Deep Science Publishing, 2026
https://doi.org/10.70593/978-93-7185-592-1_2

DeepScience
Open Access Books

Chapter 2: Modern Data Engineering for Real-Time Intelligence

2.1. Introduction

Recent advances in technology and the ongoing push toward digital transformation have strongly affected the engineering of data- and intelligence-centric systems. The deployment of Cloud infrastructures and technologies has introduced elasticity and scalability capabilities that can significantly lower the price of computation for batch analysis. At the same time, the increasing availability of inexpensive sensors and the improvement of cheap networking technologies have opened the path toward the Internet of Things (IoT) and the connected world. For years, systems were built mainly around these two elements—the vertical and horizontal scalability of Cloud infrastructures, and the data generated by the connected world. These systems were capable of gathering information from multiple sources into huge centralized systems, where the data could be processed in larger volumes.

The maturity of the previously mentioned technologies has inspired a shift of focus in the design of data- and intelligence-centric systems. Rather than engineering solutions mainly capable of looking back in time to react to events, with the hope that the same events would not reoccur again, systems are now being engineered to predict future events, to be proactive, and to offer suggestions or actions before the events happen. To accomplish this goal, new approaches for data collection, processing, and intelligence generation are required. As events are unpredictable in nature, real-time intelligence derives its value from the engineering of systems capable of taking decisions as close to the moment of an event as possible. Such intelligent decisions can be used to monitor financial markets, guaranteeing risk levels, or to improve operational performance in logistics and manufacturing, detecting potential functional failures in machines and supply chains before they happen.

2.1.1. Background and Significance

The emergence of Big Data has spurred interest and research in the area of data engineering. While attempts have been made to distinguish data science from data engineering, the newly adopted distinction does not take into account the properties of the data life cycle. Data engineering is concerned with the preparation of data for analysis. Any data analysis, whether it is exploratory, prescriptive, or predictive, has clear requirements on the structure, quality, and amount of data consumed. Real-time intelligence or analysis can be considered a special case of data engineering, as it is an opinionated, derivative form of prescriptive prediction that is designed and optimized to answer a small number of pre-defined questions. At presented here, real-time intelligence is used as a guide to the requirements, constraints, issues, techniques, technologies, and tools that must be wielded and controlled to make data ready for consumption in real-time.

The requirements of real-time intelligence influence not only data ingestion and processing, but also the results that are generated. Properties such as temporal correlation structures and similarity metrics are made explicit, and the analysis is designed as a query-on-query process.

Fig 2.1: Modern Data Engineering for Real-Time Intelligence

2.1.2. Research design

The analysis offers perspectives on Modern Data Engineering through three interdependent lenses. First, the author examines the most relevant technological foundations that enable real-time data engineering systems. Second, a selection of architectural patterns adopted by data engineers is provided. Lastly, the technological components supporting these patterns are described under the Modern Data Engineering label. Although these categories do not align with unified development phases, the analysis is justifiably considered a research design that addresses the emerging discipline.

Modern Data Engineering returns the focus to the development journey of streaming data systems. Initial attention is directed toward data ingestion architectures, stream processing paradigms, and their implications for data quality and consistency. Architectures are then presented through the Lambda and Kappa patterns, along with event-driven and CQRS-based approaches. These three categories allow the subsequent description of technological components—messaging and streaming platforms, real-time transformation and enrichment tools—united by the role of enablers. Finally, the classification of use cases and the evolution of key metrics complete the analysis.

2.2. Foundations of Real-Time Data Engineering

Real-time data engineering utilises data streamed into the system and processes it in flight so that systems can infer information about the underlying real-world processes operating in the environment the system serves. This foundation defines architectures that steer engineering decisions, outline the requirements of stream processing systems that act on the arriving data, and provide a catalogue of patterns usable within these architectures.

Data is arriving in ever-increasing volume and variety, with responders to emergent events and the exploitation of fleeting opportunities needing to ingest this data, create information, and present this information for decision making at high speed. A central security operations centre needs to sort through large bursts of data arriving from multiple monitoring systems and, using a variety of analytic techniques, detect, make sense of, and react to security threats. Financial traders take a position based on market information, but high-frequency traders must react to wealth of information that arrives in microseconds and shift their positions before more normal human traders can react. Logistics are highly interdependent, and increasing globalisation coupled with just-in-time delivery schedules mean that there are incentives to optimise to the limits of feasibility."

2.2.1. Data Ingestion Architectures

Different data processing architectures can be distinguished based on how data is ingested into the processing system.

As part of a **data lake** architecture, one of the oldest paradigms for data ingestion is batch loading of data through a periodic Extract-Transform-Load (ETL) pipeline. With this approach, additional functionality such as data schema validation, type casting, enrichment, and data quality checks are applied while moving data from a data source into a data lake. The prepared data then becomes available for analysis by business or data science teams. Event-driven data ingestion employs a Message Queue (MQ) technology that supports the Pub/Sub pattern, where applications publish data (events) and subscribers listen for new events continuously. Although batch-mode ingestion and data lakes are no longer state of the art, they remain a valid approach for many use cases. The additional delay imposed by ETL by a periodic data load can be acceptable for many business scenarios. Moreover, the batch processing engine supports adequate data quality checks before making data available to analysis.

Message Queues and **Streaming platforms** can be employed to support streaming, event-based data ingestion from one or multiple sources at any time. This is often referred to as the **Change Data Capture** (CDC) pattern: as changes occur in the data source, the change becomes available and progresses through the message queue or streaming platform. Carefully designed subscribers process the changes and apply them to their dedicated sinks.

2.2.2. Stream Processing Paradigms

The discussion of real-time intelligence would not be complete without an understanding of the various speed extremes under which stream processing systems vary, the trade-offs made by each paradigm, and their corresponding execution engines and ecosystems. Four processing paradigms for streaming systems are identified and compared along several key axes: Reactive Streams, Micro-batching, Continuous Queries, and Local Stream Processing. Each paradigm's axes of trade-offs reveal system-specific qualities.

Four different speed extremes characterize stream processing systems, namely Reactive Streams, Micro-batching, Continuous Queries, and Local Stream Processing. Reactive Stream Processing covers the open-loop case where no true inner-loop latency reduction occurs, yet stream consumption reactivity improves when introducing fixed-speed producers whose output is consumed with minimum delay. These pumping streams and their processing-dedicated executors suffice when supportfully ordered unchecked high-speed random-access streams appear at some consumption point. Thus, Reactive

Streams require ordering only for the higher latencies they enable without pressure reduction. Reset events may also drive such open loops.

Moving to later stages, the Micro-batching paradigm closes the outer loop on data movement without overwriting any semantics of Real-Time Intelligence and further reduces lower-bound latencies as inner loops pass through control. When all actors are ready, data enters the shared buffer, and their higher latency is thereby adjusted to a fixed speed without additional cost. Proper batching certainly fixes latencies from event-at-a-time processing but should preferably not add noticeable latencies to reaction times for most events. Data tables whose periodic replicates within maintenance windows keep staleness bounded naturally lend themselves to Micro-batching.

2.3. Architectural Patterns for Real-Time Intelligence

Architectural outlines for the integration of real-time processing applications in the broader data engineering ecosystem are collected in two families. The first comprises revised versions of the Lambda and Kappa architectures, two influential solutions for processing both stream and batch data. The second relates to event-driven architectures and Command-Query Responsibility Segregation (CQRS), which support use cases in operational and event monitoring for fast decision-making across multiple domains.

Lambda and Kappa Architectures Revisited

The original Lambda Architecture employs micro-batching for both the cold data store and the serving layer. The Kappa Architecture shifts the design to pure stream processing, allowing the system to be seen as a single continuous data-flow function. A re-interpretation of both architectures shows them as special cases of a broader design space, whose dimensions are the data ingestion mode, the processing paradigm, the merging function, and the serving layer. In these terms, the original architectures appear as combinations of specific parameter values with pseudo-bulk processing for both the cold and warm archives, and with the serving layer realized through a stream-table converter. Most importantly, Kappa and the broader design space share the principle of a serverless cold-store engine. Conversely, when backward compatibility is not a requirement, Lambda can adopt stream-parallel evolution over the micro-batches. Such considerations lead to new rethought architectural versions.

2.3.1. Lambda and Kappa Architectures revisited

Lambda architecture is a widely used pattern for real-time data engineering. It consists of a batch processing layer, a stream processing layer, a serving layer, and a data sink. The batch processing layer computes correct and complete views of the data stored in an

immutable landing zone, usually a data lake in modern implementations. The stream processing layer serves as a means for fast data processing, and the serving layer combines results from both layers to support low-latency queries.

Lambda architecture's design has several ramifications. First, data is processed both in batch and streaming mode. Although both modes can return different results, the serving layer reconciles them. Second, data can be reconstructed at any time point, allowing for fault-tolerant and consistent processing. However, keeping data in the landing zone is costly. Third, the architecture computes correct results but at a higher latency. Real-time users can query the serving layer to obtain fast results, while batch-processing users can query the batch-processing layer to obtain complete results.

Lambda architecture also has limitations. Maintaining two processing approaches can double operational costs. Results are stored twice, and continuous reconciliation adds a bottleneck. Moreover, data must be kept in an immutable landing zone to ensure consistency. To mitigate these drawbacks, Kappa architecture suggests using only a streaming processing engine to process a single stream of data in real time and spawn a new process to recompute views whenever applications need them.

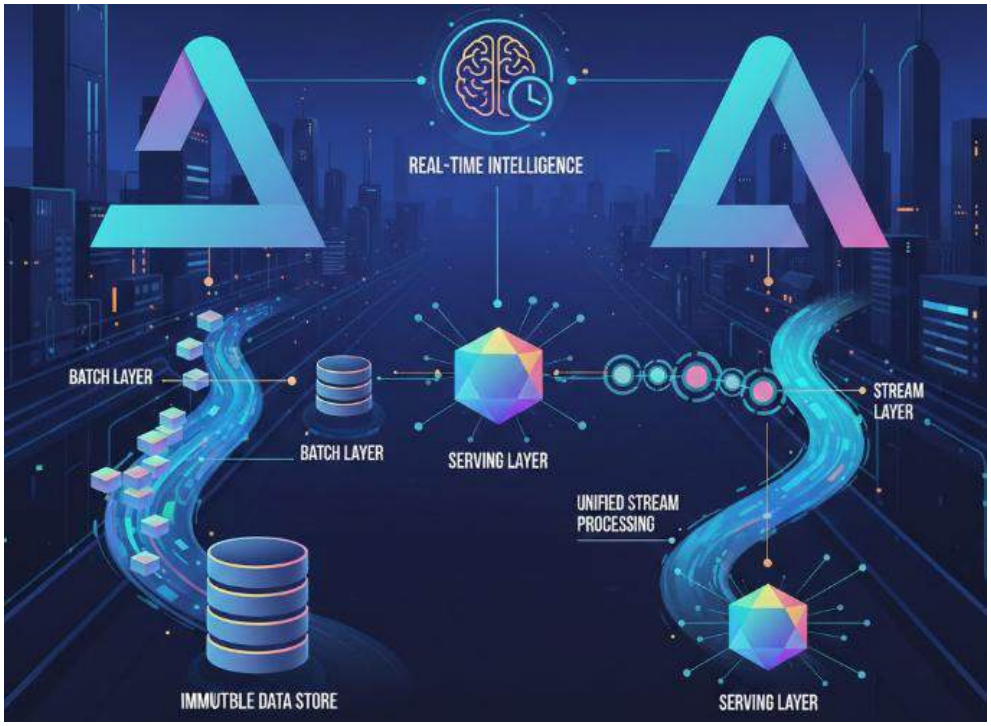

Fig 2.2: Lambda and Kappa Architectures revisited

2.3.2. Event-Driven and CQRS Approaches

The recently coined term “event-driven architecture,” implies a broad and pervasive architectural style that seems to have little in common with designs based on the hardly related classic event-driven programming paradigm. Every architecture is driven to some extent by a main processing path: in Event-driven systems, such as business process automation and workflow systems, the primary external events flow into a long-running process. Event-driven architecture (EDA), as implemented with message queues, is simply a technique for decoupling the producer code of an application from its consumer. With the inevitable increase in event-processing volume, data-staging components evolve into batch-like systems. External events are queued up for non-real-time and subsequent maybe-time processing. The C4 model (connect, collect, consume, control for analysis) has recently been rebranded as the CQRS style: Connect = Command, Collect = Queries, Consume = Asynchronous Communication.

Another popular architecture pattern, heavily inspired by Domain-Driven Design (DDD) principles, is CQRS (Command Query Responsibility Segregation), which states that commands should be separated from queries, and as a corollary, that they can use different models for updates and reads. The Command-side of CQRS implements data changes, while the Query-side involves customized denormalized read models based on up-to-date Data Lakes or Data Drains integrated into the target data sources in a Novell Cool Races way. CQRS, however, should not be confused with Event Sourcing, and there are many variants around, such as Event-Driven CQRS or Command-Shaped CQRS. CQRS are typically regarded as styles available via any reliable event-processed messaging middleware. CQRS do not solve anything in themselves; they merely highlight the simplifications that can be brought in, especially when combined with other techniques.

2.4. Technologies and Toolchains

Constructing a solid foundation for real-time data engineering requires careful selection and integration of a variety of appropriate technologies and tools to enable efficient implementation of critical architectural patterns, robust messaging functionalities, and necessary stream processing operations. A dedicated messaging or streaming platform is typically at the center of the architecture. Providing publish-subscribe message dissemination, as well as durable message storage, partitioned message queues, scalable consumption, and processing for multiple message subscribers, such platforms retain all message/stream data for a defined period or until a specified retention policy is invoked. Such systems allow for decoupling of producer and consumer applications, function as highly available scaling components, enable application development with the event-driven paradigm, and facilitate optimized support for Complex Event Processing (CEP).

Once developed, enrichment messages can also be published back to the messaging platform, where they can be leveraged by analytical and intelligence applications, and subsequently removed or archived by appropriate consumers. Many organizations employ these platforms to address specific requirements for greater flexibility, scaling, and/or reliability for message transportation across a variety of predefined and ad-hoc use cases and applications. Stream-processing tools and systems that build on these communication platforms commonly provide the functionality needed for query definition and execution for event detection with potential alerting, integration of messaging QoS models, summary statistics and indicators, and/or CDC for database updates. Examples include Storm, Flink (and over/Kafka), Spark Streaming, and, for the SQL-theoretic community, Samza, and Esper.

2.4.1. Messaging and Streaming Platforms

Many applications, especially those in financial services, need to process data at high throughput and low latency. To meet these requirements, data engineering patterns and technologies that enable low-latency data processing have gained popularity. Examples include Lambda and Kappa architectures, Event-Driven and CQRS approaches, Data Mesh concepts, and others. The tools that support these approaches are also maturing rapidly, such as the streaming framework Apache Flink, the event streaming platform Apache Pulsar, and the scalable stream processing engine Confluent. Nonetheless, other data engineering technologies remain important. Mission-critical applications need to ensure timely, reliable, and consistent processing of data, and these nonfunctional requirements can encompass quality, consistency, safety, and regulatory compliance.

Messaging and event streaming platforms like Kafka and Pulsar transport and persist data for later offline processing in batch or micro-batch mode. They decouple the source of the events from the sinks, which either consume these events directly or use them to populate materialized views. They enable systems to implement a reactive model, where the system reacts to events. These products often cater to a Publish/Subscribe model of usage as well. Because of the high data throughput, they are often deployed in a tiered cluster architecture, where older data is stored at lower-cost hardware or less powerful nodes. Tools like Apache Hudi and Delta Lake layer support managing the lifecycle of such tiered data.

2.4.2. Real-Time Data Transformation and Enrichment

Transformation and enrichment of ingested data streams cover a rich array of processing scenarios in which data formats are converted, new features derived, and data merged for more useful models. The common property of these operations is that they do not

need to abide by the strict conditions on latency, scalability, fault-tolerance, and durability imposed on the specialized tools handling the ingestion. When working with data at scale, different tool choices are usually made for they have inherently different requirements. Here, the three main types of real-time transformation and enrichment tasks are introduced in turn.

The first type is conversion of the ingested data to a more amenable representation. Often it is needed to map JSON documents onto a fully-structured canonical representation, such as Parquet or ORC. Convertitude has been specifically built for this task. It uses a hybrid model (SQL for structural mapping, Scala for functional mapping) so that both type of transformations can leverage their intrinsic strengths. The tool can execute the workload on a dedicated converter or on a streaming cluster. Additionally, though optimization was not part of the initial design, the execution for both execution modes is just an optimized converting layer that can combine multiple transforms for efficient execution.

2.5. Data Quality, Consistency, and Compliance

Real-time data engineering exposes organizations to new data quality challenges. In a Lambda/Kappa architecture, a batch subsystem is responsible for generating a trusted, master dataset maintained in a data lake. However, a more practical solution for real-time intelligence systems operating in a fast data paradigm needs to guarantee data quality at all processing layers and to make data remediation possible even when working with late-arriving data.

Fig 2.3: Data Quality, Consistency, and Compliance

In streaming environments, consistency is commonly expressed with respect to a given consistency model. An important property here is that an application may tolerate data

inconsistency for a limited time frame (e.g., during peak hours), but persistence in a non-consistent state over a longer period may incur serious risks. Supporting temporal deviation from consistency can be achieved by applying forking patterns on transactional data, thus allowing progressive correction of processing outputs. Moreover, streaming environments can accommodate non-linear dependencies between events in a controlled manner, as well as out-of-order processing of late-arriving data. Finally, recent research works in Cambridge University suggest that data provenance for processing warehouses can be extended to the integration of DLT-based transaction queues with real-time ETL processes.

2.5.1. Consistency Models in Streaming Environments

The more the application of distributed stream processing systems becomes mainstream, the more the demand for strong consistency guarantees arises, yet many systems and programming models tend to trade strong consistency for data availability. Traditional notions of consistency, which stem from the CAP theorem for replicated storage, are ill-suited in a stream processing context because the reliable transport of data through messages-passing middleware systems—communicated through data stream channels—introduces an implicit first-in-first-out ordering. Streaming consistency variations thus focus specifically on providing guarantees within the processing subsystem of a distributed system.

Appropriate streaming consistency models are significant from both a practical system-design and an application-programming perspective. From a system-design viewpoint, the achievement of real-time performance is often conflated with a trade-off on consistency. Consequently, applications programmed in an intuitive way are susceptible to introducing subtle distribution-related inconsistency bugs. The adoption of a streaming-agnostic architecture with a proper programming model helps cover much of the application development cycle but cannot eliminate distribution-centric homogeneity in large stream-processing topologies.

2.5.2. Lineage, Provenance, and Auditing

Two generic challenges of working with data in general are quality and security. Data can suffer from quality issues that stem from its origin, infer from being stale or incomplete and result in inconsistency when different copies pose different answers to a query. Ideally, data should be controlled throughout its life cycle, from creation to consumption. This includes maintaining its lineage (record of transformations applied) and provenance (origin, other processes it participated in), data auditing functions to validate and keep track of modifications as well as guarantee compliance with privacy

regulations. In streaming environments, which provide access to information during its ingestion journey, quality, safety and lineage measures can only protect the data assets in transit. The points introduced are present in data engineering for intelligence (dependent of data freshness) and even more prominent in event-driven systems, where data is mutable and events may carry data removing originally documented auxiliary constraints.

Every processing step leading to the final consumer of data information should guarantee its effective correctness and allow tracing the sources (and sources of the sources) from where the information originated, but such measures are invariably delegated. Data stream processors provide easy integration with Kafka Connect and Debezium to impose this constant checking and recovery of quality and security for any Kappa-based architecture. The new released parsing and presentation of financial time series have been tested with it, allowing to automatically create health checks and audit logs with a few annotations hidden behind a factory method. A query auditing might hide yet another initial query designed to check permissions for making each original query at the given time. Tracing events and detecting failures in real time, in both consumer and data preparation sides, warrants covering the entire end-to-end objectives and allows detecting eventual feelers in the control chain.

2.6. Real-Time Intelligence Use Cases and Metrics

Financial market data and risk monitoring Transactions originating from multiple liquidity venues generating financial market data must be ingested, stored, enriched, aggregated, and exposed within a few hundred microseconds of their occurrence for computations and update messages to be generated at real-time speeds. The risk of holding exposed positions in tradable instruments must be monitored continuously and mitigated through automatic clearing house mechanisms. Risk models utilizing all transaction data as input must produce predictions that are distributed to trading systems in around one second. Manufacturing and logistics operational productivity metrics must be computed and exposed with a delay of several minutes after real-time replenishing trigger events. Dedicated end-to-end flow delivery test solutions achieving a bounding real-time symptom emergence time in the order of one minute are also needed.

Operational intelligence in manufacturing and logistics tracking devices monitoring the precise location and motion state of manufacturing and logistics resources must be monitored, exposing current semantic location for any customer on a map in real time. The monitoring service must also expose a filtering-based, query-selectable event sink for the movement pattern, covering customer divisions, and triggering replenishing event detections by means of specialized plugins as necessary. Dedicated event delivery test

solutions achieving a service incarnation bounding deployment time in the order of minutes complement the service.

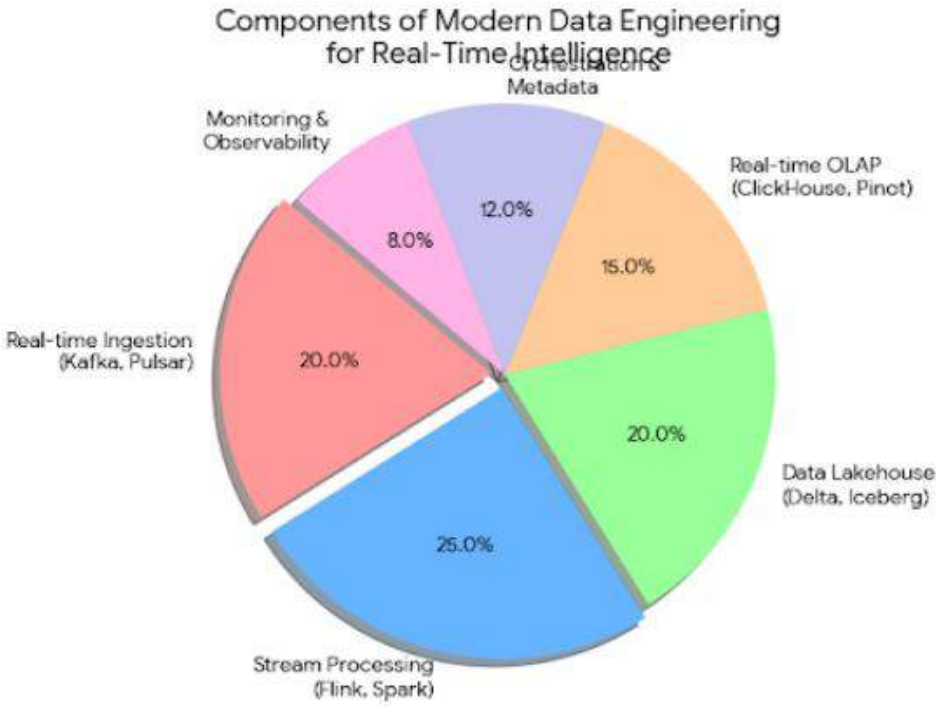

Fig 2.4: Modern Data Engineering for Real-Time Intelligence

2.6.1. Financial Market Data and Risk Monitoring

Financial market data contains a wealth of information about the economic condition of different countries, trends of various-industries, and the performance of individual companies. Investors, analysts, data scientists, and other stakeholders are highly interested in financial market data analytics.

A single global asset class has an estimated notional value of approximately 389 trillion, with the underlining country bonds, treasury bills or cash deposits continuing to be the benchmark for establishing investment risk. Apart from being the world's natural zero-risk free asset class, sovereign debt market is also closely correlated with market's risk appetite and market movements across the world. Many investment companies are heavily buying or liquidating swaps, treasury notes, and government bonds in response to capital protection demand from their investors. As a result, a lot of changes are happening in these product areas. Monitoring data across these related product areas help

identify such movements and pro-active company actions can provide strong performance against the competition.

In futures and options market, market participants are anticipating and also managing their risk with respect to the future movement of the underlying asset prices. Strong and unusual buying/ selling activities are seen at calls and puts. Hence keeping track of the "open interests 2" across these products can detect trade or market movements impending.

2.6.2. Operational Intelligence in Manufacturing and Logistics

Real-time intelligence in operational domains focuses on situational awareness, process visibility, and parameter monitoring. Such intelligence enables organizations to make decisions based on the actual state of critical parameters and ongoing processes. Situational awareness and operational intelligence concerns may also relate to assets, such as fleet composition and locations, as well as a company's own properties, installations, and inventories. These aspects are part of a broader monitoring framework that offers insight into the operations conducted by the organization, its partners, and competitors. Organizations invest substantial time and effort to access real-time data on the relevant parameters; accurate and timely intelligence contributes to the avoidance of delays or the enabling of rapid responses.

The generated intelligence is employed for management or analytical purposes and covers either a single organizational domain (process monitoring) or multiple domains (situational awareness). Situational awareness encompasses a broader scope and usually involves many actors, including suppliers and service providers, whose processes, parameters, and documents are monitored and cross-referenced with peers. Such monitoring facilitates the detection of problematic conditions or events, the alerting of responsible parties, and the triggering of corrective actions. The monitoring scope may extend beyond the process supply chain to encompass competitors, which require coordinated effort and data collection for each single data source—either continuously or at selected intervals, or companies may simply purchase or lease data from third parties.

2.7. Conclusion

Real-time intelligence enables organizations to react quickly to events, improve customer experiences, and optimize operations. Modern Data Engineering for real-time intelligence introduces the required foundations, architectural patterns, technologies, toolchains, and considerations, building upon previously discussed true data engineering

principles and the modern demand for real-time data processing, analysis, and analytics. Data engineers are now required to capture, process, and deliver ever-evolving data in motion, resulting in increasingly published data models, definitions, transformations, and readiness for analytics or serving within Business Intelligence and Analytics solutions.

Fundamentally, data processes for real-time intelligence enable organizations to detect and respond to events as they happen, rather than examining historical data at a later date. Importance drivers include better experience at low latency in web and mobile applications that are enabled by recommendation engines, the speed at which financial markets operate, faster detection of data-related errors that help organizations meet compliance regulations such as Basel III and MiFID II, online fraud detection during customer transactions, reducing wastage in manufacturing and delivery networks, and many more in the fields of Health, Life Sciences, Insurance, Retail, Telco, Travel, and Transportation. The next phase of a Data Engineering transformation is defined as Modern Data Engineering for Real-Time Intelligence.

2.7.1. Future Trends

Future trends in data engineering for real-time intelligence will be influenced by several factors. The continuing democratization of data analytics, made possible by business user-oriented data preparation and analytics platforms such as Tableau, Power BI, and Qlik can be expected to raise demand for integrated business-facing data products. Data preparation systems that sit on top of existing data lakes to nimbly extract subsets of data for analytics explorations, while leaving the data in the lake in a raw unmodified state. Real-time event auditing systems (audit logs) are likely to become more common in data repositories to provide a reliable record of what has happened and at what time (and thus eligible for regulatory demand).

Micro-batch and near real-time systems remain important because of the backing of the major cloud providers. Micro-batch processing operationalizes all the manual mini-batch activity already being done by business users; all it requires is a scheduler. Streaming-designed data preparation systems, such as by Meltano or Fivetran, are also likely to emerge. Cloud messaging and streaming platforms supporting serverless architectures, such as Google Pub/Sub, put the onus of event-based architectures onto the developer rather than the operations team. Until they mature toward a shared service within an organization, however, the micro-batch and platform services will hold sway over the built-for-purpose streaming systems..

References

Anand, S. (2025). Next-generation data engineering architectures for real-time health data interoperability. Journal of Recent Trends in Computer Science and Engineering, 13(2), 93–110.

Ashokkumar, S., Amistapuram, K., C, Bharathi., M, Dhanamalar., & J, Gokulraj. (2025). Attention-Guided Spatial Temporal Framework for Deepfake Detection on Social Video Platforms. In 2025 International Conference on Communication, Computer, and Information Technology (IC3IT) (pp. 1–6). IEEE. 2025 International Conference on Communication, Computer, and Information Technology (IC3IT). https://doi.org/10.1109/ic3it66137.2025.11341690

Annuš, J. (2024). Digital learning in the 21st century: Trends, challenges, and innovations in technology integration. Frontiers in Education, 10. https://doi.org/10.3389/feduc.2025.1562391

Enterprise-Scale Gen AI Orchestration Using Small LMs and LLM Agents for Intelligent ITSM and HRSD Automation in Enterprise Ecosystems. (2025). MSW Management Journal, 35(2), 1889-1897.

Demirezen, M. U., & Navruz, T. S. (2023). Performance analysis of Lambda architecture-based big-data systems on air/ground surveillance application with ADS-B data. Sensors, 23(17), 7580. https://doi.org/10.3390/s23177580

Hazem, H., Awad, A., & Yousef, A. H. (2023). A distributed real-time recommender system for big data streams. Ain Shams Engineering Journal, 14, 102026. https://doi.org/10.1016/j.asej.2022.102026

Ismail, A., Sazali, F. H., & Jawaddi, S. N. A. (2025). Stream ETL framework for twitter-based sentiment analysis: Leveraging big data technologies. Expert Systems with Applications, 261, 125523. https://doi.org/10.1016/j.eswa.2024.125523

Thutari, R. T., Garapati, R. S., B M, Manjula., R K, Supriya., & M, Senbagan. (2025). Adaptive Access Control and Authentication Management for IoT Using Attention-GRU and Reinforcement Learning. In 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON) (pp. 1–6). IEEE. 2025 2nd International Conference on Software, Systems and Information Technology (SSITCON). https://doi.org/10.1109/ssitcon66133.2025.11342003

Nigam, N., Sireesha, B., Renuka, Ediga, P., Segireddy, A. R., & Bokde, S. (2025). Comparative Evaluation of Cloud Security Algorithms Using Multiple Classifiers with an Optimized Intrusion Detection System. In 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1–6). IEEE. 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG). https://doi.org/10.1109/ictbig68706.2025.11323642

Ismail, L., et al. (2024). AI-powered smart grids: Revolutionizing energy efficiency in railroad operations. International Journal of Computer Engineering and Technology, 15(5), 981–991.

Gupta, D. K., Purushotham, K., Dheer, G., P, S., Gottimukkala, V. R. R., & Kapoor, S. (2025). Semantic Feature Learning Using Transformer-Based Deep Neural Networks. In 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1–6). IEEE. 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG). https://doi.org/10.1109/ictbig68706.2025.11323734

Khriji, S., et al. (2022). REDA: A cloud-based event-driven architecture for real-time data processing in wireless sensor networks. Journal of Sensors.

Kumar, I., Nagabhyru, K. C., G, Naveen. I., V, Prabhakaran. M., & V, Sruthy. K. (2025). Adaptive Meta-Knowledge Transfer Network with Feature Hallucination and Attention for Low-Shot Object Detection in Aerial Images. In 2025 International Conference on Communication, Computer, and Information Technology (IC3IT) (pp. 1–6). IEEE. 2025 International Conference on Communication, Computer, and Information Technology (IC3IT). https://doi.org/10.1109/ic3it66137.2025.11341447

Varri, D. B. S. V. (2025). Human-AI collaboration in healthcare security.

Liu, J., Soria, R., Wu, X., Wu, H., & Shang, Z. (2020). The SiTian project. arXiv. https://doi.org/10.48550/arxiv.2006.01844

Davuluri, P. S. L. N. (2020). Event-Driven Architectures for Real-Time Regulatory Monitoring in Global Banking. Universal Journal of Business and Management, 1(1), 1–14. Retrieved from https://www.scipublications.com/journal/index.php/ujbm/article/view/1362

Miletić, A., Lukovac, P., Naumović, T., Stojanović, D., & Labus, A. (2023). A data streaming architecture for air quality monitoring in smart cities. Athens Journal of Technology & Engineering, 10(4), 215–227. https://doi.org/10.30958/ajte.10-4-2

Jagtap, S., Inala, R., Venu, M., & Divya, T. V. (2025, October). Large-Scale Crowd Flow Prediction Using Temporal Convolutional Network with Spatio-Temporal Attention. In 2025 International Conference on Communication, Computer, and Information Technology (IC3IT) (pp. 1-6). IEEE.

Muvva, S. (2023). Blockchain technology in data engineering: Enhancing data integrity and traceability in modern data pipeline. International Journal of Leading Research Publication, 4(7).

Guntupalli, R. (2025, June). AI-Powered Data Analytics in Cloud Computing. In International Conference on Data Analytics & Management (pp. 280-289). Cham: Springer Nature Switzerland.

Raptis, T. P., Cicconetti, C., & Passarella, A. (2024). Efficient topic partitioning of Apache Kafka for high-reliability real-time data streaming applications. Future Generation Computer Systems, 154, 173–188. https://doi.org/10.1016/j.future.2023.12.023

Raptis, T. P., et al. (2023). Engineering resource-efficient data management for smart cities with Apache Kafka. Future Internet, 15(2), 43. https://doi.org/10.3390/fi15020043

Yandamuri, U. S. AI-Driven Decision Support Systems for Operational Optimization in Hospitality Technology.

Sathupadi, K., Achar, S., Bhaskaran, S. V., Faruqui, N., & Uddin, J. (2025). BankNet: Real-time big data analytics for secure internet banking. Big Data and Cognitive Computing, 9(2), 24. https://doi.org/10.3390/bdcc9020024

Varma, Y. (2024). Real-time fraud detection with Graph Neural Networks (GNNs) in financial services. Los Angeles Journal of Intelligent Systems and Pattern Recognition, 4, 224–241.

Pallapu, S. R., Aitha, A. R., K, Sudhakar., Vandhana, K., & Chelladurai, S. (2025). GAN-Augmented Transformer Framework for Cross-Domain Video Style Transfer. In 2025 International Conference on Communication, Computer, and Information Technology (IC3IT) (pp. 1–6). IEEE. 2025 International Conference on Communication, Computer, and Information Technology (IC3IT). https://doi.org/10.1109/ic3it66137.2025.11341104

Zhou, Y., et al. (2025). Unleashing the potential of large language models in urban data analytics: A review of emerging innovations and future research. MDPI Analytics, 8(6), 201. https://doi.org/10.3390/2624-6511/8/6/201

Polamarasetti, S., Kakarala, M. R. K., kumar Prajapati, S., Butani, J. B., & Rongali, S. K. (2025, May). Exploring Advanced API Strategies with MuleSoft for Seamless Salesforce Integration in Multi-Cloud Environments. In 2025 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) (pp. 1-9). IEEE.