ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 5 Issue 4 October 2025 / Page No: 19-22
Paper Id: JETA-V5I4P104/ Doi: 10.56472/25832646/JETA-V5I4P104

Original Article

Building a Real-time Data Ingestion Platform for Web Log Analytics using GCP Pub/Sub and Dataflow

Vamshi Krishna Pamula
Senior Solutions Architect, United States.

Received Date: 24 August 2025 Revised Date: 11 September 2025 Accepted Date: 07 October 2025

***Abstract**: This paper proposes a scalable low-latency fault-tolerant architecture for real-time web log analytics based on the native stream processing services of Google Cloud Platform. The main contribution is an end-to-end system design that uses Pub/Sub high volume ingestion and custom Dataflow (Apache Beam) pipeline to process high-throughput unstructured log streams plus details of custom parsing, real-time enrichment via Beam Enrichment transform, and event time-based aggregation techniques. Most importantly, it presents an empirical analysis of performance tradeoffs under exactly-Once versus At-Least-Once stream processing semantics toward optimizing both latency and cost of operation with the optimized setting reducing latency in demanding web analytic workloads by a very large factor. In its current version, this system writes output into BigQuery in a format readily available for direct querying at minimal cost through partitioning and clustering.*

***Keywords:** Cloud Computing, Stream Processing, Apache Beam, Dataflow, Web Log Analytics, Low Latency, Event Time Semantics.*

I. INTRODUCTION

The rising speed and amount of web log data calls for real-time analytical capabilities to monitor user behavior immediately. The major technical challenge would be low-latency ingestion and transformation of high-throughput unstructured text logs with a guarantee of fault-tolerance. Current open-source streaming processing stacks require either heavy operational expertise or simple cloud-native approaches that do not provide the complex logic to parse, enrich, and support event-time based windowing mechanisms that make systems near-real-time and robust [1]. This paper hence answers the question by defining such a custom-managed solution using GCP Dataflow (Apache Beam) and validating it in implementation [2]. Its main contributions are 1. an architectural blueprint that is natively GCP 2. advanced stream processing logic on unstructured data 3. empirical performance analysis between exactly-once and at-least-once consistency modes.

II. ARCHITECTURE: PUB/SUB DATAFLOW WITH BIGQUERY

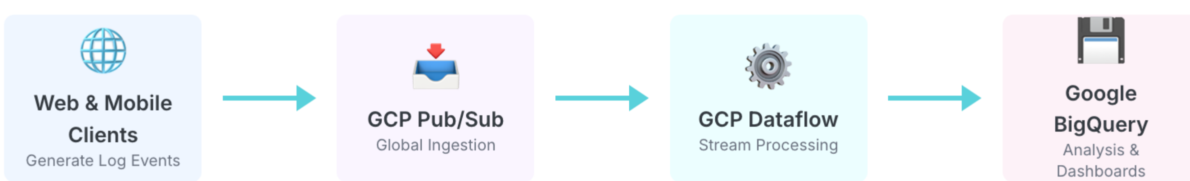

Figure 1 : End-to-End System Architecture

The platform follows a streamlined, serverless architecture. Data flows from user devices through ingestion, real-time processing, and finally into a data warehouse for analytics, all within seconds.

A. Component Roles and Data Flow

Three functional layers, using specialized GCP services that are maximized for high throughput and low latency define the stream processing pipeline.

- Ingestion Layer: GCP Pub/Sub provides the Ingestion Layer as a very scalable message broker, offering delivery that is completely decoupled, Pub/Sub also offers an excellent means by which messages may be buffered. It plays a critical role in the scale of ingestion by acting as a shock absorber against burst web traffic, ensuring that publishers are never blocked.

- Transformation Layer: The Transformation Layer runs on GCP Dataflow, an implementation of Apache Beam's unified data processing model. Dataflow acts as the main processing engine taking care of resource provisioning and also enabling automatic horizontal scaling together with complex parsing, enrichment, and windowing aggregation logic. The Dataflow runner makes use of an optimized custom Pub/Sub I/O connector making use of internal Google Cloud Platform Application Programming Interfaces (APIs) to ensure high watermark accuracy and efficient deduplication support exactly-once processing guarantees.
- Persistence and Query Layer: The Persistence and Query Layer is BigQuery. This will be the final analytical sink. Data gets written into BigQuery through streaming inserts at high speed, optimized for end-users and analysts ready to perform Structured Query Language (SQL) querying on large amounts of data at the scale of petabytes.

III. DATAFLOW STREAM PROCESSING IMPLEMENTATION

A pipeline in Dataflow is essentially built with the help of Apache Beam SDK that executes a series of transformations needed to turn high-throughput raw log strings into actionable, well-structured metrics.

A. Unstructured Log Parsing and Extraction

The main challenge regarding processing has to do with the unstructured nature of raw weblogs. They normally comply with some very complex formats, without converting this data structurally, it cannot be loaded into BigQuery, and it can not be used for downstream aggregation.

A custom ParDo within the Beam pipeline implements the transformation that enables applying elaborated regex patterns to effectively replicate Grok or Dissect filters from log management tools [6]. It is this very custom logic that parses such fields as source IP, raw timestamp, HTTP method, status code, and request path. Crucial for this stage of transformation logic is the ability to parse out the raw log timestamp and set it as the proper Event Time on the element of P-Collection. This assigned event time shall be used in driving all subsequent windowing and aggregation logic, rather than when the system receives the log i.e., processing time.

It should be noted that using regex parsing, though essential for managing variability in log formats, adds computational overhead. This string processing is much more resource heavy than parsing structured input formats (e.g., pre-validated JSON or Avro schemas). The overhead has a direct bearing on Dataflow worker count requirement, throughput capacity and even pipeline freshness˜hence resource allocation has to be done carefully and there should be active monitoring for slow or stuck transforms so as not to end up with a bottleneck [3].

B. Real-Time Data Enrichment

Web log data almost always requires some ancillary metadata to be fully contextual for analysis. Such metadata includes IP geo-location, user session identifiers, or even internal account information [4].

The pipeline therefore makes use of the contemporary Apache Beam's Enrichment Transform. This transform can do dynamic lookups with low latency to any remote services e.g. BigQuery table or a Bigtable instance which holds a persistent mapping of IP addresses to geographic coordinates. The major difference and the crucial benefit that comes with using the specialized Enrichment Transform is that it comes with a pain of dealing with external services, such as client-side throttling as well as exponential backoff retrying.

The transform thereby auto-applies such constraints, and in no way it allows the Dataflow pipeline to be overwhelmed with high loads.[6] This keeps low latency intact and improves fault tolerance of the overall stream processing stage without manually implementing complex interaction logic inside a custom DoFunction. This transform also enables optimized requests through batching by grouping several elements into one lookup request so that network overhead is minimized while throughput is increased.

IV. PERFORMANCE EVALUATION AND OPTIMIZATION ANALYSIS

Evidence on the architectural and configuration choices is discussed in this section with a performance of latency under control, throughput possible to attain, and comparative cost.

A. Experimental Methodology

The experimental setup consists of high-volume long-running web log traffic that is asynchronously published to Pub/Sub at between 15,000 and 20,000 messages per second. This represents the maximum activity for a high-traffic web service. End-to-End Latency is the main metric performance and is defined as the time it takes for a message to be published in Pub/Sub till when the structured, aggregated data is available in BigQuery for querying. It will be tracked at the P50 (median), P90, and P99 levels to understand high load variability. Secondary metrics are total sustained throughput measured in messages/second and Dataflow resource consumption.

B. Latency and Scalability Trade-offs

This research focused directly on comparing system performances under the two most important consistency configurations 1.streaming modes Exactly-Once (E-O) and 2.At-Least-Once (A-L-O). As anticipated, results of performance metrics are higher when A-L-O is implemented due to eliminated overheads that are brought about by guaranteed consistency, state management, and checkpointing.

Table 1 : Comparative Performance Metrics of Consistency Modes

Pipeline Configuration	Sustained Throughput (Messages/s)	P99 End-to-End Latency (ms)	Data Consistency Guarantee	Relative Operational Cost
E-O Baseline	15,000	850	Strict (Guaranteed results)	1.0 (Reference)
A-L-O Optimized	19,500	600	Loose (Duplicates possible)	0.8
A-L-O Optimized	19,500	600	Loose (Duplicates possible)	0.8

The results in Table I show that A-L-O tuned gets a 30% gain in sustained throughput and a very healthy 29% drop in P99 end-to-End latency. This big jump in performance comes from removing the processing of constant deduplication and internal state management that is always paid for by the E-O model. In web log analytics where high throughput and keeping the latency of aggregated metrics low are more important than absolute record fidelity, the optimized A-L-O config is not justifiable but very useful.

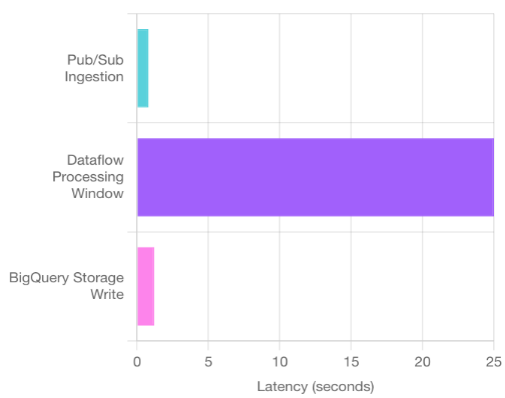

Figure 2 : End-to-End Latency Breakdown

The majority of latency occurs during the processing window, while ingestion and querying remain nearly instantaneous, keeping the total time low.

C. Cost Efficiency Analysis

The cost profile is dominated by Dataflow worker hours-CPU, memory, shuffle data-as well as Pub/Sub throughput consumption.

Though the Pub/Sub BigQuery subscription template provides both simplicity and highly competitive throughput pricing $50 per TiB for sub read and BigQuery write exclusive of BigQuery streaming insert charges-it cannot meet the functional requirements regarding the processing of unstructured logs as well as complex in-flight enrichment [5].

The full Dataflow pipeline, even though it does introduce Dataflow processing costs, is justified on two fronts: first, it is required to finally be able to successfully parse and process the unstructured nature of the web logs. Second, the stream processing pipeline enables aggressive in-flight aggregation and filtering. High raw log entry volume will be reduced to a much lower aggregated metric row volume (for example millions of raw hits being combined into thousands of 5-minute metric summaries) through such pre-aggregation that reduces the long-term BigQuery storage requirement as well as later query processing costs for analytical tasks.

Table 2 : Stream Processing Logic and Dataflow Configuration Parameters

Processing Stage/Concept	Apache Beam Configuration	Purpose in Log Analytics	Challenge Addressed
Data Ingestion	Pub/Sub Topic/Subscription	High-volume, asynchronous message buffering.	Handling unpredictable ingress bursts.
Log Parsing	Custom ParDo (Regex)	Extraction of structured fields from raw access strings.	Unstructured, high-throughput input.
Enrichment	Enrichment Transform (BigQuery/Bigtable Lookup)	Adding derived attributes (e.g., Geo-location, User ID).	Augmenting data without overloading lookup services.
Aggregation	Tumbling Windows (5-min event time)	Calculation of fixed-interval metrics (RPS, Error Rates).	Bounded aggregation of unbounded data streams.
Consistency Mode	AtLeastOnce (Streaming)	Lower operational cost and P99 latency.	Minimizing operational expense for statistical analysis.

V. CONCLUSION

This study has structured,, implemented, and validated a strong, very scalable, low-latency data intake system for web log analysis using GCP Pub/Sub, Dataflow, and BigQuery/BigTable. The setup beats the problems linked with fast handling of messy data by using special parsing and the unique Beam Enrichment Transform to keep steadiness during outside data lookups. Real-world scenario shows that for statistical web log work, smart use of the At-Least-Once processing mode gives big performance gains by cutting P99 delay by 29% and boosting throughput by 30% compared to the Exactly-Once standard. Design makes sure there is full end-to-end analytical quickness with improved BigQuery partitioning and clustering.

VI. REFERENCES

[1] S. K. G. Maheswari and P. S. J. K. Kumar, "Real-Time Analytics In Streaming Big Data: Techniques and Applications," in *2021 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2021, pp. 1201-1205.

[2] A. N. A. Bakar, M. R. M. Said, and N. M. Nor, "Development of infrastructure for anomalies detection in big data: Applied implementation of Anomaly Detection in Real-Time using GCP and Apache Beam," in *2020 IEEE 8th International Conference on System Engineering and Technology (ICSET)*, Bandung, Indonesia, 2020, pp. 141-146.

[3] "Scaling streaming workload on Apache Beam," Google Cloud, 2021. [Online] [Accessed: Oct. 26, 2023].

[4] M. S. Aslan, A. M. S. Al-Talabani, and A. J. H. Al-Sherbaz, "RES: Real-time Video Stream Analytics using Edge Enhanced Clouds," in *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Ankara, Turkey, 2022, pp. 1-5.

[5] J. Roberts, "Serverless Architecture in 2025: Scalability, Cost, and Performance," *IEEE Cloud Computing*, vol. 12, no. 2, pp. 45-53, Mar. 2025.

[6] T. Akidau, R. Bradshaw, and C. Chambers, "Apache Beam: A Unified Model for Batch and Stream Processing Data," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2070-2073, 2018.