International Journal of Leading Research Publication (IJLRP)
E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

Reducing Latency in Cloud-Based Fuel Monitoring Systems

Rohith Varma Vegesna

Software Engineer 2
Texas, USA
rohithvegesna@gmail.com

Abstract

Cloud-based fuel monitoring systems are increasingly adopted to provide continuous oversight of station operations, harnessing automated data collection, rapid analytics, and near-real-time decision-making. These systems handle substantial data from Automatic Tank Gauge (ATG) devices and pump dispensers, making low latency crucial for accurate inventory tracking, timely leak detection, and rapid response to operational anomalies. Yet, designing an end-to-end pipeline that consistently achieves minimal latency, particularly under varying network conditions, poses considerable technical challenges.

This paper introduces an integrated approach that capitalizes on managed services within a well-established public cloud environment to lower round-trip times and processing delays. The proposed architecture utilizes edge processing for buffering and compression, reducing data volume and mitigating connection disruptions. The ingestion pipeline relies on fully managed, event-driven services to orchestrate serverless functions. These functions each serve as dedicated microservices, handling tasks such as anomaly detection, inventory reconciliation, and real-time alerts. Processed data is stored in low-latency databases for immediate access through customized dashboards, offering an operational view that updates within a matter of seconds.

A pilot study demonstrates the architecture's effectiveness, highlighting measurable gains in responsiveness and uptime when compared to legacy systems. Substantial reductions in latencycoupled with improved operational efficiencyunderscore the benefits of separating critical tasks into discrete microservices, leveraging agile deployment practices, and incorporating automated monitoring. This work concludes with a discussion of potential enhancements, including scaling to diverse station networks, integrating predictive analytics, and reinforcing security at both the edge and the cloud layers.

Keywords: Fuel Monitoring, Low Latency, Cloud Architecture, Edge Computing, Serverless Functions, Microservices, Real-Time Analytics

1. Introduction

1.1 Background

Fuel stations collect extensive, high-frequency data from both ATG sensors and pump dispensers. Traditional methods for logging and analyzing this information involved periodic data uploads or manual checks, often leading to lagged insights and delayed responses. The growing availability of

cloud-based technologies prompted station operators to migrate toward real-time or near-real-time monitoring solutions, enabling centralized dashboards and automated alerts.

However, achieving consistently low latency in these deployments demands careful attention to every phase of data handling. Latency can originate from suboptimal hardware interfaces, constrained network links, or inefficiencies in the cloud-side ingestion and processing pipeline. Event-driven microservices and edge computing methods have emerged as promising approaches for mitigating bottlenecks, yet the complexity of integrating these components often proves daunting.

Organizations now seek holistic, managed cloud solutions to expedite data transit while preserving accuracy and operational resilience. These solutions must remain adaptable to varying station sizes, transaction volumes, and network conditions without imposing unwieldy overhead or risk of data loss.

1.2 Problem Statement

Although cloud-based solutions offer the potential for real-time data visibility in fuel monitoring, latency commonly remains higher than desirable. This can lead to delayed detection of critical anomalies such as leaks, untracked discrepancies in dispenser output, or lags in inventory updates. In high-throughput environments, even modest latency can degrade situational awareness and impair decision-making. A more refined, low-latency architecture is requiredone that orchestrates data collection and transmission, cloud ingestion, parallel processing, and dashboard rendering with minimal overhead.

1.3 Objectives

This paper presents a cloud-native, latency-focused strategy aiming to:

- Pinpoint typical points of delay in current fuel station monitoring setups, spanning both on-site and cloud operations.
- Introduce an architecture that employs edge pre-processing, fully managed streaming services, and serverless microservices to tackle these delays.
- Validate the proposed framework in a production scenario, demonstrating measurable gains in responsiveness, scalability, and system reliability.

2. Literature Review

Prior investigations into fuel station monitoring underline the importance of real-time or near-real-time data reconciliation to enhance overall transparency and control. Studies on low-latency communication in Internet of Things settings point to event-driven paradigms and edge analytics as key enablers for rapid data processing. Research further demonstrates that distributing responsibilities among microserviceseach targeting a specific task such as dispenser data aggregation, ATG tracking, or anomaly detectioncan streamline workflows and improve operational throughput.

In parallel, technical documentation from managed cloud services highlights frameworks for ingestion, serverless processing, and scalable data stores. These resources detail how automatically triggered functions can process high-velocity data in real-time, removing traditional server management concerns and enabling flexible scaling. Combined, these efforts form the basis of an approach that integrates edge-level optimization, cloud-native ingestion, and carefully orchestrated microservices to achieve lower latency in demanding use cases.

3. System Architecture

- Edge Data Collection and Preprocessing
  - Local controllers receive and combine ATG sensor readings with pump dispenser transactions.
  - A lightweight application at the edge conducts initial validation, filters noise and compresses data to conserve bandwidth and reduce the cloud ingestion load.
- Managed Cloud Ingestion
  - A fully managed streaming service accepts incoming data in near-real-time, automatically scaling for throughput and capacity.
  - Event-driven functions process each data record or batch in parallel, enabling immediate transformations, anomaly checks, and other logic.
- Microservices-Based Processing
  - Function-level specialization isolates tasks, allowing focused microservices to handle inventory reconciliation, alert management, or usage analytics
  - A routing mechanism ensures that the right data goes to the relevant microservice without introducing additional overhead.
- Data Storage
  - A low-latency NoSQL store captures dynamic data for rapid retrieval, such as the latest dispenser transactions and tank levels.
  - A relational database houses historical or structured information, supporting complex reporting, audits, and archiving with minimal impact on real-time operations.
- Visualization and Monitoring
  - Interactive dashboards pull updates from the NoSQL store, displaying station conditions, open alerts, and inventory metrics in near-real-time.
  - A centralized monitoring service tracks performance metrics and logs, initiating alerts if latency exceeds defined thresholds or if unusual error rates appear.

4. Implementation Strategy

Implementation commenced by deploying an edge device that connected directly to each sensor and dispenser. This device served as the first line of data management, removing trivial records, compressing essential data, and batching it before sending to the ingestion layer. Once data arrived in the cloud, the streaming service automatically triggered multiple functions, each focusing on a distinct aspect of fuel station analytics.

For instance, one function performed quick checks against tank thresholds, alerting when readings deviated significantly from expected patterns. Another reconciled dispenser transactions with reported ATG levels, flagging potential discrepancies that might indicate losses or theft. A third handled usage analytics, logging throughput metrics to inform future maintenance schedules.

Data was then relayed to a NoSQL database designed for high-speed reads and writes, ensuring that the station's real-time dashboard displayed up-to-date figures. In parallel, a secondary workflow moved summarized records into a relational database at scheduled intervals for historical analysis. To manage

the system's health, a monitoring dashboard provided insights into metrics such as invocation durations, data throughput, and any anomalies that could signal network or processing slowdowns.

5. Case Study & Performance Evaluation

A mid-volume station distributing approximately 25,000 liters per day participated in a month-long pilot of the proposed architecture. The legacy monitoring setup, still active, offered a baseline for comparing latency, reliability, and responsiveness.

Operators relied on the new system's real-time dashboard for daily oversight, cross-verifying dispenser outputs with ATG readings. Alerts for unusual consumption spikes or rapid level drops were promptly displayed on-screen and forwarded via automated messages. This approach allowed staff to quickly address potential issuesranging from mechanical malfunctions to suspected leaksand provided valuable data to refine management decisions.

Key measurements included end-to-end latency, measured from the moment a dispenser event occurred until it appeared on the monitoring dashboard. Network bandwidth usage was also tracked to assess the impact of edge compression and filtering strategies. After the pilot, data logs were compiled and compared to those of the legacy system to quantify improvements.

6. Results and Discussion

6.1 Pilot Implementation

The pilot consistently achieved average latencies of 4 to 5 seconds, representing a notable improvement over the previous system, which often exhibited delays upward of 20 seconds. Users reported faster identification of discrepancies and near-immediate alerting on potential anomalies, providing time to investigate suspected fuel losses or sensor faults before they escalated. Compression and selective filtering at the edge resulted in roughly 15 percent lower bandwidth consumption, contributing to stable performance even during higher traffic periods.

Microservices allowed the system to handle peaks in specific operational tasks without diminishing the performance of others. For instance, a surge in pump transactions could trigger multiple function executions for dispenser data handling, while inventory reconciliation tasks maintained their low latency. By decoupling these processes, the architecture scaled effortlessly to match variable loads.

6.2 Performance Metrics

- **Latency**: Reduced from 20 seconds or more to an average of 4–5 seconds from dispenser event to dashboard display.
- **Bandwidth Optimization**: Achieved around 15% reduction in data transfer volume through local preprocessing and compression.
- **System Uptime**: Maintained a 99.8% rate, with minimal downtime confined to brief windows for maintenance updates.
- **Alert Responsiveness**: Operators received critical notifications within seconds, shortening potential incident response times from hours to minutes.

These findings underscored the viability of a serverless, event-driven approach that leverages managed cloud services and edge-level preprocessing to significantly improve fuel station monitoring.

7. Conclusion and Future Work

By integrating edge-based data collection, efficient streaming, and microservices tailored to specific analytical tasks, the presented architecture effectively minimizes latency in a cloud-based fuel monitoring environment. The pilot implementation confirmed substantial performance gains compared to traditional setups, illustrating the value of adopting a modular, low-latency design.

Future investigations may involve scaling this model to multiple geographically separated stations and evaluating the solution under diverse network constraints. Incorporating advanced analyticssuch as predictive maintenance informed by machine learningand enhancing end-to-end security protocols offer additional paths for refinement. Ultimately, this framework serves as a flexible blueprint for delivering near-real-time insights in high-volume, mission-critical fuel station operations.

8. References

1. Silva, Daniel &Asaamoning, Godwin &Orrillo, Héctor & Sofia, Rute & Mendes, Paulo. (2019). An Analysis of Fog Computing Data Placement Algorithms. 10.1145/3360774.3368201.
2. Fahs, Ali Jawad & Pierre, Guillaume. (2020). Tail-Latency-Aware Fog Application Replica Placement. 10.1007/978-3-030-65310-1_37.
3. Shao, Yanling& Li, Chunlin& Tang, Hengliang. (2018). A Data Replica Placement Strategy for IoT Workflows in Collaborative Edge and Cloud Environments. Computer Networks. 148. 10.1016/j.comnet.2018.10.017.
4. Ammar Awad Mutlag, Mohd Khanapi Abd Ghani, N. Arunkumar, Mazin Abed Mohammed, Othman Mohd,Enabling technologies for fog computing in healthcare IoT systems,Future Generation Computer Systems,Volume 90,2019,Pages 62-78,ISSN 0167-739X,https://doi.org/10.1016/j.future.2018.07.049.
5. Amazon Web Services. (2020). Amazon Kinesis Data Streams Developer Guide. Retrieved from https://docs.aws.amazon.com/kinesis/latest/dev/introduction.html
6. Amazon Web Services. (2020). AWS Lambda Developer Guide. Retrieved from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
7. Amazon Web Services. (2020). Amazon DynamoDB Developer Guide. Retrieved from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
8. Amazon Web Services. (2020). Amazon CloudWatch User Guide. Retrieved from https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html