10 PySpark Performance Optimization

Performance Optimization: Partitioning, Bucketing, Caching, Broadcast Joins & Spark Optimization

Basic Level Questions (1-10)

1. **What is data partitioning in Spark and why is it important for performance?** *Focus on: Data locality, parallelism, shuffle reduction, query performance*
2. **Explain the difference between partitioning and bucketing in Spark.** *Focus on: Hash-based distribution vs range-based distribution, use cases, performance implications*
3. **What is caching in Spark and when should you use it?** *Focus on: Memory vs disk storage, iterative algorithms, reuse patterns, cache() vs persist()*
4. **What is a broadcast join and when is it beneficial?** *Focus on: Small table optimization, shuffle elimination, memory requirements, automatic vs manual broadcast*
5. **What are the different storage levels available for caching in Spark?** *Focus on: MEMORY_ONLY, MEMORY_AND_DISK, serialization options, replication factors*
6. **How does Spark determine the number of partitions for a DataFrame?** *Focus on: Default parallelism, file size considerations, spark.sql.files.maxPartitionBytes*
7. **What is the difference between narrow and wide transformations in Spark?** *Focus on: Shuffle operations, stage boundaries, performance implications*
8. **How do you manually repartition a DataFrame and when would you do it?** *Focus on: repartition() vs coalesce(), performance trade-offs, data distribution*
9. **What is the Catalyst optimizer and how does it improve query performance?** *Focus on: Rule-based optimization, predicate pushdown, column pruning, constant folding*
10. **What are some common signs that a Spark job needs performance optimization?** *Focus on: Long execution times, memory errors, data skew, excessive shuffling*

Intermediate Level Questions (11-20)

11. **How would you optimize a Spark job that's experiencing significant data skew?** *Focus on: Salting techniques, custom partitioners, broadcast joins, repartitioning strategies*
12. **Explain the concept of bucketing and how it improves join performance.** *Focus on: Pre-shuffled data, hash-based distribution, join optimization, sort-merge joins*
13. **How do you choose the optimal number of partitions for a given dataset?** *Focus on: Data size, cluster resources, task overhead, partition size guidelines*
14. **What are the trade-offs between different caching storage levels?** *Focus on: Memory usage, serialization overhead, fault tolerance, performance characteristics*
15. **How would you optimize a job that performs multiple joins on the same dataset?** *Focus on: Caching strategies, join reordering, broadcast optimization, materialization*
16. **Explain how predicate pushdown works and its performance benefits.** *Focus on: Filter early principle, I/O reduction, partition pruning, data source optimization*
17. **How do you handle small file problems in Spark and what performance impact do they have?** *Focus on: Coalescing, repartitioning, file compaction, metadata overhead*
18. **What is dynamic partition pruning and how does it improve performance?** *Focus on: Runtime optimization, fact-dimension joins, partition elimination*
19. **How would you optimize window function operations for better performance?** *Focus on: Partitioning strategies, ordering considerations, frame specifications, memory usage*
20. **Explain the concept of Adaptive Query Execution (AQE) in Spark.** *Focus on: Runtime optimization, dynamic coalescing, skew handling, join strategy changes*

Advanced/Difficult Level Questions (21-30)

21. Design a comprehensive performance optimization strategy for a complex ETL pipeline processing 10TB of data daily. *Focus on: End-to-end optimization, resource allocation, caching strategy, partition design, monitoring*

22. How would you implement a custom partitioner to handle extreme data skew in a specific business scenario? *Focus on: Hash function design, load balancing, key distribution analysis, performance testing*

23. Explain how you would optimize a multi-stage pipeline with interdependent caching requirements. *Focus on: Cache hierarchy, memory management, eviction policies, dependency analysis*

24. How do you troubleshoot and resolve memory pressure issues in a large Spark cluster? *Focus on: Memory profiling, garbage collection tuning, partition sizing, broadcast variable optimization*

25. Design an optimization strategy for a real-time streaming application with strict latency requirements. *Focus on: Micro-batch sizing, checkpoint optimization, state management, resource allocation*

26. How would you implement intelligent broadcast join selection for a multi-table join scenario? *Focus on: Cost-based optimization, statistics collection, dynamic broadcast decisions, memory constraints*

27. Explain how you would optimize Spark SQL queries that involve complex nested data structures. *Focus on: Schema optimization, columnar storage benefits, projection pushdown, predicate optimization*

28. How do you handle performance optimization when dealing with highly compressed data formats? *Focus on: Decompression overhead, splittability, codec selection, CPU vs I/O trade-offs*

29. Design a performance monitoring and alerting system for production Spark applications. *Focus on: Metrics collection, performance baselines, anomaly detection, automated optimization*

30. How would you implement a cost-based optimizer for automatic partition and bucketing decisions? *Focus on: Statistics collection, cost modeling, automated decision making, feedback loops*

Performance Debugging Scenarios

Real-World Troubleshooting

Scenario A: A Spark job that previously ran in 2 hours now takes 8 hours. Walk through your debugging approach.

Scenario B: You're seeing frequent OOM errors in executors during a large join operation. How do you resolve this?

Scenario C: A streaming job is experiencing increasing latency over time. What could be the causes and solutions?

Scenario D: After adding a new data source, query performance degraded significantly. How do you identify and fix the issue?

Configuration & Tuning

Critical Configuration Parameters

Memory Management:

- spark.executor.memory vs spark.executor.memoryFraction
- spark.sql.adaptive.enabled and related settings
- Garbage collection tuning parameters

Shuffle Optimization:

- spark.sql.shuffle.partitions

- spark.sql.adaptive.shuffle.targetPostShuffleInputSize
- spark.serializer selection

Caching Strategy:

- spark.sql.adaptive.localShuffleReader.enabled
- Storage level selection criteria
- Cache eviction policies

Practical Optimization Exercises

Hands-On Performance Challenges

Exercise 1: Given a poorly performing join query, identify bottlenecks and implement optimizations.

Exercise 2: Optimize a complex aggregation query with multiple grouping levels.

Exercise 3: Design an optimal partitioning strategy for a time-series dataset with multiple access patterns.

Exercise 4: Implement a caching strategy for a multi-stage analytical pipeline.