06 PySpark Basics

Spark Architecture, DataFrames, RDDs & Basic Transformations/Actions

Basic Level Questions (1-12)

1. What is Apache Spark and how does it differ from traditional MapReduce?

What to look for: In-memory processing, DAG execution, faster iterative algorithms, unified analytics engine, and ease of use.

2. Explain the core components of Spark's architecture.

What to look for: Driver, executors, cluster manager, SparkContext, and the relationship between these components.

3. What is an RDD (Resilient Distributed Dataset) and what are its key characteristics?

What to look for: Immutability, fault tolerance, lazy evaluation, distributed collections, and lineage tracking.

4. What is the difference between transformations and actions in Spark?

What to look for: Lazy evaluation for transformations, immediate execution for actions, and examples of each type.

5. What is a DataFrame in Spark and how does it differ from an RDD?

What to look for: Structured data, schema, Catalyst optimizer, higher-level API, and performance benefits.

6. Name and explain 5 basic transformations in Spark.

What to look for: map(), filter(), flatMap(), distinct(), union() with clear explanations and use cases.

7. Name and explain 5 basic actions in Spark.

What to look for: collect(), count(), first(), take(), reduce() with understanding of when to use each.

8. What is lazy evaluation in Spark and why is it beneficial?

What to look for: Deferred execution, optimization opportunities, lineage graph construction, and performance benefits.

9. How do you create a DataFrame from an RDD in Spark?

What to look for: toDF() method, createDataFrame(), schema definition, and Row objects.

10. What is the Spark SQL Catalyst optimizer?

What to look for: Query optimization, rule-based optimization, code generation, and performance improvements.

11. Explain the concept of partitions in Spark.

What to look for: Data distribution, parallelism, partition size considerations, and impact on performance.

12. What are the different ways to create an RDD in Spark?

What to look for: parallelize(), textFile(), external data sources, and transformation from other RDDs.

Intermediate Level Questions (13-24)

13. Explain the difference between narrow and wide transformations with examples.

What to look for: Data shuffling, network overhead, pipeline optimization, map vs groupByKey examples.

14. What is shuffling in Spark and when does it occur?

What to look for: Data redistribution, network overhead, groupBy, join operations, and performance implications.

15. How do you optimize Spark jobs for better performance?

What to look for: Caching strategies, partition tuning, broadcast variables, avoiding shuffles, and serialization.

16. Explain the different types of joins available in Spark DataFrames.

What to look for: Inner, outer, left, right, semi, anti joins with performance characteristics and use cases.

17. What is the difference between cache() and persist() in Spark?

What to look for: Storage levels, memory vs disk, serialization options, and appropriate usage scenarios.

18. How do you handle null values in Spark DataFrames?

What to look for: na functions, dropna(), fillna(), isNull(), coalesce(), and data quality strategies.

19. Explain the concept of broadcast variables and when to use them.

What to look for: Read-only variables, memory efficiency, join optimization, and lookup table scenarios.

20. What are accumulators in Spark and provide use cases?

What to look for: Write-only variables, counters, debugging, monitoring, and custom accumulator types.

21. How do you read and write data in different formats using Spark?

What to look for: Parquet, JSON, CSV, Avro readers/writers, options, schema inference, and format-specific optimizations.

22. Explain the execution model of a Spark application.

What to look for: Job submission, stage division, task scheduling, executor allocation, and DAG execution.

23. What is the difference between DataFrame and Dataset in Spark?

What to look for: Type safety, compile-time checking, performance characteristics, and API differences.

24. How do you handle schema evolution when reading data in Spark?

What to look for: Schema merging, compatibility checks, data type evolution, and backward compatibility strategies.

Advanced/Difficult Level Questions (25-35)

25. Design a Spark application architecture for processing 10TB of data daily with fault tolerance.

What to look for: Cluster sizing, checkpointing, error handling, resource allocation, and monitoring strategies.

26. How would you optimize a Spark job that's experiencing frequent out-of-memory errors?

What to look for: Memory tuning, garbage collection, serialization, partition size adjustment, and caching strategies.

27. Explain how Spark's Catalyst optimizer works and how it improves query performance.

What to look for: Rule-based optimization, predicate pushdown, projection pruning, constant folding, and code generation.

28. How do you implement custom partitioning in Spark and when is it beneficial?

What to look for: Custom partitioner implementation, data locality, join optimization, and skew handling.

29. Design a solution for handling data skew in Spark transformations.

What to look for: Skew detection, salting techniques, bucketing, custom partitioning, and performance monitoring.

30. How would you implement incremental processing using Spark for a large dataset?

What to look for: Watermarking, checkpoint management, state handling, exactly-once processing, and delta processing.

31. Explain how you would debug and troubleshoot a slow-running Spark job.

What to look for: Spark UI analysis, stage analysis, task metrics, GC analysis, and bottleneck identification.

32. How do you implement custom UDFs (User Defined Functions) in Spark and what are the performance implications?

What to look for: UDF implementation, serialization overhead, vectorized UDFs, performance comparison with built-in functions.

33. Design a multi-stage ETL pipeline using Spark with proper error handling and monitoring.

What to look for: Pipeline orchestration, checkpoint strategies, error recovery, data validation, and alerting mechanisms.

34. How would you handle time-based operations and window functions in Spark?

What to look for: Window specifications, time-based aggregations, sliding windows, session windows, and performance optimization.

35. Explain how you would implement a custom data source in Spark.

What to look for: DataSource API, relation implementation, schema inference, predicate pushdown, and integration patterns.

Technical Deep-Dive Code Examples

Code Review A: RDD Operations

```
# Review and optimize this code:
rdd = sc.textFile("large_file.txt")
filtered_rdd = rdd.filter(lambda x: "error" in x.lower())
mapped_rdd = filtered_rdd.map(lambda x: x.split(","))
result = mapped_rdd.collect()

# What are the potential issues and how would you improve it?
```

Code Review B: DataFrame Operations

```
# Identify performance issues in this code:
df = spark.read.parquet("input_data")
df.groupBy("category").agg({"amount": "sum"}).show()
df.filter(col("date") > "2023-01-01").count()
df.select("id", "name").distinct().write.parquet("output")

# How would you optimize this pipeline?
```

Code Review C: Join Optimization

```
// Optimize this join operation:
val largeDF = spark.read.table("large_table") // 1TB
val smallDF = spark.read.table("small_table") // 100MB
val result = largeDF.join(smallDF, "key")
result.write.mode("overwrite").saveAsTable("result_table")
```

Scenario-Based Questions

Scenario A: Memory Management

"Your Spark job fails with OOM errors when processing 500GB of data. The cluster has 10 nodes with 32GB RAM each. How do you resolve this?"

Scenario B: Performance Optimization

"A daily ETL job that used to take 2 hours now takes 8 hours. The data volume has doubled. Walk me through your optimization approach."

Scenario C: Data Quality

"You need to process CSV files with inconsistent schemas and missing values. Design a robust Spark solution."

Scenario D: Real-time Processing

"Design a Spark Streaming solution that processes 100,000 events per second with exactly-once processing guarantees."

Hands-On Programming Challenges

Challenge 1: Data Transformation

```
# Given this DataFrame:
# +----+--------+------+
# | id | product|amount|
# +----+--------+------+
# |  1 |   A    | 100  |
# |  2 |   B    | 200  |
# |  1 |   A    | 150  |
# +----+--------+------+

# Write Spark code to:
# 1. Calculate total amount per product
# 2. Find the customer with highest total purchase
# 3. Add a running total column
```

Challenge 2: Complex Aggregation

```
-- Write Spark SQL to solve:
-- Find the top 3 products by sales in each region
-- Include percentage of total regional sales
-- Handle ties appropriately
```

Challenge 3: Window Functions

```
# Implement a solution to calculate:
# - 7-day moving average of sales
# - Rank customers by monthly purchases
# - Identify customers with declining purchase patterns
```

Performance Tuning Questions

36. Memory Configuration

- "How do you configure Spark memory settings for optimal performance?"
- "Explain the difference between executor memory and driver memory."

37. Serialization Optimization

- "When would you choose Kryo serialization over Java serialization?"
- "How do serialization choices impact Spark performance?"

38. Cluster Resource Management

- "How do you determine optimal cluster configuration for a given workload?"

- "Explain dynamic allocation and when to use it."

Integration and Ecosystem Questions

39. Data Lake Integration

- "How do you optimize Spark for reading from and writing to data lakes?"
- "Explain predicate pushdown with different storage formats."

40. Streaming Integration

- "How does Spark Structured Streaming relate to batch processing?"
- "Design a lambda architecture using Spark."

Follow-Up Questions:

- "How would you monitor Spark applications in production?"
- "What's your approach to testing Spark applications?"
- "How do you handle version compatibility when upgrading Spark?"
- "Explain your strategy for capacity planning in Spark clusters."
- "How do you implement data lineage tracking in Spark pipelines?"