WORLD COMET RESEARCH GROUP

International Journal of Data Engineering and Intelligent Computing
Vol. 6, No. 1, 2023
Doi: 10.67228/30715717/IJDEIC-2023PI9T2L
PP. 01-18

Original Article

Real-Time Analytics and Cloud-Native Data Engineering for Smart Manufacturing Systems

Dhanaraj Sathiri

Independent Researcher, USA.

Received: 12-04-2022 Revised: 12-26-2022 Accepted: 01-03-2023 Published: 01-05-2023

ABSTRACT

Although cloud computing is not yet widely embraced by the industrial domain, its growing popularity in Information Technology suggests it will soon be adopted in this area as well. Smart manufacturing advocates see potential in the cloud-native ecosystem due to its capability for rapid development and cost savings. At the same time, the Internet of Things (IoT) and Big Data are fundamental components of smart manufacturing. A cloud-native environment significantly lowers the barrier for small- and medium-sized enterprises that strive to take the first step toward integrating these technologies. Yet cloud-native technology brings its own challenges, and suitable architecture design patterns must be adopted to ensure scaling and optimization during and after development. Based on an extensive body of knowledge and well-proven design patterns, these aspects have been incorporated into a cloud-native, data-engineering-oriented end-to-end solution design that supports real-time analytics in a manufacturing context. The solution design ultimately enables data from IoT devices to flow all the way through to the delivery of insights with decision-making impact. It covers the entire process—from data ingestion and storage to data-engineering pipelines and enabling cloud-native tooling—and satisfies six aspects important for cloud-native design: scalability, elasticity, multi-tenancy, support for stateless services, service mesh integration, and observability. The results, streamlined, synthesized, and presented in an integrated manner, provide a cloud-native solution-architecture perspective aligned with the real-time analytics demand of IoT-enabled manufacturing systems. They also enable or facilitate further, more specialized design decisions in the context of analytics and data engineering and serve as a knowledge repository for related explorations.

KEYWORDS

Cloud-Native Manufacturing, Smart Manufacturing Systems, Industrial IoT Analytics, Real-Time Data Pipelines, Scalable Cloud Architecture, Elastic Compute Resources, Multi-Tenancy Support, Stateless Service Design, Service Mesh Integration, Observability Frameworks, Data Ingestion Pipelines, Streaming Analytics, Edge To Cloud Integration, Manufacturing Data Engineering, Decision Support Analytics, Cost-Efficient Deployment, Small And Medium Enterprise Enablement, Big Data Infrastructure, Cloud-Native Design Patterns, Industrial Analytics Governance.

ISSN: 3071-5717
PP: 01-18.

1. INTRODUCTION

Despite surging interest in edge-computing architectures for the Internet of Things (IoT), most industrial IoT deployments continue to rely on cloud solutions. While cloud-centric platforms support data acquisition, storage, and analytics, provisioning data engineering pipelines remains complex and cumbersome. This presents a major limitation to real-time monitoring, simulation, and optimization of interconnected manufacturing systems. Comprehensive cloud-native data engineering frameworks that standardize tools and procedures for real-time generation of actionable insights from industrial telemetry data are still largely lacking.

The outcome of the study reported here constitutes an evidence-based synthesis that details the capabilities of cloud-native data engineering solutions for real-time analytics in industrial IoT applications. The proposed framework, anchored in central tenets of cloud-native design and data-processing paradigms, guides the conception and implementation of data pipelines that prepare telemetry data for real-time use cases spanning predictive maintenance, process optimization, and intelligent prioritization of production schedules. The work also highlights a best-practice approach for the development of data models tailored to time-series data.

1.1. Overview of the Study and Its Objectives

A Cloud-Native data engineering framework for real-time analytics in IoT-enabled manufacturing systems is proposed and evaluated. The objective is to establish reusable design patterns and components that enable rapid construction of reliable, cost-effective analytics pipelines.

Smart automation of manufacturing systems relies on real-time analytics of IoT telemetry. Cloud-Native architectures provide scalability, elasticity, multi-tenancy, stateless services, and a service mesh. However, data engineering for real-time analytics remains immature. Streaming data ingestion, storage, and processing are critical to timely insights but are inadequately explored. As a result, solutions are often poorly engineered, with limited reuse and long delivery cycles.

The work synthesizes current knowledge to create a comprehensive framework for Cloud-Native data engineering in support of real-time analytics. The resulting patterns and components support the entire data engineering life cycle, addressing concurrencies from telemetry telemetry and beyond. Special attention is paid to analytics pipelines that feed into smart automation use cases. Empirical support is drawn from the industrial deployment of a real-time anomaly detection solution based on Internet-scale telemetry data — as indicated by the range of detected cloud-native principles.

Cloud-Native data engineering for real-time analytics in IoT-enabled manufacturing systems contributes the following. First, it presents a design framework that encompasses the complete data engineering life cycle, integrating the guiding principles of Cloud-Native architecture at every level. Second, it identifies patterns specifically focused on real-time analytics, which often throttle the delivery of data-driven insights. Finally, it outlines a set of tools and patterns that support and govern the design of data engineering pipelines, enabling timely information to reach the right stakeholder at the right time.

ISSN: 3071-5717
PP: 01-18.

Fig 1 - Cloud-Native Data Engineering for Industrial IoT: A Framework for Real-Time Analytics and Smart Automation Patterns

2. THEORETICAL FOUNDATIONS

Cloud-native data engineering for real-time analytics in IoT-enabled manufacturing systems; present objective, evidence-based synthesis with formal structure.

The core concepts, relevant theories, and context required to justify the design choices and evaluation of a Cloud-Native Data Engineering Framework for Real-Time Analytics in Smart Automation of IoT-Connected Manufacturing Systems are established in this section. The principles underlying Cloud-Native architectures are first examined, supported by a rationale for their explicit incorporation in the proposed framework. The discussion then turns to the main characteristics of modern real-time data processing paradigms, differentiating them from traditional batch processing and highlighting the factors influencing the selection of an appropriate approach for a given use case.

Cloud-Native architecture is inspired by eight principles, namely: Scalability, Elasticity, Multi-Tenancy, Stateless Services, Service Mesh, Observability, and Fault Tolerance. A Cloud-Native environment consists of many loosely coupled components that communicate via well-defined APIs or protocols. The elasticity of Cloud-Native systems and services improves resource efficiency and the associated costs, enabling organizations that have not yet fully adopted the Cloud to benefit from Cloud-Native solutions for their systems. The Stateless Service principle enables the use of Service Mesh and Observability metrics and opens the road to Horizontal Scaling of services via the use of Load Balancers. The introduction of a Service Mesh decouples inter-service communication from the application logic, while the addition of the Sidecar pattern in deployed services enables the collection of rich metrics from the sidecars without polluting the service code.

2.1. Cloud-Native Architecture Principles

The following architecture design follows cloud-native principles, favoring scalability over performance. The increase in Internet-of-Things (IoT) adoption across various application domains enables a high, continuous, and real-time data stream generated by thousands or millions of connected devices. Virtualized or containerized solutions can effectively host scalable system components such as database services or servers. A cloud vendor can dynamically replicate and assign compute

ISSN: 3071-5717
PP: 01-18.

resources to tenant components such as Data Ingestion, Event Processing, and Data Engineering Pipelines.

Elastic scalability supports seamless and automatic scaling of workloads, enabling shared computing costs for the same architecture. The multiple-tenancy model supports efficient scaling across different data engineering workflows and massive data volumes. The underlying cloud infrastructure monitors and manages multi-tenancy, facilitating and hiding range changes. Services are stateless or designed with a Stateless Service Mesh, with session information managed by distributed, fault-tolerant, multi-region database services. Stateless services can scale, allocate, and release resources independently, enhancing service availability and efficiency.

Full observability is supported from the application and infrastructure layers to data-layer metrics. Full observability simplifies architecture change and incident recovery. Services include full monitoring and tracing coverage. Full observability and monitoring allows simple matching of data-latency requirements with stream-processing techniques. Finally, fault tolerance ensures that the correct business process state is reached. Data engineering pipelines guarantee business process–definition correctness in the presence of component failures.

2.2. Real-Time Data Processing Paradigms

Three paradigms support real-time data processing: stream processing, event-driven architecture, and complex-event processing. Stream processing serves continuous data flows with little or no latency. Ingested data is treated as a series of time-ordered statements, each of which can be evaluated independently. Event-driven architecture positions detected interesting events or patterns as the primary stimuli for other components of the applications, which remain idle until the arrival of an event. Such applications often respond faster than data delivered to and stored in a data lake or data warehouse but attend only to specific, significant events. Complex-event processing extends the model further, enabling triggering of actions by patterns involving the occurrence of a time-sequenced set of events rather than incidental, single occurrences.

The response times demanded by many applications often generate a trade-off between latency and throughput, performance, and guarantees. Another essential aspect of system performance is consistency, the degree to which visible data does not depend on the time when an operation is invoked, and how it relates to latency. Traditionally, consistency is ensured through transactions, but transactions can inhibit scalability and availability, creating a trade-off for deployment. Solutions such as linearizable reads and eventual consistency are often sufficient but cannot prevent all anomalies.

3. SYSTEM ARCHITECTURE

The system architecture comprises an end-to-end design capable of ingesting, storing, and processing real-time telemetry data from IoT-connected manufacturing systems. It encompasses workflows for data ingestion, processing and storage, and support for data engineering pipelines; data orchestration and provenance; condition-based analytics; and data science algorithm deployment.

ISSN: 3071-5717
PP: 01-18.

These capabilities are instantiated using an assortment of open-source and cloud-native solutions, whose specific implementations are considered when discussing the various components.

System architecture capable of ingesting, storing and processing real-time telemetry data from IoT-connected manufacturing systems. End-to-end design incorporates components for data ingestion, storage and processing; implementation of data engineering pipelines; data orchestration and provenance support; condition-based analytics; and deployment of data science algorithms. Capabilities are instantiated using open-source and cloud-native solutions, with specific implementations considered at the component level.

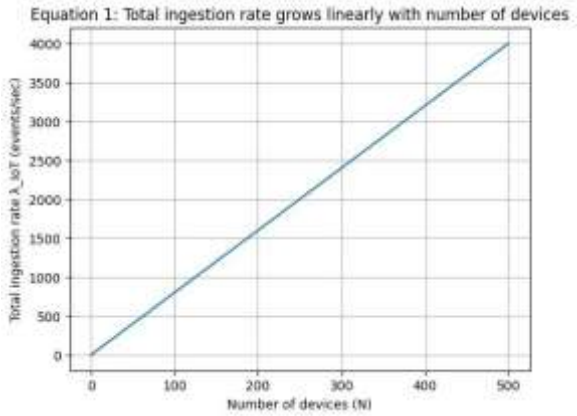

Fig 2 - Total Ingestion Rate as a Function of the Number of Devices

Equation 1: IoT Data Stream Ingestion Rate

$$\lambda_{IoT} = \sum_{i=1}^{N} r_i$$

("Equation 1: IoT Data Stream Ingestion Rate".)

Step-by-step derivation

1. **Define per-device rate**
   - Let device i emit telemetry at rate:

$$r_i = \frac{events\ from\ device\ i}{second}$$

Units: **events/s**.

2. **Total stream is the union of all device streams**
   - If devices are independent sources, the total incoming stream rate equals the **sum of their rates**.
3. **Sum across all N devices**

$$\lambda_{IoT} = r_1 + r_2 + \cdots + r_N = \sum_{i=1}^{N} r_i$$

3.1. Data Ingestion and Streaming

Data connectors and streaming capabilities are key enablers in a cloud-native data engineering framework. A connector links an external data source or sink to the data ecosystem, handling low-level operational aspects such as authentication, protocol adherence, schema management, and

ISSN: 3071-5717
PP: 01-18.

backpressure handling, typically via a publish–subscribe model. The streaming framework provides the data pipeline engine focused on data transit, enabling logical groups of streaming jobs and dynamic scaling, often with optimum resource triggering policies.

Real-time data ingestion should target a maximum latency between source change and data record availability across the ingestion system. In most situations, a ten-second maximum is acceptable; for some video/audio connections, a sub-second ceiling is needed. This latency must consider backpressure handling and elements outside the streaming framework. Low-latency data ingestion usually requires socket, message queue, or streaming protocol usage; file-based data sources are seldom suitable. A connector specialized in Live555 protocols can directly pull RTP/RTCP packets from IP cameras/devices. NDI (Network Device Interface) is another emerging protocol optimized for low-latency video transmission over the network. Concerning ingestion latency, sockets and messages queues are the quickest methods. However, a Pull-based protocol like HTTP2 streaming can also be low latency if correctly optimized.

Table 1: Device-Level Image Event Contribution Analysis

Device i	r_i (events/sec)	Contribution to λ_{IoT}
1	5	5
2	12	12
3	3	3
4	20	20
5	10	10
Total		$\lambda_{IoT} = 50$ events/sec

3.2. Data Storage and Lakehouse Concepts

Modern manufacturing systems produce large volumes of data that must be stored in structured and summarized forms suited for end-user queries and analytics. To provide flexible and performant data storage, the data engineering component must ingest information into a lakehouse architecture, where data is consumed downstream based on immediate operational needs. A cloud-native data lake fulfills services, capacity, and multi-tenancy scalability requirements. End-user insights primarily originate from validation of machine learning models and other analytics on training, iteration, evaluation, and production phases, while metadata management boosts knowledge discovery. At rest, data are structured in tar files and stored in an object-storage service that natively integrates with processing solutions deployed in the lakehouse layer, in turn built on an object-storage engine and optimizing workloads through table formats and partitioning. A unified catalog of data sources, formats, and schemas supports end-user query and processing workloads and enables their support for data retention, governance, and security operations.

For cloud-native data engineering, the terms lakehouse and data lakehouse identify a shared operational layer, primarily for structured data consumption, that supports large-scale batch processing alongside low-latency analytics. This main cloud-native data product is structured as a fusion of the two main reference architectures: a data lake for raw information and a data warehouse

ISSN: 3071-5717
PP: 01-18.

for refined end-user summaries. A cloud-native metadata manager with a unified catalog for all incoming data stores, sources, formats, credentials, and schemas enforces authorization and governance controls. Data arrive in the lakehouse from different sources that address varying service-level objectives. The data management strategy formalizes the placement of items according to their velocity and timeliness, with the proportion of structured data within the general data lake increasing as latency requirements relax, while Durability and retention levels follow the corresponding SLAs.

4. DATA ENGINEERING PIPELINES

Cloud-native data engineering pipelines for IoT-connected manufacturing systems deliver key real-time insights with minimal latency. The design process draws on theoretical principles and adapts pipeline templates to a specific business domain governed by regulatory compliance, operational constraints, and safety considerations.

End-to-end data pipelines integrate ingestion, storage, processing, and serving. Telemetry data ingested from various sources pass through a five-step flow. Data modeling defines source schemas that take time-series characteristics into account, while transformation and enrichment patterns specify operations such as cleansing, normalization, deduplication, feature extraction, and external data enrichment. Queries support monitoring, alarm triggering, reporting, and predictive maintenance across multiple process steps and batches—alongside anomaly detection for quality assurance—and deliver results to control loops and workflow components for real-time scheduling and process optimization.

Heterogeneity, rapid stream evolution, and regulatory compliance introduce extra complexity at the ingestion stage of telemetry data. The governing schema registry serves as a single source of truth; schema validation ensures data quality, while schema evolution supports processing. With the upper ingestion latency constraint set to 30 seconds, the pipeline accommodates backpressure from later stages without dropping any data.

Fig 3 - Latency-Optimized Cloud-Native Pipelines for IIoT: A Governance-First Framework for Real-Time Telemetry Orchestration and Schema Evolution

4.1. Data Modeling for IoT Telemetry

A common denominator of the data models created to satisfy particular use cases, often strongly influenced by the associated optimization tasks, is the focus on different forms of telemetry data. This family of data types is typically produced at high frequency by many—sometimes even

ISSN: 3071-5717
PP: 01-18.

thousands—of devices. The organization of this type of information is a key factor for achieving low latency in analytics pipelines. The main requirements for defining the schemas of incoming data streams are stated first, followed by a discussion of time-series data and its impact on selection of the right storage technology. These topics are then complemented with a look at dimensional models and compression strategies. The associated ad-hoc design and maintenance issues are addressed along with treatments of deduplication and external data enrichment.

Data engineering for IoT telemetry involves a specific problem field: the design of schemas for the production of data, models for the events being produced, and patterns for deduplication and external-enrichment features. A set of very high-frequency signals is typically received from the connected devices. This incoming information is similar to an over-dimensioned set of telemetry signals. At this scale, and without proper governance, it can degenerate into what is often termed IoT junk data. The business requirements for monitoring and control are not fulfilled, and it becomes difficult to maintain the data engineering pipelines. Therefore, the approach to modeling and controlling the telemetry signals deserves attention.

4.2. Transformation and Enrichment Patterns

When raw incoming data can involve considerable redundancy, transformation must manage discounting of duplicates and corrections to time-sequenced-values used for applications such as anomaly detection. Cleansing is also necessary when confidential or inaccurate data may obscure or bias machine-learning models. Three common enrichment patterns offer data producers a means of improving quality and interpretability for receivers. The first employs an external data source to derive a information-rich additional column or to extract a subcomponent from a nested data format. The second provides a location-specific translation of an ID column, such as mapping of a factory site ID to its location, and the third extends existing telemetry with normally-closed state indicators to expose failure or maintenance events that may no longer be electrically monitored.

Background from external APIs is especially useful for anomaly models that distinguish normal operation from the unusual, as any naturally-scattering variables contribute significantly to model accuracy. For example, weather conditions normally affect the activity of IoT-connected machines used outside, and therefore weather-related features must be included in the training data. Enriching a telemetry message with temperature or humidity from a nearby weather station can help fulfillment centers that ship packages warn couriers to pay special attention to their loads. However, using such external APIs in real-time is risky. It adds a time delay to the telemetry processing pipeline, increases the completion time for all data records while only being used for a small fraction during the model training phase, and introduces yet another mechanism that may fail with little or no notice.

Serious model failures occur when an unexpected longitude or latitude is detected without a valid nearby site. Location mapping can therefore be performed at pipeline level, translating location IDs in telemetry records into human-readable names or city identifiers with associated weather station IDs that are closer than a predefined threshold. In addition to reference data that make it easier for a user to explore the operation history of their machines, machine-learning models for any of the

ISSN: 3071-5717
PP: 01-18.

involved factories are simpler to construct since the data become geo-located and wind compass direction available. Such mapping can also be extended to other types of location IDs outside the Factory Automation domain, such as service areas for couriers or delivery trucks.

Table 2: Pipeline Components and Corresponding Latency Breakdown

Component	Symbol	Latency (ms)
Processing time	t_{proc}	120
Network time	t_{net}	45
Storage/commit time	t_{store}	60
Total		$L_{pipe} = 225$ ms

5. CLOUD-NATIVE TOOLING AND PLATFORM COMPONENTS

Mapping the required tooling and component set to the roles and responsibilities of data engineering reveals the areas of specialization needed in the platform supporting implementation. Ingestion, storage, data modeling, transformation, and streaming–applied monitoring and governance capabilities are considered in turn. Each data-engagement phase is discussed in context of the class of requirements, the range of mainstream technology choices available, and any distinct direction within the solutioning approach.

For the core streaming capabilities, the design employs a hybrid combination of public and private Cloud. For continuous ingestion from fast-paced systems a suitable event streaming framework is needed, with choice of message channel ideally being orthogonal to the choice of distributed compute framework for the subsequent processing and analytics. Aprise of the immediate requirements allows consideration of Kafka and Pulsar. Both have well-honed capabilities around delivering efficiently managing at scale the high ingress and, often, egress-throughput requirements fundamental to most telemetry and telemetry-log use cases, supporting SSD-terminated topics and disk-persistent partition segments accommodating background deletion and compression activities. Pulsar has a more holistic architecture, with support for multi-cloud, multi-datacenter, and multi-tenancy, and native backpressure.

Deployment timing will dictate ingestion latency requirements, as device-to-cloud telemetry is likely to be observed on stretches of tens of minutes with a single value produced per device. Ingestion latency close to zero will therefore greatly reduce latency at the next stage of processing. Lighter-weight protocols and file-formats better suited to rapid ingestion than JSON over HTTP may be employed: Kinesis- or MQTT-like protocols of which protocol buses are available as connectors to Kafka, Pulsar, or native Azure Event Hubs, and Protocol Buffers, or FlatBuffers once-ingested with the respective stream-compute-based telemetry log processing activities for windowed storage and buffering.

ISSN: 3071-5717
PP: 01-18.

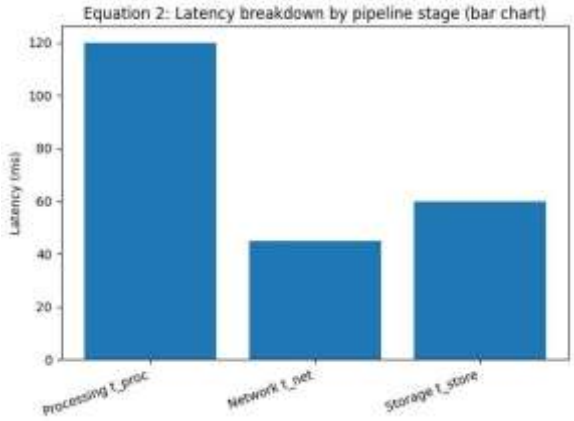

Fig 4 - Latency Distribution Across Different Pipeline Stages

Equation 2: Cloud-Native Data Pipeline Latency

$$L_{pipe} = t_{proc} + t_{net} + t_{store}$$

("Equation 2: Cloud-Native Data Pipeline Latency".)

Step-by-step derivation

4. **Define end-to-end latency**
   - Pipeline latency L_{pipe} = time from when a telemetry event is produced/arrives to when it is stored/available downstream.
5. **Decompose the pipeline into sequential stages**:
   - **Processing** (decode, validate, transform, windowing, feature extraction, etc.)
   - **Network transit** (device→broker, broker→compute, compute→storage, cross-AZ, etc.)
   - **Storage write** (commit to lakehouse/object store/DB)
6. **For serial stages, total time is additive**

If the record must pass through these stages *in order*, then:

$$L_{pipe} = (time\ in\ processing) + (time\ in\ network) + (time\ in\ storage\ commit)$$

So:

$$L_{pipe} = t_{proc} + t_{net} + t_{store}$$

5.1. Streaming Frameworks and Compute Resources

Institutions of all sizes are increasingly adopting Data Science to deliver value from the vast amount of data generated by their operations, customers, and products. Recent advances in computing technologies allow companies to obtain results almost in real-time, enabling them to reduce their response time to internal and external events, anticipate possible future scenarios, and make the best possible decisions. However, conventional Data Engineering processes, which are based on the traditional Waterfall Model, are often inefficient in a context where data are constantly generated and must therefore often be processed and analyzed in real-time. New approaches and processes are required that allow Data Science teams to scale and replicate their work, governing it and making it accessible to the lowest levels of the organization while minimizing performance and data quality issues. Streaming Services enable near real-time data processing through either Continuous Queries against data streams or Mini-Batches based on dynamic Windowing of Data Streams. Continuous

ISSN: 3071-5717
PP: 01-18.

Queries significantly simplify and accelerate development and also guarantee lower latency. Streaming Frameworks such as Apache Kafka and Apache Pulsar combined with Sharding, Partitioning, and/or Processing Capabilities of platforms such as Apache Flink or Apache Spark can efficiently support a wide range of Data Science and Data Engineering workloads.

Real-time Data Science even for Non-Data Scientists requires access in real-time to operations' data, clear indication of the Level Of Service (LOS) expected when using the Model, indications of the business value at risk when actual Service Level Objectives are not satisfied, and clear Decision-Action processes when acting on the Event. The cost of real-time Data Science Modeling and Analysis should be considered against the business value generated and should target highly valued Business Processes, Capabilities, and Opportunities (BPCO). Service Level Objectives (SLO), Data Governance, Data Quality, and Reputation factors all play important roles in real-time Data Science Delivery.

5.2. Storage Solutions and Data Governance

Data storage involves the organized retention of data for immediate and future retrieval and processing. Storage selection hinges on access frequency, latency expectations, scalability, metadata richness, and generative capabilities. Low-latency storage can involve conventional databases (often NoSQL for Time-Series DBs) and (distributed) in-memory databases and caches. During the streaming process, data usually reaches at least one durable backing store. Object storage's innate scalability, low cost, and high durability make it the first choice for any seldom-querying backup or data with an expected low query frequency. Replicated stream storage enables re-accessing high-fidelity audio/video streams.

Data modeling frequently emphasizes the creation of data marts/tables in a data warehouse, where querying speed takes precedence over storage consumption. Industry surveys routinely point to the consistency disparity in manual ETL process areas, establishing the necessity for a central query-accessible ground truth and metadata management compared to data exploration within lake environments. These considerations neatly integrate into a lakehouse architecture. By employing the lakehouse construct for the primary streaming data storage solution, it becomes possible to satisfy both customer requirements and standardization. The metadata inventory of the underlying lakehouse can ensure compliance, support data operations/generative actions, and incorporate a full data lineage. Its usage as the primary role of the lakehouse layer adds another advantage: storage costs can be kept at a minimum by strategically employing hot and cold storage.

Data governance also emerges as an instrumental component. A proper governance framework should meet regulatory demands, offer a unified view of the data landscape, grant organizations reassurance over the usage of public-cloud storage services, and provide an enterprise-ready access-control model. To fulfill such requirements, a combination of a data-masking engine, a metadata-cataloging solution, and a data-lineage product is necessary.

ISSN: 3071-5717
PP: 01-18.

6. REAL-TIME ANALYTICS FOR SMART AUTOMATION

Smart automation of IoT-connected manufacturing systems generates large amounts of streamed data. Increasing accuracy, responsiveness, and efficiency of production processes will further reduce resource consumption and environmental footprint. Consequently, data engineering plays a critical role in augmenting efficient data processing and governance, transforming telemetry streams into actionable insights. A cloud-native real-time analytics architecture, based on core data engineering patterns, accelerates resilient rating and failure detection of machines and products, as well as zero-latency scheduling of related manufacturing processes. Consequently, the labor and material costs of unplanned stoppages, excessive stock of finished goods, and missed orders are mitigated.

Continuous training of anomaly detection, predictive maintenance, scheduling, and process optimization models not only improves model accuracy but also guarantees scheduling of manufacturing activities as required. For anomaly detection, multiple statistical models process different features of machine telemetry, exposing distinct airways and faults, including thermal copper overheating. Predictive maintenance detects potential failure of an asset by estimating remaining useful life of the machine. A scheduling algorithm prioritizes the fabrication of high-risk products to lower the occurrence of defects. In combination with a real-time scheduling engine, it predicts next actions of assets to minimize the makespan of intertwined operations. A dedicated feedback loop adjusts the scheduling function according to the accuracy of responses of different critical assets.

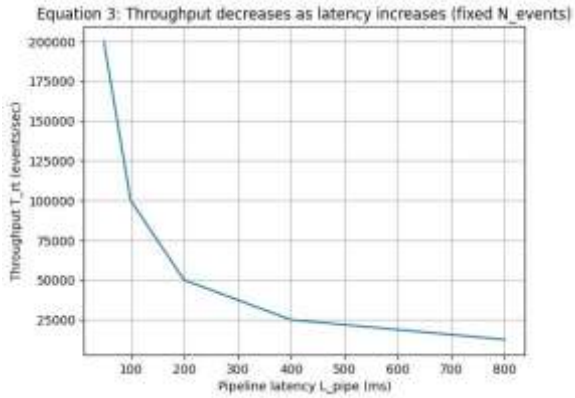

Fig 5 - Throughput Variation with Increasing Pipeline Latency

Equation 3: Real-Time Analytics Throughput

$$T_{rt} = \frac{N_{events}}{L_{pipe}}$$

("Equation 3: Real-Time Analytics Throughput".)

Step-by-step derivation

7. **Start with the definition of throughput**
   - Throughput is "how many events per unit time":

$$T_{rt} = \frac{\#\ events\ processed}{time\ taken}$$

8. **Choose a measurement window / batch / interval**

ISSN: 3071-5717
PP: 01-18.

- Let N_{events} be the number of events that traverse the pipeline in some time interval.

9. **Use end-to-end pipeline time**
   - The time used here is L_{pipe} (pipeline latency).
10. **Substitute into the throughput definition**

$$T_{rt} = \frac{N_{events}}{L_{pipe}}$$

Units:

- N_{events}: events
- L_{pipe}: seconds
- T_{rt}: events/second

Table 3: Throughput Variation with Increasing Pipeline Latency

L_{pipe} (ms)	L_{pipe} (s)	N_{events}	T_{rt} (events/sec)
50	0.05	10,000	200,000
100	0.10	10,000	100,000
200	0.20	10,000	50,000
400	0.40	10,000	25,000
800	0.80	10,000	12,500

6.1. Anomaly Detection and Predictive Maintenance

The data engineering pipelines enable a diverse set of real-time analytics scenarios of increasing business value. At the base level, low-value analytics such as anomaly detection are among the first applications toward smart automation. Concepts have been extensively studied in the literature but are not always applicable in practical scenarios due to deployment costs, limited training data, and the challenges of a working strong supervision scheme. The predictive maintenance of factory assets is a common high-value application. It follows from monitoring the telemetry of the physical asset and detects patterns in either the temporal or spectral domains that signal the need for replacement, repair, or maintenance. Concepts span time-series, statistical methods, supervised learning, and deep learning. Terabytes of data can be generated over a product lifecycle, substantially aiding the model training process. Labeling the data is a challenging task; a supervisor is normally needed to mark the state of the product over its lifecycle, which may not always be feasible or might be expensive. The temporal distribution of this label is often highly imbalanced, leading to a predictive maintenance model that many not succeed in deploying properly.

The internet of things (IoT) enables data collection on working assets, but labelled datasets are still rare. Signal processing, time-series and data-mining techniques with little or no model supervision can detect conditions necessitating maintenance. New tools can detect and remove normal states of the asset, thus transforming the telemetry data to an anomaly-detection setup. It is essential to define features and metrics for performance evaluation, establish thresholds, implement response mechanisms, and consider distribution drift. In sensitive critical manufacturing operations, responses to alarms are fulfilled with priority. Resources are arranged to analyse the cause of the warning or detected anomaly, when possible, and a work order is created. With predictive maintenance, well-

ISSN: 3071-5717
PP: 01-18.

defined features related to the alert condition drive the training and testing process. Supporting such service, training data may not be imbalanced for the events of interest.

6.2. Real-Time Scheduling and Process Optimization

Supporting smart automation demands not only detecting and acting on unexpected events but also making scheduling and optimization decisions online. Typical tasks in manufacturing, logistics, and process control can be described with constraints (e.g., resources, order priorities, deadlines) and objective functions (e.g., production cost, energy consumption, service level). In theory, those problems can be solved for a future time window, but in practice, time is limited. Therefore, even low-quality solutions can help respond to the current state. This role can be fulfilled by \(\text{control}^4 \).

Three clouds interact under a data-sharing agreement—operations (customer), monitoring and analysis (vendor), and optimization (third party). The product-producing process generates a flow of events that violates defined constraints (such as deadlines). The monitoring cloud receives the events, evaluates monitoring models, and notifies the optimization cloud. The violated constraints, control parameters, and state information are then used as input to generate a nonproductive solution of the optimization problem. This solution is finally sent to the control loop, resulting in a controlled process. Solutions are proposed only for the part of the process represented by the violated constraints and with control guarantees.

Quality assurance is ensured by testing the solution generation in a simulation environment where the problem is solved with ample time and the models are reversible. Possible constraints, objectives, metrics, thresholds, and actors are predefined within the solution-execution process.

7. CONCLUSION

Cloud-native data engineering provides a blueprint for scalable systems enabling real-time analytics that drive smart automation in IoT-enabled manufacturing. Synthesis of design patterns, architectural concepts, and tooling recommendations demonstrates that near-real-time requirements can be tractably addressed, reducing decision response times. Patterns for rapid ingestion and pre-processing of time-series telemetry support low-latency pipelines, while batching of less time-critical data and observability using change-data-capture techniques optimize processing efficiencies. Future work will focus on exploring time- and resource-efficient designs for anomaly detection and predictive maintenance.

Enablement of smart automation requires use-case-specific models, as well as adherence to response-time requirements. Scheduling delegates analytical decisions to real-time data pipelines and low-latency models, thus enforcing timeliness in the scheduling decision itself. Constraint-aware scheduling exploits look-ahead techniques to enable horizon horizons commensurate with response times in the control loops of sequence-driven processes, whereas control-feedback scheduling integrates process considerations into manufacturing control systems for improved order fulfilment. Scalability, resilience, and elasticity across the supporting analytical framework allow for consideration of response and sensor deployment costs in solution specification.

ISSN: 3071-5717
PP: 01-18.

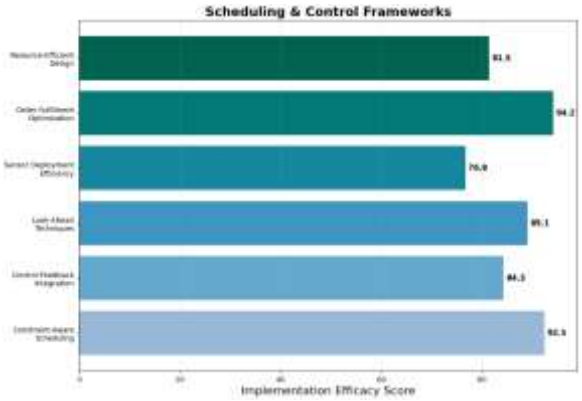

Fig 6 - Scheduling & Control Frameworks

7.1. Final Thoughts and Future Directions

Cloud-native data engineering frameworks that enable real-time analytics of IoT telemetry data can empower smart automation of manufacturing systems—especially in optimizing resource efficiency and system availability. Yet successful integration into industry requires further investigation, across three interrelated dimensions.

First, whether existing open-source solutions deliver the expected value proposition during real-world evaluation is an open question. Cloud-native tools can achieve improved scaling and elasticity, at a competitive cost, and when integrated with associated workloads, their distinct advantages may flow naturally. Nevertheless, evidence remains anecdotal; indeed, one of the greatest challenges currently lies in making the associated claims visible and provable in greenfield applications, especially in the industrial IoT domain.

Second, as cloud-native technology becomes increasingly mainstream, service providers will continue to extrapolate trends, investing in specializations that promote smart automation and respond to the needs of global manufacturing supply chains. These investments should provide increasing access to the technology area, including familiar services supported by developers who can asymmetrically source less common skills.

Finally, while industry 4.0 and Industry 5.0 principles support growing adoption of real-time data analytics, cloud-native product vendors have yet to build these capabilities into their SaaS offerings. Integrating principles of smart manufacturing and real-time data processing within established platform-as-a-service solutions—from hyper-scalers and other cloud vendors alike—has the potential to accelerate and de-risk deployments.

REFERENCES

[1] Mangalampalli, B. M. (2022). Automated Invoice Validation Systems Using Advanced SQL Analytics in Healthcare Insurance. Front Health Inform, 11.

[2] Davenport, T., & Kalakota, R. (2019). The potential for artificial intelligence in healthcare. Future Healthcare Journal, 6(2), 94–98.

[3] Puli, V. O. R., & Maguluri, K. K. (2022). Deep learning applications in materials management for pharmaceutical supply chains. Migration Letters, 19(6), 1144-1158.

[4] Inala, R. (2022). Cross-Domain MDM Integration Using AI-Driven Data Governance: A Case Study In Financial Technology Architecture. Migration Letters, 19(2), 280-304.
[5] Vankayalapati, R. K., Pandugula, C., Ganti, V. K. A. T., & Mishra, G. (2022). AI-powered self-healing cloud infrastructures: A paradigm for autonomous fault recovery. Migration Letters, 19(6), 1173-1187.
[6] Srinivas Kalisetty (2020). Intelligent Supply Chain Ecosystems: Cloud-Native Architectures and Big Data Integration in Retail and Manufacturing Operations. Open Journal of Educational Research, 1(1), 1-19. https://doi.org/10.31586/ojer.2020.1343
[7] Mashetty, S. Data-driven insights for community investment through mortgage-backed securities. International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering (IJIREEICE), DOI, 10.
[8] Paleti, S. (2022). Financial Innovation through AI and Data Engineering: Rethinking Risk and Compliance in the Banking Industry. Available at SSRN.
[9] Yandamuri, U. S. (2022). Cloud-Based Data Integration Architectures for Scalable Enterprise Analytics. International Journal of Intelligent Systems and Applications in Engineering, 10, 472-483.
[10] Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., & Dean, J. (2019). A guide to deep learning in healthcare. Nature Medicine, 25, 24–29.
[11] Loganathan, R. (2021). Integrated Risk and Compliance Frameworks for Global Data Center Operations: A Governance-Centric Approach. Universal Journal of Computer Sciences and Communications, 1(1), 1-26.
[12] Mangala, N. (2022). Real-Time Data Quality Monitoring and Gating Frameworks in Cloud-Based Data Pipelines. International Journal of Research and Applied Innovations, 5(6), 8197-8219.
[13] Mukesh, A., & Aitha, A. R. (2021). Insurance Risk Assessment Using Predictive Modeling Techniques. *International Journal of Emerging Research in Engineering and Technology*, 2(4), 68-79. https://doi.org/10.63282/3050-922X.IJERET-V2I4P108
[14] Mattaparthi, R. (2022). Engineering Predictive Industrial Systems Through IoT-Driven Asset Monitoring and Machine Learning Prognostics. International Journal of Future Innovative Science and Technology (IJFIST), 5(1), 7790.
[15] Chakilam, C., Suura, S. R., Koppolu, H. K. R., & Recharla, M. (2022). From Data to Cure: Leveraging Artificial Intelligence and Big Data Analytics in Accelerating Disease Research and Treatment Development. Journal of Survey in Fisheries Sciences. https://doi. org/10.53555/sfs. v9i3, 3619.
[16] Ghassemi, M., Oakden-Rayner, L., & Beam, A. L. (2021). The false hope of current approaches to explainable artificial intelligence in health care. The Lancet Digital Health, 3(11), e745–e750.
[17] Reddy, V. A. R. (2022). Designing Fault-Tolerant Data Ingestion Pipelines for High-Volume Healthcare Transactions. Frontiers in Health Informatics, 11, 861-889.
[18] Nandan, B. P. (2022). AI-Powered Fault Detection In Semiconductor Fabrication: A Data-Centric Perspective. Lebcir, I., Mageswari, SU, Bhosale, YH, Nagubandi, AR, & Mahabooba, MM Agile Strategic Management in the Age of Disruption: Leveraging AI and Data Analytics for Competitive Advantage.
[19] Adusupalli, B. (2021). Multi-Agent Advisory Networks: Redefining Insurance Consulting with Collaborative Agentic AI Systems. Journal of International Crisis and Risk Communication Research, 45-67.
[20] Pamisetty, A. (2022). A Comparative Study of AWS, Azure, and GCP for Scalable Big Data Solutions in Wholesale Product Distribution. International Journal of Scientific Research and Modern Technology, 71-88.
[21] Mangala, N. (2021). CI/CD Pipeline Automation for Enterprise Data Artifacts Using Azure DevOps. Universal Journal of Business and Management, 1(1), 1-18.
[22] Inala, R. (2020). Building Foundational Data Products for Financial Services: A MDM-Based Approach to Customer, and Product Data Integration. Universal Journal of Finance and Economics, 1(1), 1-18.
[23] Maguluri, K. K., Yasmeen, Z., & Nampalli, R. C. R. (2022). Big Data Solutions For Mapping Genetic Markers Associated With Lifestyle Diseases. Migration Letters, 19(6), 1188-1204.
[24] Peddi, R. K. (2021). Optimizing Case Management Workflows in Global Data Center Colocation Services. Universal Journal of Computer Sciences and Communications, 1(1), 1-21.
[25] Polineni, T. N. S., Pandugula, C., & Ganti, V. K. A. T. (2022). AI-driven automation in monitoring post-operative complications across health systems. Global Journal of Medical Case Reports, 2(1), 32-46.
[26] Lakkarasu, P., & Kalisetty, S. (2022). Hybrid Cloud and AI Integration for Scalable Data Engineering: Innovations in Enterprise AI Infrastructure. Math. Stat. Eng. Appl., 71(4), 16774-16784.

[27] Challa, K., Burugulla, J. K. R., Pandiri, L., Pamisetty, V., & Paleti, S. (2022). Optimizing Digital Payment Ecosystems: Ai-Enabled Risk Management, Regulatory Compliance, And Innovation In Financial Services. Migration Letters, 19, 1748-1769.
[28] Pamisetty, A. (2022). Big Data can Generate Major Opportunities for Manufacturing Supply Chains. International Journal of Scientific Research and Modern Technology, 1(12), 238-251.
[29] Mashetty, S. (2022). Innovations in mortgage-backed security analytics: A patent-based technology review. Available at SSRN 5238910.
[30] Aitha, A. R. (2022). Cloud Native ETL Pipelines for Real Time Claims Processing in Large Scale Insurers. Universal Journal of Business and Management.
[31] Mangalampalli, B. M. (2021). Scalable Data Warehouse Architecture for Population Health Management and Predictive Analytics. World Journal of Clinical Medicine Research, 1(1), 1-18.
[32] Reddy, V. A. R. (2022). Data-Driven Healthcare Operations: Architecting Unified Member, Provider, and Claims Intelligence Platforms. International Journal of Science, Research and Technology, 5(5), 8511-8521.
[33] Pamisetty, V. (2022). Transforming fiscal impact analysis with AI, big data, and cloud computing: A framework for modern public sector finance. Big Data, and Cloud Computing: A Framework for Modern Public Sector Finance (November 30, 2022).
[34] Recharla, M. (2020). Targeted Gene Therapy for Spinal Muscular Atrophy: Advances in Delivery Mechanisms and Clinical Outcomes. International Journal of Science and Research (IJSR), 9(12), 1921-1934.
[35] Botlagunta, P., & Chitta, S. (2022). Advanced optical proximity correction (OPC) techniques in computational lithography: Addressing the challenges of pattern fidelity and edge placement error. Glob. J. Med. Case Rep, 2, 58-75.
[36] Sriram, H. K., Adusupalli, B., Singreddy, S., & Malempati, M. (2021). Revolutionizing risk assessment and financial ecosystems with smart automation, secure digital solutions, and advanced analytical frameworks.
[37] Murali, Revolutionizing Risk Assessment and Financial Ecosystems with Smart Automation, Secure Digital Solutions, and Advanced Analytical Frameworks (December 27, 2021).
[38] Grote, T., & Berens, P. (2020). On the ethics of algorithmic decision-making in healthcare. Journal of Medical Ethics, 46(3), 205–211.
[39] Paleti, S. (2021). Cognitive core banking: A data-engineered, AI-infused architecture for proactive risk compliance management. AI-Infused Architecture for Proactive Risk Compliance Management (December 21, 2021).
[40] Kalisetty, S., & Kothpalli Sondinti, L. R. (2022). AI-Native Cloud Platforms: Redefining Scalability and Flexibility in Artificial Intelligence Workflows. Linguistic and Philosophical Investigations, 21(1), 1-15.
[41] Yandamuri, U. S. (2021). A Comparative Study of Traditional Reporting Systems versus Real-Time Analytics Dashboards in Enterprise Operations. Universal Journal of Business and Management, 1(1), 1-13.
[42] Mashetty, S. (2022). Enhancing Financial Data Security And Business Resiliency In Housing Finance: Implementing AI-Powered Data Analytics, Deep Learning, And Cloud-Based Neural Networks For Cybersecurity And Risk Management. Migration Letters, 19(6), 1302-1818.
[43] Haleem, A., Javaid, M., Khan, I. H., & Vaishya, R. (2022). Significant applications of artificial intelligence in healthcare: A review. Current Medicine Research and Practice, 12(3), 128–134.
[44] Maguluri, K. K., & Ganti, V. K. A. T. (2019). Predictive Analytics in Biologics: Improving Production Outcomes Using Big Data.
[45] Mangala, N. (2021). Optimizing Large-Scale ETL Pipelines Using Medallion Architecture on Azure Data Lake. Journal of Artificial Intelligence and Big Data, 1(1), 1-20.
[46] Sheelam, G. K., & Nandan, B. P. (2022). Integrating AI And Data Engineering For Intelligent Semiconductor Chip Design And Optimization. Migration Letters, 19, 2178-2207.
[47] Paleti, S. (2022). Fusion bank: Integrating AI-driven financial innovations with risk-aware data engineering in modern banking. Decision Making, 2326, 9865.
[48] Pamisetty, A. (2022). Integrating Big Data, AI, and Financial Modeling in Cloud-Based Insurance and Banking Ecosystems. AI, and Financial Modeling in Cloud-Based Insurance and Banking Ecosystems (December 05, 2022).
[49] Pandugula, C., & Yasmeen, Z. (2019). A Comprehensive Study of Proactive Cybersecurity Models in Cloud-Driven Retail Technology Architectures. Universal Journal of Computer Sciences and Communications, 1(1), 1253.
[50] Inala, R. (2021). A New Paradigm in Retirement Solution Platforms: Leveraging Data Governance to Build AI-Ready Data Products. Journal of International Crisis and Risk Communication Research, 286-310.

ISSN: 3071-5717
PP: 01-18.

[51] Aitha, A. R. (2021). Dev Ops Driven Digital Transformation: Accelerating Innovation In The Insurance Industry. Journal of International Crisis and Risk Communication Research.

[52] Loganathan, R. (2022). Converging Security Architecture and Compliance Management in Enterprise Data Center Ecosystems: A Unified Control Framework. International Journal of Scientific Research and Modern Technology, 1(12), 295-312.

[53] Reddy, V. A. R. (2021). Challenges in Standardizing Member Eligibility Data Across Multi-Payer Healthcare Ecosystems. International Journal of Medical Toxicology and Legal Medicine, 24(3), 1-19.

[54] Kelly, C. J., Karthikesalingam, A., Suleyman, M., Corrado, G., & King, D. (2019). Key challenges for delivering clinical impact with artificial intelligence. BMC Medicine, 17, 195.

[55] Mattaparthi, R. (2021). Unified Data Lineage and Quality Governance Framework for Multi-Source Sensor Streams in Heavy-Duty Powertrain Manufacturing. Online Journal of Mechanical Engineering, 1(1), 1-15.

[56] Chava, K., Chakilam, C., & Recharla, M. (2021). Machine Learning Models for Early Disease Detection: A Big Data Approach to Personalized Healthcare. International Journal of Engineering and Computer Science, 10(12), 25709-25730.

[57] Adusupalli, B. (2021). Multi-Agent Advisory Networks: Redefining Insurance Consulting with Collaborative Agentic AI Systems. Journal of International Crisis and Risk Communication Research, 45-67.

[58] Pamisetty, V. (2021). A cloud-integrated framework for efficient government financial management and unclaimed asset recovery. Available at SSRN.

[59] Krittanawong, C., Johnson, K. W., Rosenson, R. S., Wang, Z., Aydar, M., & Baber, U. (2021). Deep learning for cardiovascular medicine. Journal of the American College of Cardiology, 78(6), 589–600.