Design and Implementation of a DataOps-Based Asynchronous Pipeline for Large-Scale Data Ingestion

Leonardo Afonso Amorim[1], **Vinicius Alboneti Aguiar**[1], **Sávio Salvarino Teles de Oliveira**[1], **Arlindo Rodrigues Galvão Filho**[1], **Ricardo Costa**[1], **Marcos Prado**[2]

[1] Instituto de Informática – Universidade Federal de Goiás (UFG),
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brazil

leonardoafonsoamorim@egresso.ufg.br,vinicius_aguiar2@discente.ufg.br,

savioteles@ufg.br,arlindogalvao@ufg.br,

ricardodouglasrodrigues@protonmail.com,,

marcos.prado@engdb.com.br

[2]Engineering Brasil - Rua Dr. Geraldo Campos Moreira, 375 - 10º andar
São Paulo-SP - CEP 04571-020 - Brazil

Abstract. *This paper presents the design and validation of a data operations (DataOps)-oriented asynchronous pipeline designed to optimize large-scale data ingestion into cloud data warehouses. The proposed solution addresses ingestion bottlenecks caused by large file transfers through a combination of strategic file fragmentation and parallel processing using serverless services, including Google Cloud Functions and Cloud Run. Rather than introducing a new architectural model, the contribution lies in demonstrating the practical effectiveness of integrating these cloud-native techniques under real-world conditions. Experimental results show a 52.8x speed-up in ingestion time, reducing the processing of a 2TB dataset from 11 days to just 5 hours. The pipeline also incorporates Infrastructure as Code (IaC), Continuous Integration/Continuous Deployment (CI/CD), and automated monitoring, delivering a scalable, reproducible, and cost-efficient ingestion strategy well-suited for real-world enterprise data workflows.*

1. Introduction

The growing demand for integrating large volumes of data into cloud-based data warehouses has exposed limitations in traditional synchronous ingestion pipelines, including high transfer latency, limited scalability, and increased operational complexity [Kleppmann 2017]. These issues become critical in environments that require near real-time processing of large datasets [Grover and Pal 2025].

Traditional data pipelines based on synchronous processing often struggle with scalability since each stage depends on the completion of the previous one. This sequential model becomes particularly inefficient in high-volume scenarios, where maximizing parallelism and minimizing idle resources are key to maintaining performance

[Reis and Housley 2022]. Additionally, ensuring data consistency, traceability, and governance is critical, especially in enterprise environments with regulatory or operational constraints [DAMA International 2017]. In response to these demands, asynchronous pipelines have emerged as a viable alternative, enabling decoupled execution of tasks, event-driven processing, and optimized resource utilization. Cloud-native services, such as serverless computing platforms, further enhance these architectures by offering elastic scalability and low operational overhead [Casale et al. 2021].

This work presents a practical evaluation of an asynchronous pipeline designed to optimize large-scale ingestion into Google BigQuery. Rather than proposing a new architecture, the contribution lies in assessing the effectiveness of a solution built upon established cloud-native services and DataOps practices, such as Infrastructure as Code (IaC), Continuous Integration/Continuous Deployment (CI/CD), and automated monitoring. The pipeline applies file fragmentation and parallel serverless execution to accelerate ingestion and improve scalability. The experimental results, obtained using a 2TB dataset, show a 52.8x reduction in ingestion time compared to a synchronous baseline.

We organized this paper as follows. Section 3 discusses related work and how other approaches tackle the challenges of large-scale ingestion. Section 4 provides an overview of the evaluated pipeline, describing its components and implementation. Section 5 summarizes key mechanisms that support asynchronous processing. Section 6 presents the experimental setup and performance results. Finally, Section 7 discusses conclusions and outlines directions for future work.

2. Background

Integrating large and heterogeneous datasets into cloud data warehouses frequently faces critical challenges such as high network latency, ingestion workflow bottlenecks, and escalating operational costs. These challenges become particularly evident when organizations rely on monolithic, synchronous transfers of large files, exacerbating resource contention and increasing vulnerability to single points of failure. Traditional ingestion pipelines typically adopt a rigid, sequential execution model, in which each processing step must complete before the next begins. While this design simplifies orchestration, it restricts scalability, hinders real-time responsiveness, and amplifies operational risks as data volumes grow exponentially.

Modern cloud environments demand architectures that are scalable and adaptive to fluctuations in workload and infrastructure conditions. Asynchronous ingestion pipelines respond to these demands by decoupling execution stages, enabling parallel task processing, and leveraging event-driven triggers to orchestrate data flows. This approach minimizes idle time, increases throughput, and improves system resilience to partial failures. Furthermore, asynchronous designs lend themselves naturally to horizontal scaling and on-demand resource allocation, characteristics that are essential for high-volume enterprise workloads.

Within this paradigm, DataOps emerges as a key enabler for operational excellence. Extending DevOps principles to the data domain, DataOps emphasizes continuous integration, automated testing, observability, and governance across the entire data lifecycle. In cloud-native ecosystems, Infrastructure as Code (IaC) facilitates reproducible environment provisioning, while Continuous Integration/Continuous Deployment (CI/CD) pipelines automate updates and reduce the risk of misconfigurations. Combined with real-time monitoring and alerting, these practices strengthen compliance, data quality assurance, and the traceability of ingestion processes—requirements increasingly demanded by regulatory and corporate governance frameworks.

Serverless computing platforms, such as Google Cloud Functions and Cloud Run, further amplify these benefits by abstracting infrastructure management, providing near-infinite concurrency, and billing based on actual usage rather than pre-allocated resources. This elasticity allows ingestion pipelines to dynamically scale with workload intensity, avoiding overprovisioning and reducing costs. When paired with file fragmentation strategies—splitting massive files into optimally sized chunks for parallel loading—serverless architectures can significantly reduce ingestion times and improve resource efficiency.

In this work, we evaluate a real-world implementation of an asynchronous ingestion pipeline that integrates these principles to accelerate the loading of large-scale data into Google BigQuery. Our approach combines fine-grained file fragmentation, serverless execution, and DataOps-driven automation to deliver a scalable, auditable, and cost-efficient ingestion process. By leveraging a fully managed, serverless data warehouse designed for high-speed SQL analytics on massive datasets, the pipeline demonstrates how modern architectural and operational practices can overcome the inherent limitations of traditional ingestion methods.

3. Related Work

Recent studies have addressed scalable data ingestion, asynchronous processing, and the adoption of DataOps in cloud-based data architectures.

Rahman et al. [Rahman 2024] proposed an asynchronous architecture to enhance data integration in Apache Hive by utilizing parallel landing and staging processes. While aligned with our use of parallelism, their approach targets Hadoop/Hive environments and lacks DataOps integration.

Manchana [Manchana 2024] discussed DataOps practices for bridging legacy and modern systems, emphasizing the importance of automation and governance. However, no concrete implementation of asynchronous processing is evaluated.

Chavan [Chavan 2024] applied DataOps to signature verification systems, emphasizing real-time orchestration. Our work differs in that it targets ingestion performance in cloud data warehouses, with a focus on file fragmentation and parallel execution.

Bui [Bui 2024] presented a real-time ELT pipeline on GCP using Pub/Sub and Dataflow. Unlike Bui, we evaluate ingestion performance through asynchronous processing and serverless architecture, emphasizing latency reduction and cost optimization.

Other efforts, such as ORBITER [Loureiro and de Oliveira 2022] and FAIR-compliant repositories [Castro and Aguiar 2023], address deployment automation and metadata governance, respectively. However, they do not evaluate ingestion performance in the context of large-scale cloud data warehouses.

In summary, while prior work explores automation, parallelism, or governance, this paper provides a practical evaluation of a DataOps-driven, asynchronous ingestion pipeline designed to target performance improvements in Google BigQuery.

4. Pipeline Overview

The evaluated architecture leverages Google Cloud Platform (GCP) services to implement an asynchronous pipeline for ingesting large volumes of data into BigQuery. The flow begins when CSV files are uploaded to Google Cloud Storage (GCS), triggering a Cloud Run service that orchestrates the data processing steps.

The first step in the pipeline involves consolidating multiple source CSV files into a single, larger file of approximately 90MB. This aggregation process reduces the overhead associated with handling a high number of small files, which often causes metadata management issues and transfer inefficiencies. Grouping data into well-sized blocks enhances ingestion performance and aligns with Google BigQuery's best practices for optimizing input file size [Google 2024].

Following the consolidation, the next step involves loading the data into a BigQuery table. After the upload, the original files are either archived or deleted from GCS based on a predefined data retention policy. This stage ensures that storage usage is optimized and that the data becomes readily available for analytical queries.

The entire infrastructure is provisioned using Infrastructure as Code (IaC) and automated deployment scripts, which guarantee consistent configuration, scalability, and secure management of the involved services, including Cloud Functions, Cloud Run, and IAM policies.

Figure 1 illustrates the pipeline, highlighting its modular and scalable nature. Cloud Run manages the orchestration of asynchronous tasks, allowing each processing step to be handled independently, thereby promoting efficient resource utilization.

Moreover, the system includes support for status callbacks, enhancing observability and traceability, which reinforces its ability to provide scalable, resilient, and automated ingestion of large datasets into BigQuery. It forms a robust foundation for the performance evaluation discussed in the subsequent sections[1]

5. Pipeline Implementation Details

Building upon the architectural overview, the implementation details the use of Google Cloud services to enable asynchronous and scalable ingestion of large datasets into BigQuery. The key components are three Cloud Functions and a Cloud Run service, each with clearly defined responsibilities in the pipeline.

5.1. CSV Concatenation and Transfer

The first step in the ingestion pipeline involves receiving a JSON payload that specifies the target files to be processed. These files, stored in a Google Cloud Storage (GCS) bucket, are then read and concatenated in memory into a single file of approximately 90MB using DataFrames. We chose an intermediate file size to balance ingestion efficiency in BigQuery with storage cost optimization. After concatenation, the pipeline applies configurable runtime policies to either move or delete the source files. By minimizing file fragmentation and avoiding unnecessary disk writes, this step reduces ingestion latency and contributes to a more efficient overall data transfer process.

[1]The implementation of the asynchronous pipeline is publicly available at: `https://github.com/leonardoamorim/gcp-assync-f1b5.git`

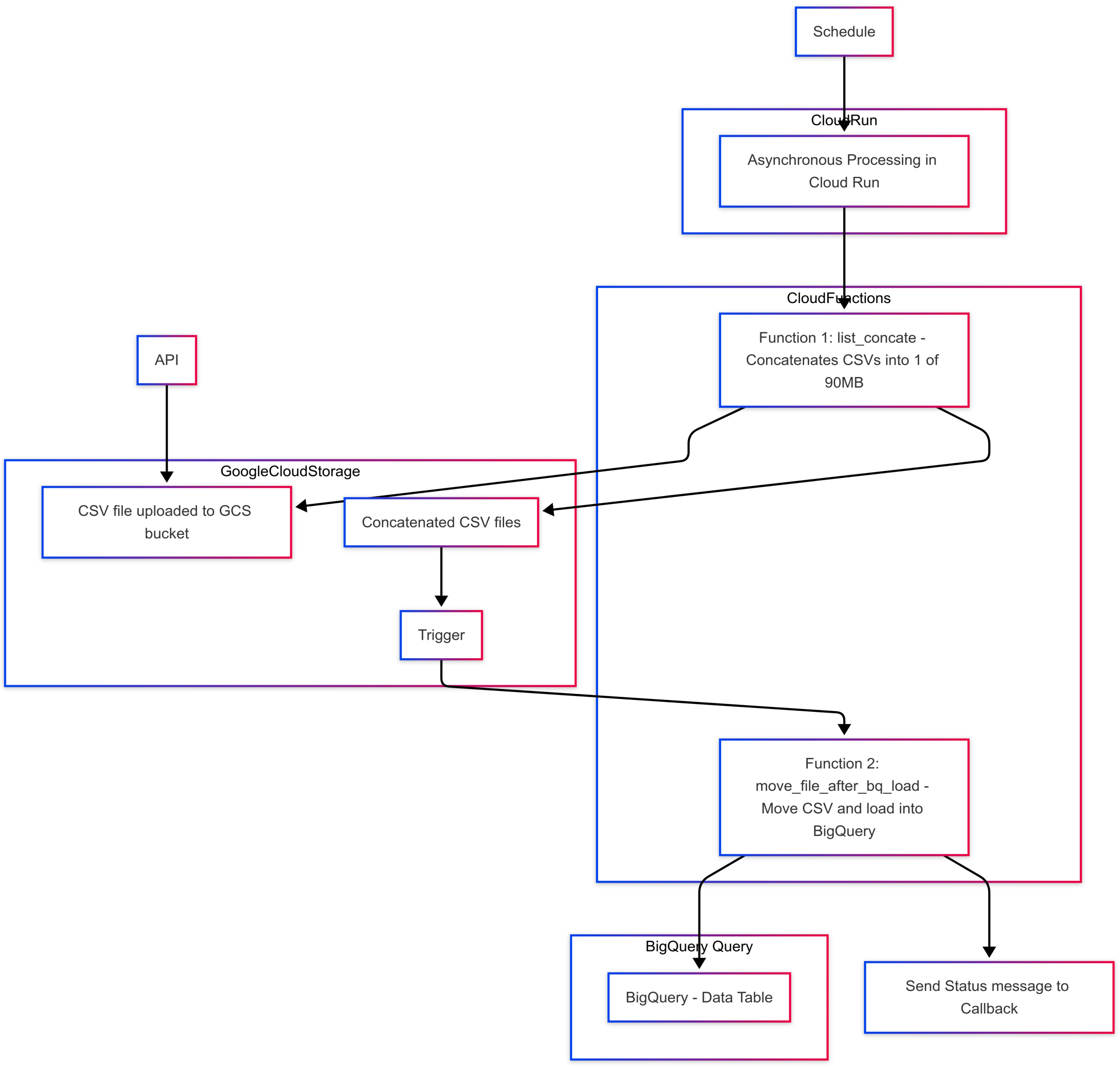
Schedule
CloudRun
Asynchronous Processing in Cloud Run
CloudFunctions
Function 1: list_concate - Concatenates CSVs into 1 of 90MB
API
GoogleCloudStorage
CSV file uploaded to GCS bucket
Concatenated CSV files
Trigger
Function 2: move_file_after_bq_load - Move CSV and load into BigQuery
BigQuery Query
BigQuery - Data Table
Send Status message to Callback

Figure 1. Overview of the Asynchronous Ingestion Pipeline using Cloud Functions and Cloud Run.

5.2. BigQuery Ingestion and File Management

The next step in the pipeline is triggered by the presence of a newly generated, consolidated CSV file. This file is automatically loaded into a target BigQuery table, with the schema inferred dynamically from its contents. Once the data is successfully ingested, the pipeline proceeds to manage the source file according to a predefined retention policy—either deleting or archiving it in Google Cloud Storage. In parallel, any errors or anomalies encountered during the load process are recorded in a monitoring table to support auditing and debugging. This sequence of actions ensures automated and efficient ingestion, minimizes the need for manual intervention, and strengthens data governance practices.

5.3. Callback and Table Consolidation

In the final step of the ingestion pipeline, metadata generated throughout the process is recorded to track execution status and support observability. The pipeline then consolidates temporary staging tables into a final, structured BigQuery table, ensuring that the data is properly organized and ready for analysis. Depending on the configured write mode, any intermediate or auxiliary files are deleted to maintain a clean environment. These actions complete the ingestion cycle, ensuring not only data availability but also consistent, end-to-end structuring within BigQuery.

5.4. Deployment Automation

The deployment process is fully automated through shell scripts that configure Cloud Functions and Cloud Run using parameters from a ‘config.json‘ file. Functions are provisioned with appropriate concurrency, memory, and triggers, while Cloud Run deploys a Docker image for orchestration. Logging and validation steps ensure reliable setup and support reproducibility across environments.

6. Experiments

To validate the efficiency of the proposed asynchronous data pipeline, we conducted experiments focusing primarily on ingestion time as the key performance metric, from which we derived the overall speed-up. The objective was to compare the pipeline’s performance against a traditional synchronous approach when processing large volumes of data, demonstrating how asynchronous execution and parallelism impact end-to-end ingestion time. Although the architecture supports scalability and fault handling, these aspects were not quantitatively evaluated in this study.

6.1. Experimental Setup

The experiments were conducted in a controlled cloud environment using Google Cloud Platform (GCP) to evaluate the scalability and efficiency of the proposed approach. We processed a 2TB dataset and addressed potential ingestion bottlenecks by strategically fragmenting files into 90MB segments. This method improved ingestion efficiency, enabled parallel processing, and reduced overall transfer time. The key infrastructure components included Google Cloud Storage (GCS) for storing incoming CSV files, Cloud Functions for handling file concatenation and ingestion, Cloud Run for orchestrating the asynchronous pipeline, and BigQuery as the target data warehouse.

6.2. Performance Metrics

The primary performance metric evaluated in this study was ingestion time, measured from the moment the data was transferred to Cloud Storage until it was fully loaded into the BigQuery table. This end-to-end measurement was used to compute the speed-up achieved by the asynchronous pipeline when compared to the traditional synchronous baseline.

While the pipeline is designed to support horizontal scalability, cost optimization, and fault tolerance, these aspects were not the focus of the current evaluation and remain as areas for future investigation. By concentrating on ingestion time, we aimed to isolate the impact of asynchronous execution, file fragmentation, and parallel processing on the overall data loading performance.

6.3. Experimental Results

The experimental results demonstrated a substantial improvement in ingestion performance enabled by the asynchronous architecture. When processing a 2TB dataset, the proposed pipeline reduced ingestion time from 11 days to 5 hours, resulting in a 52.8x speed-up compared to the baseline synchronous approach. This performance gain highlights the effectiveness of the file fragmentation strategy, which allowed for parallel data ingestion while avoiding system overload. The use of serverless components further contributed to efficient resource allocation and reduced latency throughout the pipeline.

7. Conclusion

This paper presented the design and validation of a DataOps-oriented asynchronous pipeline for large-scale data ingestion into Google BigQuery. Instead of proposing a new architectural model, the contribution lies in implementing and empirically evaluating a practical, cloud-native solution based on modular serverless components—namely Cloud Functions and Cloud Run. The ingestion strategy utilizes file fragmentation to minimize transfer latency and facilitate parallel processing in high-volume data scenarios.

Experimental results demonstrated a substantial performance gain, reducing the ingestion time of a 2TB dataset from 11 days to just 5 hours—a 52.8x speed-up. This outcome validates the effectiveness of the asynchronous design and the chosen file fragmentation strategy, effectively mitigating bottlenecks and enabling efficient parallel execution. The dynamic scalability of serverless components ensured high availability and optimal resource utilization across varying workloads. Beyond performance, the study highlights the role of DataOps practices—such as Infrastructure as Code (IaC), Continuous Integration/Continuous Deployment (CI/CD) pipelines, and automated monitoring—in enhancing automation, traceability, and operational governance. These practices contribute to the pipeline's maintainability, reproducibility, and cost-efficiency in production cloud environments. The use of on-demand infrastructure also helped minimize costs by avoiding resource overprovisioning.

The solution has been validated in a real-world enterprise environment and is currently under consideration for production deployment. In future work, we plan to expand support for additional file formats, strengthen data governance mechanisms, and explore advanced DataOps capabilities for automated validation and quality assurance. We also aim to investigate the impact of different fragmentation sizes and compare the proposed architecture with other asynchronous ingestion frameworks, such as Google Dataflow and Apache Spark Streaming. To enrich this comparison, we intend to simulate or estimate the ingestion performance of these distributed frameworks in terms of time, scalability, cost, and operational complexity. Finally, we plan to evaluate the pipeline's scalability with data volumes exceeding 5TB, further validating its robustness in high-throughput enterprise scenarios.

Beyond its technical implications, this work contributes to the broader discourse on how DataOps can transform the governance of data-intensive workflows in hybrid and multi-cloud ecosystems. Integrating automation, observability, and asynchronous design principles lays the foundation for intelligent data platforms capable of self-optimization and continuous compliance. As organizations increasingly adopt AI-driven analytics, such pipelines become essential enablers for sustainable and transparent data operations.

References

Bui, T. (2024). Real-time elt pipeline architecture in google cloud. Master's thesis, Oulu University of Applied Sciences.

Casale, G., Artac, M., Van-Gool, W., Majo, V., Weerasinghe, D. A. B., Vithanage, P. N. A. M. E., Pu, Y., Varghese, B., and Chandrasiri, K. K. R. G. K. (2021). A survey on serverless computing. *Journal of Cloud Computing*, 10(1):30.

Castro, J. and Aguiar, C. (2023). Big data architectures for fair-compliant repositories: A systematic review. In *Anais do XXXVIII Simpósio Brasileiro de Bancos de Dados*, pages 76–88, Porto Alegre, RS, Brasil. SBC.

Chavan, M. (2024). Integrating dataops practices in signature verification systems for seamless data orchestration. 2:49–64.

DAMA International (2017). *DAMA-DMBOK: Data Management Body of Knowledge*. Technics Publications, 2nd edition.

Google (2024). Bigquery standard sql syntax – create model statement. `https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-create`. Accessed: 2024-03-17.

Grover, V. and Pal, P. (2025). Ingesting insights: Data ingestion strategies and techniques for marketing data. In Balusamy, B., Grover, V., Nallakaruppan, M., Rajasekaran, V., and Milanova, M., editors, *Data Engineering for Data-driven Marketing*, pages 47–57. Emerald Publishing Limited, Leeds.

Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

Loureiro, J. and de Oliveira, D. (2022). Orbiter: um arcabouço para implantação automática de aplicações big data em arquiteturas serverless. In *Anais do XXXVII Simpósio Brasileiro de Bancos de Dados*, pages 379–384, Porto Alegre, RS, Brasil. SBC.

Manchana, R. (2024). Dataops: Bridging the gap between legacy and modern systems for seamless data orchestration. Technical report, SRC/JAICC-137. DOI: doi.org/10.47363/JAICC/2024 (3) E137 J Arti Inte . . . .

Rahman, M. A. (2024). Boosting hive efficiency: A novel dual-process architecture for asynchronous and parallel data loading. Master's thesis, Brac University, Department of Computer Science and Engineering, Brac University.

Reis, J. and Housley, M. (2022). *Fundamentals of Data Engineering: Plan and Build Robust Data Systems*. O'Reilly Media.