Apache Spark Configuration Cheat Sheet

Scenario 1: Slow Join Job

Problem: Join operation is slow with high shuffle spill.

Solutions:

- Increase shuffle partitions: spark.sql.shuffle.partitions = 800
- Optimize executor resources: spark.executor.memory = 6g, spark.executor.cores = 4
- Cache reused DataFrames to avoid recomputation
- Use broadcast join if one dataset is small:
  df1.join(broadcast(df2), "id")

Scenario 2: OutOfMemoryError in Executors

Problem: Executors fail due to insufficient memory.

Solutions:

- Increase executor memory: spark.executor.memory = 8g
- Use G1GC collector:
  spark.executor.extraJavaOptions = "-XX:+UseG1GC"
- Avoid large collect() or toPandas() calls on big datasets

Scenario 3: Skewed Tasks (Stragglers)

Problem: Some tasks take significantly longer due to data skew.

Solutions:

- Enable speculative execution: spark.speculation = true
- Use salting techniques for skewed keys
- Repartition skewed data: df.repartition("key_column")

Scenario 4: Job Crashes on Full Data Load

Problem: Job works on sample but fails with full dataset.

Solutions:
- Tune based on sample profiling using Spark UI
- Scale executor memory/cores or cluster size
- Tune shuffle partitions and default parallelism:
  spark.sql.shuffle.partitions = 1000
  spark.default.parallelism = 800

Scenario 5: High Executor Idle Time

Problem: Executors are idle, causing poor resource usage.

Solutions:
- Enable dynamic allocation:
  spark.dynamicAllocation.enabled=true
  spark.dynamicAllocation.minExecutors=4
  spark.dynamicAllocation.maxExecutors=100
- Reduce shuffle partitions if over-partitioned