMIZE and ZORDER hot tables for query skipping.
• Enable optimizeWrite and autoCompact for small file management.
• Run VACUUM periodically (retain 7–30 days based on compliance).

6. Repeatable Checklist

1. Analyze plan/UI.
2. Reduce data early.
3. Enable AQE/DPP.
4. Fix skew.
5. Remove UDFs.

6. Tune partitions.
7. Compact outputs.
8. Cache only if reused.
9. Adjust executor sizing.
10. Validate again via Spark UI.