DATA PIPELINE FAILURES

Real Interview Scenarios & How to Handle Them

A practical guide for Data Engineers to answer real-world data pipeline failure questions with confidence

By Ankita Gulati

Senior Data Engineer | Interview Mentor

Interview Edition • Practical • Real Scenarios

Table Of Content

Scenario 1

Daily Spark Runtime Explodes

Problem Statement

You are responsible for a daily Spark batch job that processes approximately 1 TB of data.

- Normal runtime: 20 minutes
- Current runtime: 90 minutes
- Downstream business reports are blocked

The job does not fail but runs significantly slower than usual

Objective: Diagnose the root cause and bring the job back within the 30-minute SLA.

Clarifying Questions

Before proposing solutions, the engineer must reduce ambiguity.
Key questions include:

- Did the data volume or distribution change?
- Were there schema, join, or aggregation changes?
- Are all Spark stages slow or only specific ones?
- Do some tasks take much longer than others?
- Are executors underutilized or stuck waiting?

Clarifying Information & Assumptions

After initial investigation, the following facts are confirmed:

- Data volume increased by ~15%
- No changes in code, schema, or Spark version
- Cluster configuration remains unchanged
- No job failures or retries occurred

These assumptions indicate the slowdown is **not expected growth-related behavior**.

Key Observation

A **15% increase in data volume** should result in a small, near-linear increase in runtime.
A 4–5× runtime increase strongly suggests:

- Uneven data distribution
- Execution imbalance across Spark tasks
- Presence of straggler tasks

This shifts focus from scaling resources to execution behavior analysis.

Solution: Step-by-Step Deep Explanation

Step 1: Analyze Job Execution Using Spark UI

The first step is to inspect how Spark is executing the job, not how much data it

processes. Key indicators:

- Large gap between average task time and maximum task time
- One or two tasks running significantly longer than others
- Several executors idle while a few remain busy

This pattern confirms that a small subset of tasks is dominating total runtime.

Step 2: Understand Why Straggler Tasks Occur

Spark divides data into **partitions**, and each task processes one partition.

Stragglers occur when:

- Some partitions contain **disproportionately large data**
- Those partitions take much longer to process
- The entire waits for the slowest task to finish

This behavior is a classic symptom of **data skew**.

Step 3: What Data Skew Means Conceptually

Data skew happens when:

- A small number of keys appear extremely frequently
- Joins or aggregations group large volumes of data under those keys
- Spark assigns those heavy keys to one or few