13 Streaming Data with Spark

Streaming Data with Spark: Structured Streaming, Real-time Processing & Azure Event Hubs Integration

Basic Level Questions (1-10)

1. **What is Structured Streaming in Apache Spark and how does it differ from DStreams?** *Focus on: Unified batch/streaming API, fault tolerance, exactly-once processing, API evolution*
2. **Explain the concept of micro-batching in Spark Structured Streaming.** *Focus on: Batch processing model, latency considerations, trigger types, processing guarantees*
3. **What are the main components of a Structured Streaming application?** *Focus on: Input sources, transformations, output sinks, triggers, checkpointing*
4. **What is Azure Event Hubs and what role does it play in streaming architectures?** *Focus on: Message ingestion, partitioning, throughput units, retention, integration patterns*
5. **How do you read streaming data from Azure Event Hubs in Spark?** *Focus on: Event Hubs connector, connection strings, consumer groups, configuration options*
6. **What is checkpointing in Structured Streaming and why is it important?** *Focus on: Fault tolerance, recovery mechanisms, state management, exactly-once processing*
7. **What are the different output modes available in Structured Streaming?** *Focus on: Append, Complete, Update modes, use cases, performance implications*
8. **How do you handle late-arriving data in streaming applications?** *Focus on: Watermarking, event time vs processing time, data correctness*
9. **What are the common triggers available for Structured Streaming?** *Focus on: ProcessingTime, Once, Continuous, trade-offs between latency and throughput*
10. **How do you monitor the health and performance of a Structured Streaming application?** *Focus on: Streaming UI, metrics, lag monitoring, throughput analysis*

Intermediate Level Questions (11-20)

11. **How would you implement exactly-once processing guarantees in a Spark streaming application?** *Focus on: Idempotent operations, checkpoint recovery, source reliability, sink durability*
12. **Explain how to handle schema evolution in streaming data pipelines.** *Focus on: Schema inference, compatibility checks, graceful degradation, version management*

13. **How do you implement window operations in Structured Streaming for time-based aggregations?** *Focus on: Tumbling windows, sliding windows, session windows, watermarking*

14. **What strategies would you use to optimize Event Hubs integration for high-throughput scenarios?** *Focus on: Partitioning strategies, consumer group management, throughput units, batching*

15. **How do you handle backpressure and flow control in streaming applications?** *Focus on: Rate limiting, adaptive processing, queue management, resource allocation*

16. **Explain how to implement stream-to-stream joins in Structured Streaming.** *Focus on: Inner joins, outer joins, state management, watermarking, performance considerations*

17. **How would you implement a Lambda architecture pattern using Spark Streaming?** *Focus on: Batch layer, speed layer, serving layer, data consistency, complexity trade-offs*

18. **What are the best practices for managing state in stateful streaming operations?** *Focus on: State stores, memory management, checkpointing, state cleanup, performance*

19. **How do you implement custom sources and sinks for Structured Streaming?** *Focus on: Source/Sink interfaces, reliability guarantees, error handling, configuration*

20. **How would you design a streaming ETL pipeline that handles data quality issues in real-time?** *Focus on: Validation logic, error handling, dead letter queues, monitoring, alerting*

Advanced/Difficult Level Questions (21-30)

21. **Design a real-time fraud detection system using Spark Streaming and Event Hubs that can process millions of transactions per second.** *Focus on: Scalability architecture, low-latency requirements, state management, model deployment*

22. **How would you implement exactly-once processing for a complex multi-stage streaming pipeline with external system integration?** *Focus on: End-to-end guarantees, transactional semantics, compensation patterns, idempotency*

23. **Design a streaming data lake architecture that supports both real-time and historical analytics with consistent schema evolution.** *Focus on: Data organization, schema management, query optimization, storage formats*

24. **How would you implement a distributed streaming application that maintains global state across multiple Event Hubs partitions?** *Focus on: State partitioning, consistency models, coordination mechanisms, failure handling*

25. **Design a real-time recommendation engine using Spark Streaming that can handle concept drift and model updates.** *Focus on: Online learning, model versioning, A/B testing, performance optimization*

26. How would you implement a streaming data pipeline that guarantees ordered processing across multiple partitions? *Focus on: Ordering guarantees, coordination mechanisms, performance trade-offs, consistency models*

27. Design a multi-tenant streaming platform that provides isolation and resource management for different workloads. *Focus on: Resource allocation, isolation mechanisms, monitoring, SLA management*

28. How would you implement a streaming data pipeline that can handle varying data volumes and automatically scale resources? *Focus on: Auto-scaling strategies, resource optimization, cost management, performance monitoring*

29. Design a disaster recovery solution for critical streaming applications with minimal data loss and downtime. *Focus on: Cross-region replication, failover mechanisms, data consistency, recovery procedures*

30. How would you implement a streaming data governance framework that ensures compliance and data quality in real-time? *Focus on: Real-time validation, policy enforcement, audit trails, compliance reporting*

Real-time Processing Scenarios

Practical Implementation Challenges

Scenario A: Your streaming application is experiencing increasing lag during peak hours. Walk through your troubleshooting and optimization approach.

Scenario B: You need to join two high-volume streams with different arrival patterns while maintaining low latency. How would you design this?

Scenario C: Your Event Hubs integration is dropping messages during traffic spikes. How would you identify and resolve this issue?

Scenario D: You need to migrate a batch processing pipeline to streaming while maintaining data consistency. What's your approach?

Event Hubs Integration Patterns

Advanced Integration Techniques

Connection Management:

- Connection pooling strategies
- Retry policies and exponential backoff
- Circuit breaker patterns
- Health check implementations

Partition Management:

- Dynamic partition assignment
- Partition key strategies
- Load balancing across partitions

- Handling partition splits/merges

Error Handling:

- Dead letter queue implementation
- Poison message handling
- Retry mechanisms
- Circuit breaker patterns

Performance Optimization

Streaming Performance Tuning

Throughput Optimization:

- Batch size tuning
- Parallelism configuration
- Resource allocation strategies
- Network optimization

Latency Optimization:

- Trigger configuration
- Processing time minimization
- State management optimization
- Checkpoint frequency tuning

Resource Management:

- Memory optimization
- CPU utilization
- Storage optimization
- Network bandwidth management

Fault Tolerance & Recovery

Reliability Engineering

Failure Scenarios:

- Network partitions
- Node failures
- Data corruption
- Dependency failures

Recovery Strategies:

- Checkpoint-based recovery
- State reconstruction
- Partial failure handling

- Graceful degradation

Monitoring & Alerting:

- Lag monitoring
- Throughput tracking
- Error rate monitoring
- SLA compliance tracking