إنتركي
INTERKEY

White paper

Architectural Strategies for Resilient, Storage-Aware CDC Pipelines

How an intelligent control plane keeps Change Data Capture running when storage fills up

Prepared by
Interkey

From Vision to Reality

1. Executive Summary

Change Data Capture (CDC) is essential for real-time data integration, yet standard implementations often fail due to unmanaged storage growth. In high-throughput environments, the accumulation of shadow tables in source databases and unconsumed messages in Kafka topics leads to "disk-full" scenarios that cause system-wide outages. This paper outlines a strategy for building an Intelligent Replication Controller that uses closed-loop feedback to manage storage and ensure 99.99% replication uptime.

2. The Problem: The "Triple-Storage" Burden

Traditional CDC tools operate on a "push-and-pray" model. They do not account for the physical disk limits of the transit layers.

1. **Source Side:** CDC sidecar tables or transaction logs expand if the capture process lags.
2. **Transit Side:** Kafka brokers store data across multiple replicas, multiplying the storage footprint of every database change.
3. **Consumer Side:** If the target (Data Warehouse/Lake) slows down, Kafka topics swell, creating a bottleneck that can crash the brokers.

3. Proposed Solution: The Intelligent Control Plane

The "Foolproof" tool must act as an orchestrator between the Source DB, Kafka, and the Destination.

3.1. Proactive Backpressure Mechanism

Instead of a linear flow, the tool implements a feedback loop.

- **Threshold Monitoring:** The tool continuously polls Disk I/O, Disk Space (%), and Consumer Lag.
- **Dynamic Throttling:** If Kafka disk usage exceeds a "Yellow Zone" (e.g., 75%), the tool reduces the ingestion rate (R). If it hits the "Red Zone" (85%), it pauses the source capture.

3.2. Automated Storage Recovery

The tool should not just monitor; it should act.

- **Triggered Cleanup:** When source DB storage is critical, the tool executes forced log truncation or CDC cleanup procedures.
- **TTL Shifting:** The tool can dynamically adjust the retention.ms property in Kafka. During high-load spikes, it can shorten retention to prioritize the most recent data and save the cluster.

4. Technical Components of the Tool

4.1. The Adaptive Initial Snapshot

To prevent the "Initial Dump" from crashing the system, the tool utilizes Watermark-based Chunking:

1. Divide the table into logical chunks based on Primary Key ranges.
2. Verify Kafka's available_bytes before starting each chunk.
3. Commit offsets only after the chunk is successfully acknowledged by the sink.

4.2. Metadata & Idempotency Layer

To ensure "foolproof" delivery, the tool must be Idempotent:

- **Global Offset Store:** Use an external store (e.g., Redis or Zookeeper) to track the High Watermark of the Source LSN (Log Sequence Number).
- **Deduplication Logic:** Every record is wrapped in a metadata header containing the Source Timestamp and Transaction ID, allowing the sink to ignore duplicates during a "crash-recovery" replay.

4.3. Schema Evolution Shield

The tool must intercept DDL (Data Definition Language) changes.

- **The Guardrail:** If a column is dropped or a data type is changed, the tool pauses and triggers an alert. This prevents "poison pill" messages from breaking the downstream consumer.

5. Mathematical Model for Capacity Planning

To prevent storage overflow, the tool uses the following formula to calculate the Safe Ingestion Rate (R):

$$R < \frac{Disk_{Available}}{Retention_{Time} \times Replication_{Factor}}$$

Where:

- **Disk_available** is the remaining capacity on the Kafka broker.
- **Retention_time** is the required window for the sink to recover from downtime.
- **Replication_factor** is the number of copies Kafka keeps of each partition.

6. Conclusion

A foolproof CDC tool is not defined by its speed, but by its awareness. By implementing a control plane that monitors storage health and exerts backpressure on the source, organizations can move from fragile, manual replication to a self-healing data infrastructure.