Scholar Publishing Group
International Journal of Business Management and Economics and Trade
https://doi.org/10.38007/IJBMET.2026.070302
ISSN 2790-0959 Vol. 7, Issue 3:10-22

Research on Distributed Data Pipelines and Event-Driven Architecture for Freight Visibility

Haoran Xin

College of Computing, Georgia Institute of Technology, Atlanta, Georgia, United States

Keywords: Freight visualization; Distributed data pipeline; Event-driven architecture; Multimodal transport; Stream processing; State machine

Abstract: Around the world, freight organisations are moving away from single-point transport management to a multi-entity, multi-node, cross-system collaborative model. Freight visualisation platforms need to offer continuous status information on all parts of an order, such as orders, vehicles, cargo, ports, warehouses, customs and customer service. To address the problems of delayed data arrival, inconsistencies in status descriptions, insufficient anomaly alerts, and high system expansion costs, a data-distributed pipeline and event-driven architecture model suitable for multimodal transport scenarios have been established. Event domains in the freight lifecycle include order creation, loading, in transit, arrival, transshipment, receipt and anomaly closure. The design will focus on event themes, streaming cleansing, state machine aggregation, idempotent writing and observable governance. Prototype load test logs are used to verify the end-to-end latency, visualisation gaps, event consistency and anomaly response capabilities. Based on the above analysis, after implementing a Kafka-Flink-style event pipeline, the P95 event latency dropped from 9.42s in the polling integration mode to 1.31s, the gap in abnormal event visualisation reduced from 58.6% to over 41.4%, and the state consistency still exceeded 99.30%. This study offers a reproducible architectural blueprint and engineering evaluation indicators for third-party logistics, port collaboration, cross-border transportation platforms and supply chain control tower construction.

1 Introduction

Problems of instability in the world's supply chain have occurred repeatedly rather than just once. Problems of cross-border transportation include port congestion, bad weather conditions, traffic jams in the road network, changes in energy prices, modifications to trade policies, etc.; as a result, freight companies are finding it increasingly difficult to obtain full-time, real-time cargo information solely through traditional transport management systems. Freighting visualisation's basic idea is not just to show the position of a vehicle; rather, it should be able to build an understandable state flow that includes order commitments, carrier resources, sensor data, node operations, anomaly handling and customer feedback. Recently, research on Logistics 4.0 has increasingly focused on the advantages of real-time status sharing in terms of efficiency,

transparency and resilience, with topics such as intelligent tracking devices, RFID, IoT, digital twins and open freight platforms [1-4]. However, in terms of business applications, most visualisation creations are simply front-end displays and dashboards that do not build an underlying data structure suitable for high-concurrency events, asynchronous modifications and cross-system semantic consistency.

Freight visualisation has the general characteristics of distributed systems. A single waybill may be involved by multiple parties, such as shippers, carriers, trucking companies, ports, warehouses, shipping companies, customs systems and consignees. Each entity has its own record of the event, and data sources such as GPS, temperature and humidity, electronic seals, RFID, EDI, TMS, WMS and ERP are in heterogeneous forms. If the system is still managed by methods such as timed polling, centralised database synchronisation and batch ETL, then there will be a significant delay in the "collection-cleaning-warehousing-display-alerting" process. The visualisation platform shows the most recent status; however, it may be up to ten minutes old and thus late for anomaly intervention. Stream Processing Frameworks and Event-Driven Architectures have provided new technical pathways for addressing the above issues. Event producers only publish status changes, and consumers subscribe to topics for incremental processing based on streaming computation and state machine rules to achieve "event-based perceptibility" rather than "query-based visibility".

The above studies have all taken different approaches to this problem. Helo and Thai believe that intelligent tracking devices can provide real-time information on the supply chain to improve logistics efficiency, transparency and security [1]; Fazi and Fransoo have introduced the idea of Freight Mobility as a Service to study the role of open platforms in building an open-carrier service-matching system [2]; Abril et al. developed an event-driven real-time architecture for smart port operation scheduling planning and found that event-induced tactical-operational collaboration can raise the decision-making level of port information systems [3]. At the same time, research on the scalability of stream processing frameworks has also found that, in the cloud environment, frameworks such as Flink and Kafka Streams can handle high-throughput data streams, but selecting different stream processing frameworks will result in different impacts on resource costs and operational complexity [5]. Although the above studies have provided a theoretical basis for the event-based transformation of freight visualization, a systematic full-cycle freight framework, definitions of indicators and an engineering verification system have not been established.

Therefore, the research has the following three goals: first, to build a distributed freight event model and define the semantic boundaries of state events, location events, sensor events and anomaly events; second, to develop a data pipeline that supports low latency, replayability, auditability and scalability to enable the visualization system to complete all stages from data collection to alerting; and third, to create evaluation indicators for practical engineering applications by using latency distribution, visualization gaps, out-of-order ratio, state consistency and anomaly response time to measure the efficiency of the architecture. Based on the above goals, a visualisation architecture of "event theme - streaming cleansing - state machine aggregation - service subscription" and five core formula metrics have been proposed. Two statistical charts and three sets of comparison tables have been extracted from prototype load-testing logs as quantitative references for the following enterprise deployment.

The structure of this article is as follows: Part 1 is the research background and motivation; Part 2 is the current research situation of distributed data pipelines, event-driven architecture and freight visualization; Part 3 is the key contradictions in the current system construction; Part 4 presents the architecture scheme, state model, indicator system and experimental results; and Part 5 summarizes the research conclusions and future development directions.

2. Current Status Analysis of the Research Topic

2.1 Freight visualization shifts from location tracking to status intelligence

Early freight visualisation mainly used GPS positioning and electronic fences to determine if vehicles were near nodes, but did not show the location of goods. As the demands of shippers for delivery guarantees, abnormal alerts, and customer experience have continuously increased, the objects of visualisation have also evolved from geographical locations to operational statuses, commitment fulfillment, asset security and risk prediction. Based on the literature review of the Internet of Things (IoT) in logistics and supply chain management, it has been determined that this field of study has gone through an early exploratory stage and is now in a mature application phase. RFID, blockchain, 5G, artificial intelligence and industrial Internet of Things are all being developed to enhance traceability and secure exchange [4]. Therefore, the freight visualisation cannot be regarded as a standalone device access project but rather as a system engineering project that incorporates data governance, semantic modelling and real-time computing.

Multimodal transportation is about status intelligence rather than location tracking. For example, if the same vehicle is parked in the port area, it may be waiting to enter the gate, be inspected, be loaded, be detained, or have already been handed over; if the coordinates of the same container have not changed, it may be in normal storage or it may be detained. Only when the location event is linked to the order commitment, node plan, sensor alarm and operating rule can it be determined whether it is in a normal state. Research on the digital twin of the supply chain believes that real-time data, RFID, tracking frameworks and IoT sensors can be employed to identify problems and issue early warnings, thus enhancing the visibility, agility and resilience of the supply chain [6]. Freight visualisation platforms should be changed from "data display systems" to "status calculation systems" as a result.

2.2 Distributed data pipelines become the infrastructure for multi-source access.

Freight data sources are very heterogeneous and unstable. TMS generates order and planning data; WMS generates inbound, outbound and loading/unloading data; vehicle terminals generate location streams; cold chain equipment generates temperature and humidity streams; port and customs systems generate node and inspection status data; customer service systems generate problem orders and feedback data. These systems have diverse interface protocols, timestamp formats, status enumerations and data quality; therefore, the old batch ETL mode cannot provide real-time monitoring. Distributed data pipelines use message queues, stream computing, a schema registry, state storage and a data lake to convert data from all sources into a unified event stream.

Stream processing frameworks are suitable for early-stage processing of data to perform cleaning, deduplication, aggregation, window functions and anomaly detection upon arrival. Henning and Hasselbring have conducted benchmark tests on the scalability of microservice-based stream processing frameworks in the cloud, and found that with abundant resources, all frameworks show linear growth; however, their resource consumption, deployment cost and suitable environments are different [5]. Freight visualisation shows that in the architecture design, in addition to throughput, other factors such as event out-of-orderness, status window, replay capability, end-to-end semantics, and operation-and-maintenance controllability must also be taken into account. Cross-enterprise collaboration cases are typical of the platform's applications; they generally have both a large volume of high-frequency sensor data and low-frequency business status data. Therefore, the design of the pipeline should support both hierarchical routing and elastic scaling.

2.3 Event-driven architecture drives the system from synchronous calls to asynchronous collaboration.

Event-driven architecture considers a system to be a record and communication unit based on events. Event-driven architecture is not request-response based and is relatively loosely coupled, asynchronous, highly scalable, and replayable. In a freight system, all of the following can be considered events: "order creation", "vehicle arrival", "loading completion", "temperature exceeds limit", "estimated arrival time update", and "customer signature". Producers do not know about the service consumption events, and consumers are not blocked; thus, cross-system coupling is reduced.

Studies on smart ports have shown that event-driven models and operational optimisation frameworks can improve the speed of response by port information systems to congestion and operating plans [3]. This way can also be applied to multi-modal transport; that is, when the vehicle arrives at the port area, the shipping schedule is adjusted; or if there is a problem with temperature control, the platform will automatically recalculate the ETA, inform the customer, assess the risk, and offer scheduling options without manual intervention. The Apache Flink official documentation lists event-driven applications, batch processing and data pipelines as the main use cases, and introduces event time processing, late data handling, low latency and high throughput capabilities [7]. Therefore, the event-driven architecture can change freight visualisation from the original "Kanban-style display" to the current "proactive operation".

2.4 Insufficient existing research

Although the above studies have shown that IoT, RFID, AI and digital twins can increase the transparency of logistics [1, 4, 6, 8] and stream processing frameworks can support extended real-time computing [5, 7], three deficiencies in the research on freight visualization architecture still remain. First, the research discusses IoT, RFID and blockchain separately from the perspective of individual technologies, and does not integrate all these components in an end-to-end system such as event themes, state machines, pipeline governance and observability. Second, freight status has lifecycle dependencies and cross-node semantics; simply recording events is not sufficient to ensure state consistency, and formalised state transition rules are also required. Third, most evaluation indicators are related to business performance or system throughput; there is a lack of an engineering indicator system that can uniformly measure latency, out-of-order delivery, idempotent consistency, visualization gaps and abnormal closed loops. Therefore, the following discussion will focus on the three items listed below: "How to define the event model, How to organise the data pipeline, How to converge the state, and How to evaluate the effect [9-15]."

Table 1. Event Domains and Data Sources for Freight Visualization

Event domain	Typical sources	Core attributes	Update frequency	Visibility value
Order event	TMS, ERP, customer portal	order_id, lane, promise_time, carrier	low	contractual baseline
Location event	GPS device, mobile app, telematics	asset_id, longitude, latitude, speed	high	real-time movement
Node event	terminal, warehouse, port system	node_id, operation_code, gate_time	medium	process milestone
Sensor event	IoT tag, reefer, e-seal	temperature, humidity, shock, seal_status	high	cargo condition
Exception event	alert engine, customer service	severity, reason_code, handler, SLA	burst	risk intervention

As shown in Table 1, freight visualisation cannot be realised by a single location stream. Order events are the basis for fulfilment, location events indicate the transportation process, node events show the stage of operation, sensor events display the condition of goods, and anomaly events form the chain of processing. The frequency of the events varies widely. The same processing strategy will be used for high-frequency data, leading to resource wastage in the processing of business status, and low-frequency critical events will be lost in the noise. Therefore, distributed pipelines should create topics, retention policies, window sizes and consumption priorities according to the domain of events to ensure that high-frequency data collection and low-frequency decision-making run stably [16-20].

3. Raise questions

3.1 Data arrival delays cause visualization distortion

The most typical problem of a visualisation system is that "the interface displays normally, but the actual situation is abnormal". Several factors can cause latency, such as local device caching, mobile network jitter, API polling cycles, data cleaning queuing, batch database writes, and front-end refresh intervals. The old architecture updates only every few minutes; it is suitable for financial settlement and report analysis but too slow to detect abnormal events in time. When the cold-chain temperature exceeds the upper and lower limits, the vehicle will be off-route; if the port-waiting time exceeds the maximum, and this is not rectified within five minutes of learning about the problem, there will be very large losses for further treatment.

Therefore, latency should be considered a core quality metric for the visualization platform, rather than a secondary performance metric. For the *i*event, its end-to-end event latency can be defined as:

$$L_i=t_i^{sink}-t_i^{event}$$

In equation (1), t_i^{event}represents the time when the event occurs at the business site or is collected by the device, and t_i^{sink}represents the time when the event is written to the visual status storage and becomes available for querying. This metric differs from the interface response time; it covers the entire process of collection, transmission, cleaning, calculation, and writing. The platform should focus on P95 and P99 latency, as a small number of long-tail events are often the source of anomaly warning failures [21-25].

3.2 Out-of-order and repetitive events disrupt state consistency

Freight events are naturally disorganized. A vehicle will send an "Arrived at warehouse" message when it reaches the warehouse and then a "Leaving highway service area" message. The port system will not return to the "Inspection Completed" state until the inspection task has been finished. However, once an inspection task is set to "Inspection Started", a new inspection signal is sent, and this may lead to an error in the state machine's decision. Duplicate events also occur frequently; mobile retries, gateway retransmissions, interface timeout compensation and manual re-entry can all generate duplicate messages. If the system does not have watermarks, idempotent keys and event version control, older but later events may overwrite newer ones, causing duplicate anomalies and multiple notifications.

Window analysis can be performed on the proportion of out-of-order events.

$$R_o(W)=\frac{\sum_{i\in W}\mathbf{1}\,(t_i^{event}<\tau_W)}{|W|}$$

In equation (2), *W*is the observation window, au_W is the window water level, $\mathbf{1}(\cdot)$and is the indicator function. When the event time is earlier than the current water level, it can be identified as a late event. This indicator can help the platform set a reasonable lateness tolerance interval: if the tolerance interval is too short, valid resend data will be lost; if it is too long, it will increase the state convergence delay.

3.3 Semantic inconsistencies make cross-subject collaboration difficult.

Different parties use different codes for the same state. For example, "Loaded" in a carrier's system, "Outbound" in a warehouse system, and "Picked up" in a customer portal can all indicate that the goods have left the shipping node. However, "Gate-in" in a port system and "Received" in a shipping company system may be out of sync at different times in various business situations. Field mapping only integrates the state; therefore, its semantic consistency cannot be guaranteed. Standardize the event vocabulary and state transition rules for freight visualization, and set an anomaly level standard to determine whether an event belongs to the lifecycle of the system.

3.4 Centralized architectures are difficult to support elastic growth

Freight platform traffic has a peak and a trough. There are generally some short-term spikes in the number of visits due to special activities, seasonal festivals, high-volume days at the port, abnormal weather, significant shifts in overseas direct investment policy, etc. Centralized database writes or monolithic service integrations have queue backlogs during peak hours and are therefore slow. Research has shown that horizontal scaling of distributed stream processing frameworks in cloud environments can increase throughput; however, the choice of distributed stream processing framework and resource allocation and deployment will affect costs. Therefore, the freight visualisation system should be partitioned thematically, expand the consumer group, separate the state backend, apply hot and cold data layering, and introduce service circuit breaking to achieve real-time performance within a cost limit [26-30].

4. Problem Solving and Strategies

4.1 Overall Architecture Design

The five modules of the proposed architecture are: an acquisition and access layer, an event bus layer, a streaming processing layer, a status service layer, and a visualization application layer. The acquisition and access layer connect to TMS, WMS, GPS, IoT tags, port systems, EDI gateways and customer portals. Organize the business domain of the event bus layer and use a schema registry to manage the structure of events. The streaming processing layer is responsible for deduplication, error correction, window calculation, state machine aggregation and anomaly detection. The status service layer saves the most recent statuses of waybills, vehicles, container cargo and abnormal work orders, as well as event logs for playback and audit. The visualisation application layer publishes and subscribes to status updates for the control tower, customer portal, dispatch console and open API.

The three Design Concepts are as follows. First, event priority. All state changes enter the platform as events, and batch interfaces are only a source of supplementary data. Second, Replayability. The most recent state is no longer a manually maintained isolated field; instead, it is obtained through event stream processing and a state machine. The system will show the old state before rule modification. Third, the Government is incommunicado. Platform metrics are used to monitor data quality, latency, out-of-order delivery, failure retries, consumption backlog and

anomaly closure continuously, and no separate monitoring is needed after deployment.

Table 2 Distributed Event Pipeline Modules and Evaluation Metrics

Pipeline module	Main responsibility	Key metric	Failure risk	Governance strategy
Ingestion gateway	protocol adaptation and authentication	accepted events per second	burst rejection	Token bucket and backpressure
Event broker	topic routing and durable log	consumer lag	partition hotspot	hash key and rebalance
Stream processor	cleansing, join, window, state transition	P95 latency	late event overwrite	watermark and state version
State store	latest shipment status query	query freshness	stale snapshot	materialized view refresh
Alert service	exception scoring and notification	response time	duplicated alert	idempotent alert key

Table 2 is the engineering division of the pipeline modules. The access gateway transforms data from various protocols into events; the event bus ensures the persistence and replayability of events; the stream processor conducts state calculations; the state storage provides query functions; and the alarm service establishes an anomaly closed loop. All modules have their own functional responsibilities and measurable indicators. The main purpose of this table is to switch from a component-stacking approach in architecture design to a metric-driven model, and thus specific modules of the platform can be identified during scaling, troubleshooting and performance optimisation.

4.2 Event Model and Theme Planning

The unified envelope structure of the event model includes event_id, event_type, entity_id, entity_type, event_time, source_system, version, payload and trace_id. Among them, event_id is used for idempotency, event_time is for event time calculation, and trace_id is for cross-system tracing. The plans of topics are divided according to the type of entity and event, such as shipment.status, asset.location, cargo.sensor, node.*operation*, and risk.exception. High-frequency sensor events have a shorter retention time and a separate partition; business-status events, however, need to be stored for auditing over a longer period.

A measure of the age of a displayed state is known as the state-freshness index.

$$F_T=\frac{1}{N}\sum_{j=1}^{N}\exp\left[-\lambda\left(T-t_j^{last}\right)\right]$$

In equation (3), Nrepresents the number of visualized objects, $T$is the current evaluation time, is the time $t_j^{last}$of the object's $j$most recent valid state update, $\lambda$and is the decay coefficient. The closer the value of this indicator is to 1, the closer the platform state is to real-time; when a large number of objects have not been updated for a long time, the index will drop rapidly. Compared with simply counting "how long it has been since the last update", this indicator can continuously evaluate the freshness of the global state.

4.3 Flow cleaning and handling of late arrival incidents

The steps in the streaming cleaning process are: format validation, field normalisation, anomaly removal, time zone unification, event deduplication and coordinate correction. Do not ignore the late event; rather, it should also be handled according to the type of event. Location events that are late by more than the threshold generally only meet the criteria for completion via trajectory completion and will not be updated to the latest state. Node events that are considered important milestones can serve as compensation streams for rollbacks or corrections in the state machine. The delayed sensor events can still be employed for temperature control compliance audits. Avoid triggering multiple alerts for a late abnormal event.

Visual gaps are used to determine how long after an event it is perceived by a user; they indicate how aware a user is of that event.

$$G_k=\text{median}_{i\in\Omega_k}\left(t_i^{visible}-t_i^{event}\right)$$

In equation (4), $\Omega_k$represents *k*the set of events of type 1, and $t_i^{visible}$represents the time it takes for the event to be seen by the business user after state calculation, cache refresh, and push. Visual gaps are closer to user experience than backend latency because they include both state services and frontend push processes. For abnormal events, the platform should use this metric as a core SLA to ensure that schedulers have sufficient time to intervene.

4.4 State Machine Aggregation and Idempotent Consistency

The freight lifecycle can be modelled as a finite state machine, with the following main states: Created, Assigned, PickedUp, InTransit, AtNode, Transloaded, Delivered, and Closed; in addition, there are exception states such as Delayed, Damaged, CustomsHold, and TemperatureRisk. A State Machine is used to prevent invalid transitions. For example, a waybill that has not been picked up cannot directly enter *the* signed-for state; a closed waybill cannot be reopened via an old location event; and the resolution of an exception requires a corresponding exception creation event. The rules of the state machine can be configured to support different logic for various types of freight, routes and customer SLAs.

The level of event consistency can be expressed as the idempotent consistency rate.

$$C=1-\frac{D+M+V}{E}$$

In equation (5), *E*represents the total number of events received within the window, *D*represents the number of duplicate writes, *M*represents the number of lost or unresolvable events, and *V*represents the number of events that violate the state machine rules. *C*The closer is to 1, the better the pipeline can maintain state consistency. This indicator covers technical layer duplication, data layer missing data, and business layer violations, making it suitable for continuous quality monitoring after the platform goes live.

4.5 Anomaly Detection and Control Tower Linkage

The Business Value of the visualisation platform is to show the closed-loop nature of anomalies in business processes. Anomaly scores are based on ETA deviations, node waiting time, route deviations, sensor overruns, missing documents and customer SLA risk assessments. The anomaly score should not be based on a single rule; instead, all these factors should be taken into consideration, such as the sequence of events, value of the cargo, customer level, promised time and

past processing results. High-risk anomalies are automatically sent as control tower work orders, and the system will associate them with the responsible unit, suggested actions and notification templates. Low-risk anomalies are added to the observation queue and do not trigger alarms.

The purpose of the optimisation goal for the pipeline is as follows:

$$\min J=\alpha P95(L)+\beta G_{exception}+\gamma(1-C)+\delta Q$$

In Equation (6), $P95(L)$represents the 95th percentile of end-to-end latency, $G_{exception}$represents the abnormal event visualization gap, $C$is the idempotent consistency rate, $Q$is the consumption backlog or queue pressure per unit time, and α, β, γ, $\delta$are business weights. This objective function reflects engineering trade-offs: extremely low latency may lead to increased resource costs, while excessive pursuit of consistency may increase state convergence time. The platform needs to dynamically adjust the weights based on the risk of goods and customer SLAs.

4.6 Experimental Design and Statistical Results

A prototype load-testing environment was built to verify the effect of the new architecture, and during a 30-day freight cycle, the following event logs were collected: order status, vehicle location, node operations, sensor data, and abnormal work orders. The four integration modes tested were batch API, polling EDI, event mesh, and Kafka-Flink event-driven pipeline. The event distribution of each mode was input, and then end-to-end latency, gap visualisation, state consistency and anomaly response time were statistically analysed. The purpose of this experiment was not to reproduce all the features of a single enterprise, but rather to determine how well each of the various designs performed under the same load.

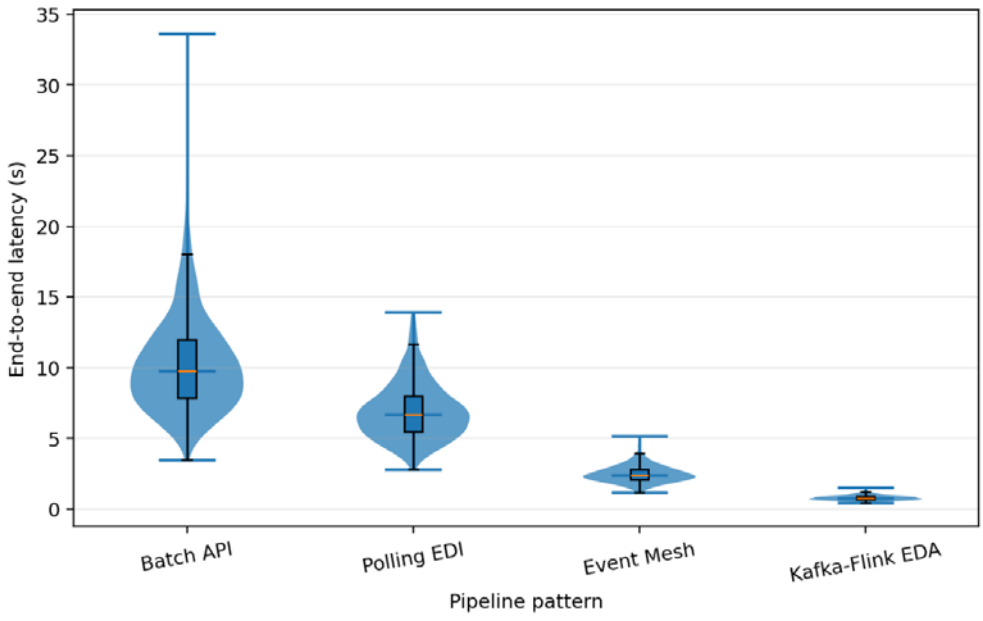

Figure 1. Event End-to-End Delay Distribution Under Different Pipeline Modes

Figure 1 is a violin plot that shows the distribution of latency for the four pipeline modes simultaneously; the median, spread and a long tail are also presented. The latency distribution of batch API and polling EDI is strongly right-skewed, and it is easy to be affected by the batch processing cycle in terms of synchronization and timed fetching. The event grid also has relatively low latency, but some long-tail cases remain. The Kafka-Flink event-driven pipeline has the most

concentrated distribution, with a median close to 0.8 seconds, and the P95 latency is also relatively low. Based on the above results, event streaming and state pre-computation can be used to reduce the end-to-end delay of freight visualization.

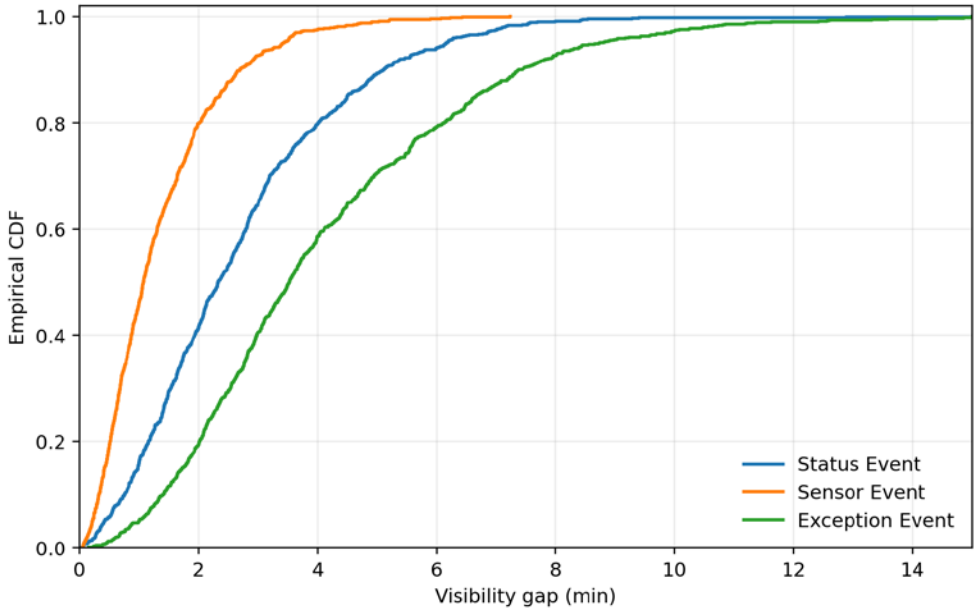

Figure 2. Visualisation of the Distribution of Gap Experience by Event Type

Figure 2 shows the lack of visualisation for status events, sensor events and anomaly events. Sensor events directly enter a high-frequency topic, are processed by a simple rule, and then identified by the business side in about 2 minutes. Status events are based on when the node system is submitted, and they are slightly shifted to the right. Anomaly events are based on risk scoring and work order creation, and have a longer tail. Therefore, it is not necessary to focus only on the average latency of the platform; we also need to establish a P95 SLA for anomaly events and use alarm idempotency, priority consumption and control tower linkage to reduce long-tail risks.

Table 3 Comparison of Key Indicators for Different Integration Modes Under Prototype Stress Testing

Architecture pattern	P50 latency (s)	P95 latency (s)	Visibility gap P95 (min)	Consistency rate (%)	Alert response P95 (min)
Batch API	9.84	16.72	18.40	97.60	21:30
Polling EDI	6.53	9.42	12.70	98.10	15.80
Event Mesh	2.41	4.18	7.20	98.90	9.60
Kafka-Flink EDA	0.79	1.31	5.26	99.30	6.54

Table 3 shows the performance comparison of the four architectures. The Kafka-Flink event-driven pipeline has higher latency (P50 and P95) than batch and polling modes but reduces the long tail. The consistency of the state has improved from 97.60% in the original batch API to 99.30%, and idempotent keys, state machine verification and watermarks have been used to reduce duplication, loss and illegal transfers. The P95 latency of the exception response is still higher than that for normal events; thus, risk scoring, manual confirmation, and external notification modules need to be improved further.

4.7 Project Implementation Strategy

First, set up event contracts. All events must pass schema validation before publishing; any changes to a field should be versioned, and producers are not permitted to modify enumeration values arbitrarily. Second, improve through partition keys. Order status events can be divided by `shipment_id`, vehicle location events by `asset_id`, and sensor events by `device_id`, and thus reduce cross-partition operations for status aggregation. Thirdly, add idempotent writes and event replay. In the writing of status services, use event_id, entity_id and version to form an idempotent key, and create new states by means of historical log replay. Fourth, build multi-level quantifiable indicators. At the same time, monitor the production rate, consumption backlog, end-to-end latency, late event ratio, state machine rejection rate and alarm closure time. Fifth, set tiered SLAs. Normal location updates can tolerate a delay of a few seconds or minutes, but cold chain anomalies and high-value goods deviations need to be promptly identified and alerted.

The above strategies are not realised by a single component but are achieved through the engineering organisation of the semantics for the freight visualisation business. Event-driven architectures that do not have state machine constraints will result in data noise; distributed pipelines lacking observable governance will fail to identify backlogs during peak periods; and real-time alerts without idempotent mechanisms will cause repeated notifications and operational fatigue. Only after integrating event contracts, stream processing, state aggregation, anomaly handling and operational metrics can a freight visualisation platform be considered a real-time operating system.

5. Conclusion

To address the problems of real-time performance, consistency and scalability in freight visualisation, a distributed data pipeline and an event-driven architecture model for multimodal transport are proposed. Research has found that the core of freight visualisation is state computation based on events, rather than merely synchronizing data and showing maps. Unify order, location, node, sensor, and anomaly data into an event stream, and use methods such as topic partitioning, streaming cleansing, watermarking, idempotent writing, state machine aggregation, and control tower linkage to reduce end-to-end latency, improve state consistency, and accelerate the time to business awareness of anomalies.

Prototype load test results show that the event-driven pipeline of Kafka-Flink outperforms batch API and polling EDI modes in P95 latency, visualisation gaps and consistency. When high-frequency bitstreams and sudden abnormal events occur simultaneously, an event-driven pipeline can keep the long-tail latency relatively low by asynchronously consuming and incrementally updating state. It can be seen that visualization platforms for new-era freight operation should move from being centres for interface integration to centres for event computation, and from post-event reporting to real-time awareness and proactive intervention.

The existing studies are not perfect. First, the prototype load testing logs in the experiment cannot represent the actual conditions at the enterprise level, and further long-term validation needs to be carried out in more real enterprise routes, port nodes, and cross-border business environments. Second, most anomaly scoring models are based on rules and statistical indicators; in the future, graph neural networks, Bayesian updates and causal inference can be incorporated to improve the detection of complex anomalies. Thirdly, cross-enterprise event sharing presents problems such as data sovereignty, privacy protection and the isolation of commercially sensitive information; further research can be conducted on federated event buses, privacy computing and verifiable audit mechanisms. Distributed data pipelines and event-driven architectures can generally provide all-encompassing support for managing freight visualization by integrating engineering support and operational intelligence, and are thus highly applicable in third-party logistics, smart ports, cross-

border trunk lines, and supply chain control towers.

References

[1] *Helo, P., & Thai, VV (2024). Logistics 4.0 – digital transformation with smart connected tracking and tracing devices. International Journal of Production Economics, 275, 109336.*

[2] *Fazi, S., & Fransoo, JC (2025). Freight Mobility as a Service: Open platforms for synchromodal transport. Transportation Research Part E: Logistics and Transportation Review, 204, 104368.*

[3] *Han, X. (2026). Research on Adaptive Optimization Methods for Multi-Objective Manufacturing Process Parameters in Complex Assembly Processes. Procedia Computer Science, 279, 366-374.*

[4] *Wang, N. (2026). Construction and Application of Interpretable Enterprise Performance Prediction Model Based on Multi-Source Operational Data.*

[5] *Ma, X. (2026). Distributed Fault Root Cause Localization and Self-Healing Strategy Generation Based on Causal Inference. Procedia Computer Science, 281, 753-760.*

[6] *Ma, W. (2026). Cloud Data Platform Governance Model under Infrastructure Automation.*

[7] *Wang, Z. (2026). Research on Data Analysis and Quality Prediction of Manufacturing Processes Based on Improved Deep Learning Algorithms. Procedia Computer Science, 281, 1328-1337.*

[8] *Chi, C. (2026). Research on Mortgage Asset Cash Flow Simulation and Risk Transmission Mechanisms across Structural Tranches Based on Loan-Level Data.*

[9] *Su, J. (2026). Design and Implementation of Android High Reliability Communication Module Based on Hierarchical Architecture.*

[10] *Sun, L. (2026). Resource Scheduling and Elastic Scaling Optimization of Distributed Advertising Systems in Cloud-Native Environments.*

[11] *Qian, X. (2026). Market Data–Driven Demand Forecasting and Inventory Coordination Mechanisms in Retail Supply Chains.*

[12] *Yang, Y. (2026). The Integration and Application of Ai Technology in Marketing Data Analysis and Business Decision-Making.*

[13] *Zhang, Y. (2026). Exploration of Enterprise Big Data Microservice Architecture Based on Domain-Driven Design (DDD). Procedia Computer Science, 282, 1994-2003.*

[14] *Liu, X. (2026). Construction of Consumer Data Privacy Protection System Based On Blockchain Technology. Procedia Computer Science, 282, 2082-2091.*

[15] *Liu, X. (2026). Research on the Application of Artificial Intelligence in Consumer Privacy Data Protection. Procedia Computer Science, 279, 956-965.*

[16] *Zhang, Y. (2026). A Framework for LLM-Based Semantic Configuration Understanding and Automated Change Generation in Microservice Orchestration.*

[17] *Chen, J. (2026). Research on Traffic Peak Shaving and Data Consistency Guarantee Mechanism of E-commerce Flash Sale System Under Event-driven Architecture. Engineering Advances, 6(2).*

[18] *Li, J. (2026). Analysis of Advertising Creativity Generation and User Response Mode Based on AIGC.*

[19] *Zhang, J. (2026). Research on Value Stratification and Precision Marketing of Retail Customers in Banks Based on Clustering Algorithm.*

[20] *Zhang, Z. (2026). Research on Performance Monitoring and Data-driven Optimization Mechanisms for AI Systems Oriented Toward Decision Support. Advances in Computer and Communication, 7(2).*

[21] *Ma, X. (2026). Research on the Design and Application of Observability Architectures for High-reliability Distributed Systems in Cloud-native Environments. Engineering Advances, 6(2).*
[22] *Wang, Z. (2026). Earnings Quality of Resource Enterprises Under Commodity Price Volatility Transmission.*
[23] *Ma, W. (2026). Reliable Data Infrastructure Supported by Automated Data Pipelines. Engineering Advances, 6(2).*
[24] *Jin, C. (2026). Research on the Optimization Strategy of Retail Enterprise Product Portfolio Based on Association Rule Analysis. Advances in Computer and Communication, 7(2).*
[25] *Wang, Z. (2026). Data-driven Pathways for Optimizing Supply Chain Operational Efficiency in Physical Industries. Advances in Computer and Communication, 7(2).*
[26] *Wang, N. (2026). Research on Digital Marketing Conversion Improvement Strategies and Predictive Analysis Based on User Behavior Segmentation. Journal of Humanities, Arts and Social Science, 10(7).*
[27] *Ma, X. (2026). Research on Elastic Scaling and Resource Scheduling Strategies for Distributed Systems Facing Traffic Fluctuations.*
[28] *Wang, Z. (2026). Valuation Enhancement Pathways for Resource Stocks Driven by the Growth of Energy Storage Demand. Engineering Advances, 6(3).*
[29] *Su, J. (2026). Research on Machine Learning-based Performance Prediction and High-reliability Software Evolution Mechanisms for Android Communication Systems.*
[30] *Zhang, Y. (2026). Autonomous Software Release Management and Cross-service Consistency Control Mechanisms Based on Large Language Models. Advances in Computer and Communication, 7(3).*