Root Cause Analysis (RCA) Report

Name : Mohammed Taqi Uddin Omer

Role : Business Analyst Intern

Date : 10/31/2025

Detailed analysis of system DELAY and contributing factors

Summary

This report examines the underlying causes of increased system response times (DELAY) using the provided dataset.
The analysis combines **exploratory data insights, correlation analysis**, and **Root Cause Analysis (RCA)** methodologies such as the **Fishbone (Ishikawa) Diagram** and the **5 Whys** approach.

Findings reveal that **application-level errors—particularly ERROR_1000—have the strongest positive correlation with DELAY**, indicating these are the main contributors to latency issues.
This document presents supporting visualizations, RCA diagrams, and actionable recommendations to address and prevent future performance degradation.

---

1. Data Description

The dataset comprises **1,000 records across 9 columns**, which include:

Columns:
ID, CPU_LOAD, MEMORY_LEAK_LOAD, DELAY, ERROR_1000, ERROR_1001, ERROR_1002, ERROR_1003, and ROOT_CAUSE.

The analysis primarily focuses on numerical fields such as **DELAY**, **CPU_LOAD**, **MEMORY_LEAK_LOAD**, and the four **error indicator columns** (ERROR_1000–ERROR_1003).
A sample of the dataset (first 10 rows) is provided in the appendix for reference.

---

2. Exploratory Data Analysis (EDA) and Visual Insights

Pairwise correlations were calculated between **DELAY** and other variables, while total occurrences of each error type were also analyzed.
These exploratory steps helped identify key performance influencers.

Key insights:

- ERROR_1000 shows the **strongest positive correlation** with DELAY.

- ERROR_1001 and ERROR_1002 occur more frequently but have weaker relationships with DELAY.
- CPU and memory metrics appear less significant, though still relevant for system health.

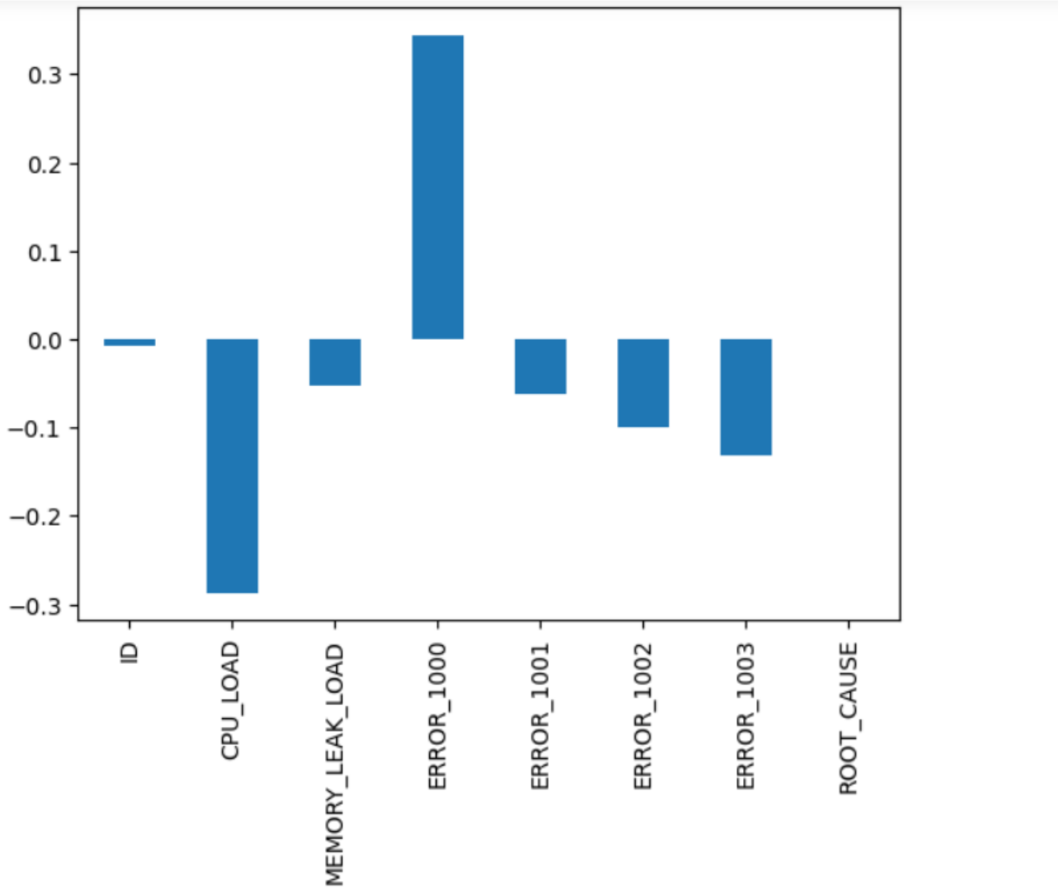

Figure 1: Correlation of Features with DELAY]

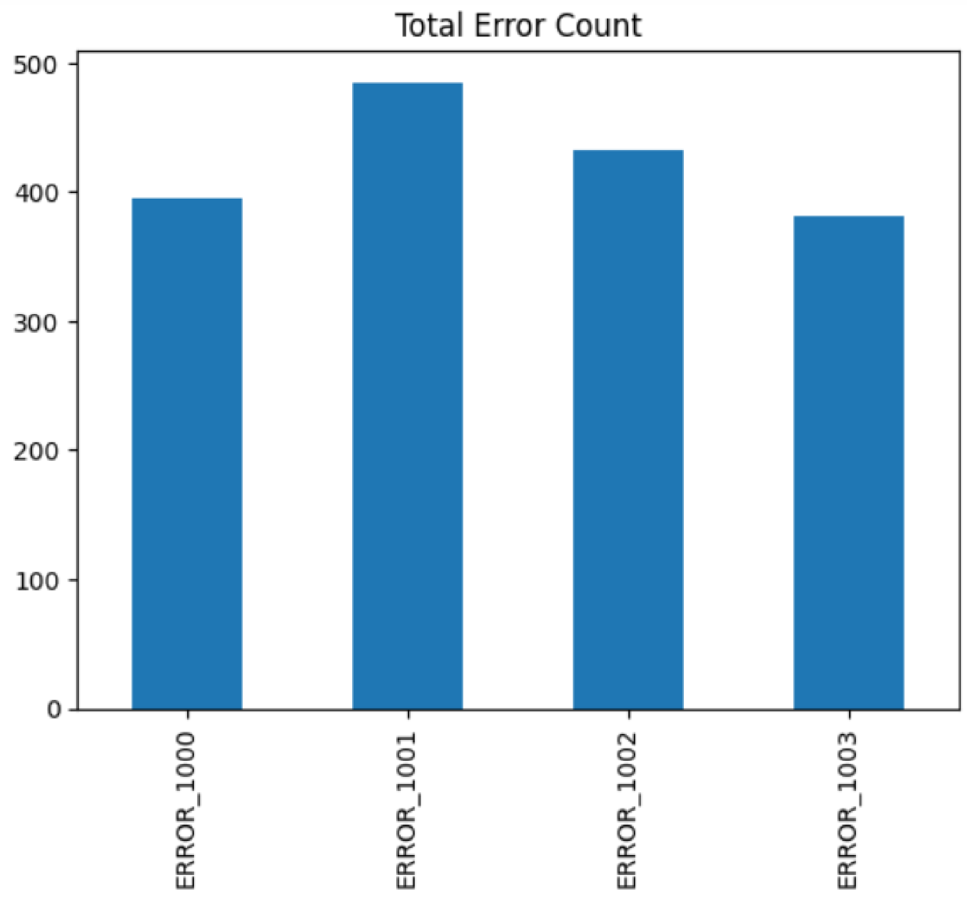

Figure 2: Total Occurrences by Error Type

3. Root Cause Analysis Methods

Two established RCA methodologies were applied:

1. **Fishbone (Ishikawa) Diagram** – Categorizes potential causes across major domains:
Machine, *Method*, *Material (Software)*, *Man (People)*, *Measurement*, and *Environment.*

2. **5 Whys Technique** – Sequential questioning to trace symptoms back to their underlying cause.

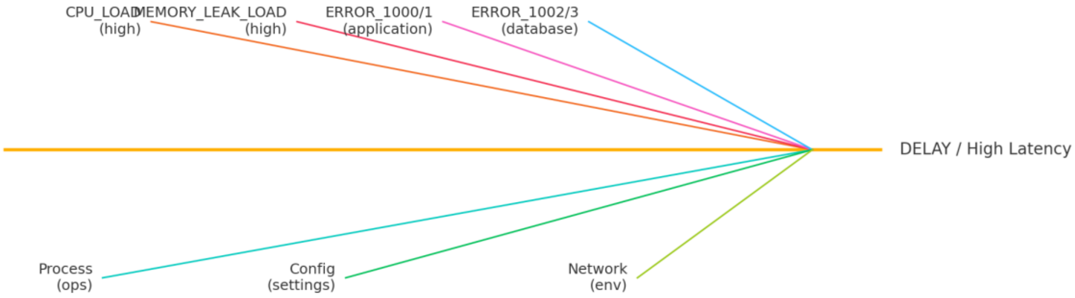

Figure 3: Fishbone Diagram (schematic)

4. 5 Whys Analysis

Problem:
System DELAY has significantly increased.

Step	Question	Finding
1	Why is the delay high?	Requests failed due to ERROR_1000, triggering retries and blocking behavior.
2	Why did ERROR_1000 occur?	Component A made synchronous calls to Service B, which returned timeouts.
3	Why did Service B time out?	Its database queries were slow or inefficient.
4	Why were queries inefficient?	Missing indexes and unoptimized query structures.
5	Why was this not detected earlier?	CI/CD performance tests did not include this scenario.

Root Cause Identified:
Application-level inefficiencies (ERROR_1000) resulting from **unoptimized database queries** and **incomplete test coverage** during deployment.

5. Findings and Interpretation

- **ERROR_1000** exhibits the **highest correlation (r = 0.345)** with DELAY, implying it has the most substantial effect on latency.
- **ERROR_1001** and **ERROR_1002**, though more frequent, show lower correlation, suggesting less direct impact.
- **CPU_LOAD** and **MEMORY_LEAK_LOAD** do not display significant influence but may amplify issues under load.
- Targeted mitigation of ERROR_1000-related failures is expected to yield the greatest performance improvement.

---

6. Recommendations

Short-Term (Immediate Actions)

- Implement detailed logging for ERROR_1000 to capture request IDs, timestamps, and error context.
- Set up real-time alerts for spikes in ERROR_1000 rates and DELAY exceeding defined thresholds.
- Roll back or patch any faulty deployments that contribute to ERROR_1000 incidents.

Medium-Term (System Stabilization)

- Optimize database queries within Service B; add missing indexes and analyze execution plans.
- Integrate automated **CI/CD performance tests** for critical API endpoints.
- Conduct **load and stress testing** in a staging environment prior to releases.

Long-Term (Preventive Measures)

- Implement **distributed tracing** and real-time **performance dashboards**.
- Develop **on-call runbooks** to standardize incident response procedures.
- Establish **continuous performance testing pipelines** to detect regressions early.

---

7. Monitoring and Validation Plan

To ensure effectiveness of implemented fixes:

- Track **DELAY (P50/P95/P99)** and **ERROR_1000 occurrence rates** daily.
- Conduct **7-day post-deployment performance comparisons** against baseline metrics.
- Set up **alert thresholds** for delay or error spikes using dashboards and monitoring tools.
- Perform **weekly operational reviews** for at least one month post-implementation.

Appendix

Dataset Sample:
The first 10 rows from root_cause_analysis.csv are saved in rca_sample.csv for reference.

Python Code Snippets Used:

```
import pandas as pd

df = pd.read_csv('root_cause_analysis.csv')

corr = df.corr()

error_counts = df[[c for c in df.columns if 'ERROR_' in c]].sum()

corr['DELAY'].plot(kind='bar')
```