# Must Know Before Your Next Databricks Interview

**Document by – Siddhartha Subudhi**

Visit my LinkedIn profile

**1. Explain the architecture of Databricks.**

**Answer:**
Databricks operates on a distributed computing architecture. It uses Apache Spark's engine to run clusters. It consists of three layers:

- **Storage Layer**: Azure Blob Storage, Azure Data Lake.
- **Management Layer**: Handles cluster management, scaling, job scheduling.
- **Compute Layer**: Distributed Spark clusters where transformations and actions are executed.

---

**2. What are Delta Tables in Databricks?**

**Answer:**
Delta Tables are a storage layer that brings ACID (Atomicity, Consistency, Isolation, Durability) transactions to Apache Spark. It provides the following:

- Time-travel
- Data versioning
- Efficient updates and deletes
- Schema enforcement and evolution
- Optimized for batch and stream processing.

---

**3. How do you optimize a Spark job in Databricks?**

**Answer:**
To optimize a Spark job:

- **Use DataFrame APIs** over RDDs for optimizations.
- **Broadcast joins** for smaller tables.
- Use **Partition pruning** and **predicate pushdown**.
- Cache the DataFrames that are reused.
- **Use coalesce and repartition** wisely.
- Avoid **wide transformations** such as groupBy without using optimizations.
- Apply **cluster tuning** (choosing correct instance types and worker nodes).

---

**4. Explain spark.sql.shuffle.partitions and its impact on performance.**

**Answer:**
This parameter controls the number of partitions created during a shuffle operation in Spark SQL. The default is often 200, which can be inefficient for large datasets. Reducing this value can reduce shuffle write costs, whereas increasing it can balance the workload across the cluster for very large datasets.

---

**5. Describe a use case where Databricks Auto Loader is beneficial.**

**Answer:**
Auto Loader is beneficial when dealing with continuous data ingestion in near real-time. It automatically detects new files and increments the process. For example, if a company needs to ingest logs or IoT sensor data continuously into a Delta Lake, Auto Loader can handle file notification and ingestion efficiently.

---

**6. How would you handle slowly changing dimensions (SCD Type 2) in Delta Lake?**

**Answer:**
Use the **MERGE INTO** operation with Delta Lake to handle SCD Type 2. The merge operation compares the incoming dataset with the existing table and updates or inserts the records based on conditions, capturing historical changes.

---

**7. How do you debug a failed job in Databricks?**

**Answer:**

- Use the **Spark UI** to inspect job execution, DAG stages, and tasks.
- Check **logs** for error traces (driver, executor logs).
- Look at the **environment variables** and Spark configuration parameters.
- Analyze the **cluster's memory, storage, and network utilization**.

---

**8. How does Databricks handle job scheduling and orchestration?**

**Answer:**
Databricks allows job scheduling with triggers such as time-based, file arrival, or via REST API. For orchestration, Databricks supports **Jobs API**, **Databricks Workflows**, and can integrate with external orchestration tools like **Apache Airflow** or **Azure Data Factory**.

---

**9. Write a PySpark code to find the third-highest salary in a DataFrame.**

**Answer:**

```
from pyspark.sql import Window

from pyspark.sql.functions import col, dense_rank

windowSpec = Window.orderBy(col("salary").desc())

df_with_rank = df.withColumn("rank", dense_rank().over(windowSpec))

third_highest_salary = df_with_rank.filter(col("rank") == 3).select("salary")

third_highest_salary.show()
```

---

**10. What is the difference between coalesce and repartition in Spark?**

**Answer:**

- **Repartition** increases or decreases the number of partitions by shuffling the data.
- **Coalesce** is more efficient when reducing the number of partitions since it minimizes shuffling by moving data into fewer partitions without a full shuffle.

---

**11. How do you handle schema evolution in Delta Lake?**

**Answer:**
Delta Lake supports **automatic schema evolution** when writing data using the merge or write methods with the option mergeSchema=True. This allows adding new columns or changing data types without errors.

---

**12. How can you optimize a Delta Lake table for performance?**

**Answer:**

- Use the **OPTIMIZE** command to compact small files into larger files.
- Use **Z-order clustering** for faster query performance on specific columns.
- Regularly **VACUUM** to remove old data files and free up storage.

---

**13. Write a PySpark code to calculate the moving average for a column in a DataFrame.**

**Answer:**

```
from pyspark.sql import Window
from pyspark.sql.functions import avg

windowSpec = Window.orderBy("date").rowsBetween(-2, 0)  # 2-row window
df_with_moving_avg = df.withColumn("moving_avg", avg("value").over(windowSpec))
df_with_moving_avg.show()
```

---

**14. Explain the key differences between Azure Data Factory and Databricks.**

**Answer:**

- **ADF** is an orchestration tool for data movement and transformation, primarily focused on ETL workflows.

- **Databricks** is a Spark-based analytics platform for big data engineering and ML workloads.
- ADF is for scheduling and triggering data pipelines, while Databricks is for compute-intensive tasks like distributed data processing.

---

**15. How does Databricks handle incremental data loading?**

**Answer:**
With Delta Lake, you can load incremental data using the **MERGE INTO** statement or by leveraging **watermarking** in streaming data pipelines to capture new or modified data efficiently.

---

**16. How would you configure a cluster to run a Spark job on Databricks efficiently?**

**Answer:**

- Choose the correct **cluster mode** (Standard, High Concurrency, Single Node).
- Select **autoscaling** to automatically resize the cluster based on workload.
- Adjust the **instance type** based on memory or CPU needs.
- Configure **Spark parameters** for optimal performance, like shuffle partitions, memory, executor instances, etc.

---

**17. How do you handle data skew in Spark on Databricks?**

**Answer:**

- Use **salting** by adding a random number to keys before joining.
- **Broadcast smaller tables** to avoid skewed joins.
- Use the **skew hint** in SQL joins.

---

**18. How do you handle large file writes in Delta Lake?**

**Answer:**

- Use the merge or overwrite operations with partitions.
- Set up **auto-optimization** settings like compaction to reduce the number of small files.
- Use **Optimize** for file compaction after loading.

---

**19. How can you enable high availability in Databricks clusters?**

**Answer:**

- Use **multiple availability zones** for the clusters.
- Set up **auto-restart** in case of failures.
- Use **Databricks job retries** for fault tolerance.

---

**20. What are Databricks MLFlow's key components?**

**Answer:**

- **Tracking**: Tracks experiment metrics and parameters.
- **Projects**: Packaging of code in a reproducible way.
- **Models**: Model registry for deployment.
- **Registry**: Version control for machine learning models.

---

**21. How would you migrate on-premise ETL workflows to Databricks?**

**Answer:**

- Assess current workloads and dependencies.
- Rebuild the data pipelines in Databricks using **Spark APIs**.
- Use **Delta Lake** for storage and optimization.
- Automate the orchestration using **Azure Data Factory** or **Databricks Jobs API**.

---

**22. Explain the process of dynamic partitioning in Spark.**

**Answer:**
Dynamic partitioning allows Spark to create partitions based on data distribution at runtime. You can enable it using spark.sql.dynamicPartition and ensure efficient partition creation based on the key column values.

---

**23. How do you handle streaming data in Databricks?**

**Answer:**

- Use **Structured Streaming** with Apache Spark for real-time data processing.
- Use Delta Lake to handle streaming ingestion with **exactly-once guarantees** and schema enforcement.
- Configure **checkpointing** and **watermarking** for managing state and event time.

**24. How do you use Azure Key Vault with Databricks for secret management?**



**Answer:**
Integrate Azure Key Vault with Databricks using the **Databricks Secret Scope**. Secrets can be securely stored and retrieved via Key Vault by configuring a secret scope with access control.

---

**25. What are the common challenges faced in Spark cluster tuning?**

**Answer:**

- **Inefficient shuffling** leading to long-running stages.
- **Skewed data** causing imbalance in task distribution.
- **Memory mismanagement** (e.g., improper executor memory settings).
- Improper partition sizes.
- Not using **broadcast variables** for smaller tables during joins.