International Journal Of Management And Economics Fundamental

VOLUME Vol.06 Issue01 2026
PAGE NO. 122-128

From Data Streams To Decision Intelligence: Cloud-Native Data Warehousing Architectures

Dr. Ibrahim Al-Hassan

Department of Computer Science, University of Toronto, Canada

Received: 15 December 2025; **Accepted:** 12 January 2026; **Published:** 29 January 2026

Abstract: The convergence of real-time data streams, cloud-native computing, and modern data warehousing has transformed how organizations derive value from data. Enterprises no longer rely solely on periodic batch processing and static analytical repositories; instead, they demand continuous ingestion, rapid transformation, and near-instant analytical insight to support operational and strategic decision-making. This shift has been driven by the proliferation of Internet of Things devices, digital platforms, cybersecurity monitoring, and data-intensive healthcare applications, all of which generate vast volumes of high-velocity, heterogeneous data. Within this evolving landscape, cloud-native data warehouses such as Amazon Redshift have emerged as central analytical backbones capable of integrating streaming and historical data while providing elastic scalability, high availability, and advanced analytical capabilities (Worlikar, Patel, & Challa, 2025). Yet despite the availability of sophisticated platforms, the theoretical and architectural foundations for integrating real-time stream processing, multiprocessor scheduling, and warehouse-centric analytics remain fragmented across disparate research traditions.

This article develops a comprehensive, publication-ready framework that unifies stream processing architectures, cloud-native data warehousing, and real-time scheduling theory into a coherent model for intelligent, large-scale analytics. Drawing on literature from big data stream analysis, distributed processing frameworks, real-time systems, cybersecurity, and healthcare analytics, the study argues that the performance and reliability of modern analytical systems are as dependent on scheduling and resource allocation as they are on data models and storage engines. Prior research has extensively examined individual components such as Apache Kafka, Spark, Storm, and Flink, as well as real-time scheduling algorithms for multiprocessor systems, yet few works have connected these layers to the warehouse-centric analytics that organizations ultimately depend on for decision-making (Kolajo, Daramola, & Adebiyi, 2019; Babcock et al., 2004; Anderson & Devi, 2006).

Methodologically, the study adopts a qualitative, theory-driven synthesis of the provided references, interpreting their empirical and conceptual contributions through the lens of cloud-native data warehousing. By positioning Amazon Redshift as an analytical anchor that interacts dynamically with streaming pipelines and scheduling frameworks, the article demonstrates how real-time business intelligence, cybersecurity monitoring, and healthcare analytics can be supported in a unified architectural paradigm (Delen et al., 2018; Alam et al., 2024; Buczak & Guven, 2016). The results reveal that performance, fairness, and quality of service in modern data warehouses are emergent properties of distributed scheduling, stream processing semantics, and storage-compute decoupling rather than isolated platform features.

The discussion extends these findings by engaging with competing scholarly perspectives on scalability, latency, and reliability in distributed analytics. It argues that future research must transcend platform-specific benchmarking and instead develop theoretically grounded models that integrate real-time scheduling, data stream management, and cloud-native warehousing. In doing so, the article contributes a rigorous, interdisciplinary foundation for the next generation of intelligent, real-time data warehouses capable of supporting mission-critical decision-making in complex digital ecosystems.

Keywords: Cloud-native data warehousing, real-time stream processing, distributed scheduling, big data analytics, Amazon Redshift, intelligent decision systems

INTRODUCTION: The last two decades have witnessed a profound transformation in how data is produced, processed, and interpreted. What was once an environment dominated by relatively slow-moving transactional records and periodic reporting cycles has become an ecosystem of continuous digital exhaust generated by sensors, mobile devices, social platforms, and cyber-physical systems. This shift has compelled organizations to rethink not only their analytical tools but also the very architectures that support data-driven decision-making. In this context, the emergence of real-time stream processing frameworks and cloud-native data warehouses has marked a decisive break from traditional, batch-oriented business intelligence. The theoretical and practical implications of this break are far-reaching, touching on questions of scalability, fairness, reliability, and the epistemological status of real-time knowledge itself (Kolajo et al., 2019).

Early data warehousing systems were designed around the idea of stable, slowly changing datasets that could be periodically extracted, transformed, and loaded into centralized repositories. Such systems aligned well with managerial decision-making models that emphasized retrospective analysis and periodic planning. However, as digital platforms and networked devices proliferated, the latency inherent in batch-oriented architectures became increasingly problematic. Real-time business intelligence research has demonstrated that decision-makers behave differently when they have access to continuously updated analytical insights, often shifting from reactive to proactive strategies (Delen et al., 2018). This behavioral transformation implies that the technical architecture of data systems is not merely an operational concern but a determinant of organizational cognition and strategy.

The rise of stream processing frameworks such as Apache Kafka, Storm, Spark, Flink, and Samza reflects an attempt to address this latency problem by enabling continuous ingestion and processing of high-velocity data (Kleppmann & Kreps, 2015; Katsifodimos & Schelter, 2016). These frameworks provide abstractions for handling unbounded data streams, windowed computations, and fault-tolerant state management. Yet while they excel at near-real-time computation, they are not, by themselves, complete analytical environments. Organizations still require robust data warehousing platforms that can store, index, and query both historical and streaming-derived data in a unified manner. This is where cloud-native data warehouses, exemplified by Amazon Redshift, play a critical role by bridging operational analytics and long-term data management (Worlikar et al., 2025).

Amazon Redshift represents a new generation of data warehouses that are designed from the ground up for cloud environments. Unlike traditional on-premise warehouses, it decouples storage and compute, supports elastic scaling, and integrates seamlessly with streaming and data lake ecosystems. The Redshift architecture enables organizations to ingest streaming data from services such as Kafka or cloud-native streaming platforms, transform it using SQL-based or external processing engines, and analyze it alongside historical datasets. Worlikar et al. (2025) argue that this architectural flexibility allows Redshift to function not merely as a passive repository but as an active participant in real-time analytical workflows.

Despite these advances, the academic literature on data warehousing, stream processing, and real-time systems has developed along largely separate trajectories. Research on big data streams has focused on scalability, fault tolerance, and programming models (Kolajo et al., 2019; Nasiri, Nahesi, & Goudarzi, 2019), while real-time systems theory has concentrated on scheduling, fairness, and quality of service in multiprocessor environments (Baruah et al., 1996; Anderson & Devi, 2006). At the same time, applied domains such as cybersecurity and healthcare have generated their own bodies of work on real-time analytics, often without explicit reference to the underlying scheduling and warehousing architectures that make such analytics possible (Buczak & Guven, 2016; Alam et al., 2024). The result is a fragmented intellectual landscape in which critical dependencies between layers of the data stack remain under-theorized.

The need for integration is particularly evident in domains that demand both high throughput and strict latency guarantees. In cybersecurity intrusion detection, for example, network traffic must be analyzed in near real time to identify and mitigate attacks, yet the resulting alerts and historical patterns must also be stored and queried for forensic and strategic purposes (Aldarwbi, Lashkari, & Ghorbani, 2022; Buczak & Guven, 2016). Similarly, in healthcare analytics, continuous patient monitoring generates streams of physiological data that must be processed quickly to support clinical decisions while also being archived for longitudinal analysis and research (Alam et al., 2024). In both cases, failures of scheduling, resource allocation, or data integration can have severe consequences, ranging from security breaches to patient harm.

From a theoretical standpoint, these challenges call for a synthesis of data stream management and real-time scheduling theory. Data stream systems rely on operators that process incoming tuples according to specific semantics, and the scheduling of these operators across distributed resources determines both throughput and latency (Babcock et al., 2004). Real-time scheduling research, meanwhile, has developed sophisticated models for allocating processor time to tasks with deadlines and fairness constraints (Baruah et al., 1996; Block et al., 2008). Yet few studies have examined how these scheduling principles apply when the "tasks" in question are data stream operators feeding a cloud-native data warehouse that must support both ad hoc queries and continuous analytics.

The literature gap, therefore, lies not in the absence of technological solutions but in the lack of an integrated theoretical framework that explains how stream processing, scheduling, and cloud-native warehousing interact to produce reliable, scalable, and intelligent analytical systems. While comparative studies of frameworks such as Storm, Spark, and Samza provide valuable insights into performance characteristics (Amakobe, 2016; Behera et al., 2017), they often abstract away the downstream warehousing layer where analytical value is ultimately realized. Conversely, data warehousing research tends to focus on storage, query optimization, and data modeling without fully engaging with the dynamics of real-time data ingestion and scheduling (Worlikar et al., 2025).

This article addresses this gap by constructing a comprehensive, interdisciplinary analysis grounded in the provided references. It advances the argument that cloud-native data warehouses should be understood as real-time systems in their own right, subject to the same scheduling, fairness, and quality-of-service constraints that govern embedded and distributed real-time computing. By situating Amazon Redshift within a broader ecosystem of stream processing frameworks and real-time scheduling theories, the study offers a new conceptual lens through which to evaluate the performance and reliability of modern analytical infrastructures.

The stakes of this inquiry are not merely technical. As organizations increasingly rely on real-time analytics for strategic and operational decisions, the architectures that support these analytics shape what can be known, how quickly it can be known, and how confidently it can be acted upon. Understanding the theoretical foundations of these architectures is therefore essential for both scholars and practitioners seeking to design systems that are not only fast and scalable but also fair, reliable, and epistemically robust (Delen et al., 2018; Worlikar et al., 2025).

METHODOLOGY

The methodological orientation of this study is grounded in qualitative, theory-driven synthesis rather than empirical experimentation. This choice reflects both the nature of the provided references and the conceptual goals of the research. The literature spans multiple domains, including cloud-native data warehousing, real-time stream processing, multiprocessor scheduling, cybersecurity analytics, and healthcare data systems. Each of these domains has developed its own empirical benchmarks, experimental platforms, and evaluation criteria. Attempting to replicate or harmonize these heterogeneous empirical methodologies within a single study would not only be impractical but would also obscure the deeper theoretical relationships that the present research seeks to elucidate (Kolajo et al., 2019).

Instead, the methodology proceeds by systematically interpreting and integrating the conceptual, architectural, and empirical claims found in the references. The first step involves a thematic mapping of the literature, identifying recurring concepts such as latency, throughput, scalability, fairness, quality of service, and fault tolerance. These concepts appear across different research traditions but are often defined and operationalized in divergent ways. For example, in real-time scheduling theory, latency is typically framed in terms of deadline miss ratios and response times (Anderson & Devi, 2006; Åsberg et al., 2012), whereas in stream processing research it is often measured as end-to-end tuple processing delay (Babcock et al., 2004; Nasiri et al., 2019). By aligning these conceptualizations, the methodology establishes a common analytical vocabulary.

The second step involves architectural analysis. Cloud-native data warehouses such as Amazon Redshift are treated as central nodes in a larger analytical ecosystem that includes data sources, streaming platforms, processing engines, and end-user applications. Worlikar et al. (2025) provide a detailed account of how Redshift integrates with external data sources, supports scalable query processing, and manages storage and compute resources. This architectural description is juxtaposed with the architectures of stream processing frameworks such as Kafka, Storm, Spark, Flink, and Samza as described by Kleppmann and Kreps (2015), Katsifodimos and Schelter (2016), and Behera et al. (2017). The goal is not to compare platforms in a narrow benchmarking sense but to understand how their design principles complement or conflict with the requirements of cloud-native warehousing.

The third step draws on real-time scheduling theory to interpret how computational resources are allocated across distributed analytics pipelines. Classical results on proportionate progress and fairness in resource allocation (Baruah et al., 1996) are used to frame the challenges of multi-tenant cloud environments, where multiple streaming and analytical workloads compete for shared resources. Adaptive scheduling frameworks for multiprocessor real-time systems (Block et al., 2008) and external CPU schedulers (Åsberg et al., 2012) provide conceptual tools for understanding how cloud platforms might dynamically adjust resource allocation in response to changing workloads.

The fourth step situates these architectural and scheduling considerations within applied domains such as cybersecurity and healthcare. Studies on intrusion detection systems (Aldarwbi et al., 2022; Buczak & Guven, 2016) and chronic disease management through data analytics (Alam et al., 2024) are interpreted as case studies of real-time analytics in action. These domains are particularly useful because they impose stringent requirements on latency, reliability, and data integration, thereby stress-testing the theoretical framework developed in the study.

Throughout this methodological process, the analysis remains grounded in the provided references, ensuring that all claims are traceable to the existing literature. The approach acknowledges the limitations of a purely theoretical synthesis, particularly the absence of new empirical data, but argues that such synthesis is a necessary precursor to more targeted experimental research. By clarifying the conceptual relationships between stream processing, scheduling, and cloud-native warehousing, the methodology lays the groundwork for future studies that can empirically evaluate specific architectural and scheduling strategies in real-world deployments (Worlikar et al., 2025; Delen et al., 2018).

RESULTS

The integrative analysis of the provided literature yields several interrelated findings that illuminate the nature of modern real-time data warehousing. First, it becomes evident that cloud-native data warehouses such as Amazon Redshift function not merely as passive storage and query engines but as active components of real-time analytical pipelines. Worlikar et al. (2025) emphasize that Redshift's ability to ingest streaming data, perform continuous transformations, and support concurrent analytical queries positions it as a real-time system in its own right. This challenges traditional distinctions between operational data stores, stream processors, and analytical warehouses, suggesting instead a continuum of real-time data handling capabilities.

Second, the performance and reliability of such warehouses are deeply intertwined with the properties of the upstream stream processing frameworks. Kafka's role as a distributed, fault-tolerant log provides durable, ordered streams that can be consumed by multiple processing engines and warehousing systems (Kleppmann & Kreps, 2015). Storm and Spark Streaming offer different trade-offs between latency and throughput, with Storm emphasizing low-latency tuple processing and Spark providing micro-batching semantics that favor high throughput (Amakobe, 2016; Behera et al., 2017). Flink's stateful stream processing model further blurs the line between streaming and batch analytics (Katsifodimos & Schelter, 2016). When these frameworks feed data into a cloud-native warehouse, their scheduling and state management decisions directly affect query freshness, data consistency, and system responsiveness.

Third, real-time scheduling theory provides a powerful lens for understanding these interactions. In multiprocessor real-time systems, tasks with different priorities and deadlines must be scheduled in a way that ensures fairness and timeliness (Anderson & Devi, 2006). Analogously, in a cloud-native analytics environment, streaming operators, ingestion tasks, and analytical queries compete for shared compute and storage resources. Proportionate progress, as defined by Baruah et al. (1996), suggests that each workload should receive a fair share of resources relative to its importance, a principle that can be translated into multi-tenant data warehouse scheduling. Block et al. (2008) further show that adaptive scheduling frameworks can dynamically reallocate resources in response to workload changes, a capability that is increasingly relevant in elastic cloud environments (Worlikar et al., 2025).

Fourth, applied domains demonstrate the practical implications of these theoretical relationships. In cybersecurity, intrusion detection systems rely on continuous monitoring of network traffic and rapid identification of anomalies (Aldarwbi et al., 2022). Stream processing frameworks perform initial feature extraction and anomaly detection, while cloud-native warehouses store alerts, logs, and historical data for further analysis and reporting (Buczak & Guven, 2016). Any delay or unfairness in scheduling can lead to missed or late detections, undermining system effectiveness. Similarly, in healthcare, patient monitoring systems generate streams of vital signs that must be processed in real time to support clinical decisions, yet also integrated into long-term records and analytical models (Alam et al., 2024). The warehouse thus becomes a nexus where real-time and historical analytics converge, and its scheduling policies influence both immediate and longitudinal outcomes.

Fifth, the literature on benchmarking and scalability highlights that performance is not a fixed attribute of any single platform but an emergent property of the entire pipeline. Henning and Hasselbring (2024) show that the scalability of stream processing frameworks deployed as microservices depends on both the framework's internal architecture and the surrounding cloud infrastructure. When these frameworks are integrated with a warehouse like Redshift, the overall system's scalability reflects the interaction between microservice orchestration, network throughput, storage performance, and scheduling policies (Worlikar et al., 2025; Nasiri et al., 2019). This reinforces the conclusion that isolated benchmarks of individual components are insufficient to capture the true behavior of real-time analytical systems.

Taken together, these results suggest that modern cloud-native data warehousing is best understood as a form of distributed real-time computing. Its performance, fairness, and reliability depend on the alignment of stream processing semantics, scheduling algorithms, and storage-compute architectures. This integrated perspective challenges conventional wisdom that treats warehousing, streaming, and scheduling as separable concerns and instead positions them as mutually constitutive elements of intelligent data systems (Delen et al., 2018; Worlikar et al., 2025).

DISCUSSION

The findings of this study invite a reconsideration of how scholars and practitioners conceptualize real-time data analytics. Traditional narratives often depict a linear pipeline in which data flows from sources to stream processors and then into a warehouse for analysis. While this representation captures the basic movement of data, it obscures the complex feedback loops, scheduling dynamics, and resource competitions that shape system behavior. By integrating insights from real-time scheduling theory, the present analysis reveals that cloud-native data warehouses are not end points but active participants in a continuously negotiated process of computation and storage (Baruah et al., 1996; Worlikar et al., 2025).

One of the most significant theoretical implications concerns the notion of fairness. In multiprocessor real-time systems, fairness ensures that no task is starved of resources, even when higher-priority tasks dominate the schedule (Anderson & Devi, 2006). In cloud-native data warehouses, fairness translates into the equitable allocation of compute and I/O resources among concurrent queries, ingestion tasks, and transformation jobs. When fairness is violated, certain analytical workloads may experience unacceptable delays, leading to skewed or outdated insights. This is particularly problematic in multi-tenant environments where different organizational units or applications rely on the same warehouse (Worlikar et al., 2025). The application of proportionate progress principles to such environments suggests that scheduling policies should be explicitly designed to reflect organizational priorities and service-level agreements rather than relying on ad hoc resource allocation.

Another critical issue is the trade-off between latency and throughput. Stream processing frameworks embody different points along this trade-off spectrum, and their integration with a warehouse amplifies these differences. Storm's low-latency processing may be ideal for intrusion detection, where milliseconds matter (Aldarwbi et al., 2022), but it may generate a higher overhead for warehousing due to frequent, small writes. Spark's micro-batching, by contrast, may improve throughput and reduce write amplification but at the cost of higher end-to-end latency (Amakobe, 2016; Behera et al., 2017). The warehouse must therefore be designed and configured with an awareness of these upstream semantics, a point emphasized by Worlikar et al. (2025) in their discussion of Redshift's integration with streaming data sources.

Scalability is often portrayed as an inherent advantage of cloud-native architectures, yet the literature cautions that scalability is contingent on effective coordination across layers. Microservice-based deployments of stream processing frameworks can scale horizontally, but only if scheduling, networking, and storage subsystems keep pace (Henning & Hasselbring, 2024). When these frameworks feed into a centralized or semi-centralized warehouse, bottlenecks can emerge at the ingestion or query processing stages, undermining the benefits of distributed processing (Nasiri et al., 2019). Real-time scheduling theory suggests that adaptive, feedback-driven resource allocation is essential for maintaining scalability under dynamic workloads (Block et al., 2008), a principle that cloud-native warehouses are only beginning to implement in practice (Worlikar et al., 2025).

The applied domains of cybersecurity and healthcare further underscore the ethical and practical stakes of these architectural choices. In cybersecurity, delayed or inaccurate analytics can expose organizations to significant risk, while in healthcare, they can affect patient outcomes (Buczak & Guven, 2016; Alam et al., 2024). These domains therefore demand not only high performance but also predictable and explainable system behavior. Real-time scheduling theory provides a vocabulary for articulating such predictability, through concepts such as worst-case response time and deadline guarantees (Anderson & Devi, 2006). Translating these concepts into the context of cloud-native warehousing could enable more rigorous service-level agreements and more transparent system design.

Despite these insights, the present study also faces limitations. Its reliance on existing literature means that it cannot account for the latest proprietary innovations in cloud platforms, which may not yet be reflected in academic publications. Moreover, the qualitative nature of the synthesis precludes precise quantification of performance trade-offs. Nevertheless, the theoretical integration achieved here offers a valuable framework for interpreting both current and future developments in real-time data warehousing (Kolajo et al., 2019; Worlikar et al., 2025).

Future research should build on this framework by conducting empirical studies that explicitly measure the impact of scheduling policies and stream processing semantics on warehouse performance. Experimental platforms that integrate Kafka, Flink, and Redshift-like warehouses could be used to evaluate how different resource allocation strategies affect latency, throughput, and fairness under realistic workloads. Such studies would not only validate the theoretical claims advanced here but also provide practical guidance for system designers seeking to optimize real-time analytics infrastructures (Delen et al., 2018; Nasiri et al., 2019).

CONCLUSION

This article has argued that cloud-native real-time data warehousing represents a convergence of stream processing, distributed scheduling, and analytical storage that demands an integrated theoretical understanding. By synthesizing literature from big data streams, real-time systems, and applied analytics, and by grounding the analysis in the architectural insights of Amazon Redshift (Worlikar et al., 2025), the study has shown that the performance and reliability of modern analytical systems are emergent properties of multi-layered interactions rather than isolated platform features. Recognizing these interactions is essential for designing data infrastructures that can support the increasingly complex and time-sensitive decision-making needs of contemporary organizations.

REFERENCES

1. Gurusamy, V., Kannan, S., & Nandhini, K. (2017). The real time big data processing framework advantages and limitations. International Journal of Computer Sciences and Engineering, 5(12), 305–312.
2. Worlikar, S., Patel, H., & Challa, A. (2025). Amazon Redshift Cookbook: Recipes for building modern data warehousing solutions. Packt Publishing Ltd.
3. Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–1176.
4. Kleppmann, M., & Kreps, J. (2015). Kafka, Samza and the Unix philosophy of distributed data. IEEE Data Engineering Bulletin, 38(4), 4–14.
5. Alam, M. A., Sohel, A., Uddin, M. M., & Siddiki, A. (2024). Big data and chronic disease management through patient monitoring and treatment with data analytics. Academic Journal on Artificial Intelligence, Machine Learning, Data Science and Management Information Systems, 1(01), 77–94.
6. Baruah, S., Cohen, N. K., Plaxton, C. G., & Varvel, D. A. (1996). Proportionate progress: A notion of fairness in resource allocation. Algorithmica, 15(6), 600–625.
7. Amakobe, M. (2016). A comparison between Apache Samza and Storm. Colorado Tech University.
8. Henning, S., & Hasselbring, W. (2024). Benchmarking scalability of stream processing frameworks deployed as microservices in the cloud. Journal of Systems and Software, 208.
9. Behera, R. K., Das, S., Jena, M., Rath, S. K., & Sahoo, B. (2017). A comparative study of distributed tools for analyzing streaming data. International Conference on Information Technology, 79–84.
10. Nasiri, H., Nahesi, S., & Goudarzi, M. (2019). Evaluation of distributed stream processing frameworks for IoT applications in smart cities. Journal of Big Data, 6.
11. Aldarwbi, M. Y., Lashkari, A. H., & Ghorbani, A. A. (2022). The sound of intrusion: A novel network intrusion detection system. Computers and Electrical Engineering, 104, 108455.
12. Kolajo, T., Daramola, O., & Adebiyi, A. (2019). Big data stream analysis: A systematic literature review. Journal of Big Data, 6.
13. Delen, D., Moscato, G., et al. (2018). The impact of real-time business intelligence and advanced analytics on the behavior of business decision-makers. International Conference on Information Management and Processing.
14. Katsifodimos, A., & Schelter, S. (2016). Apache Flink: Stream analytics at scale. IEEE International Conference on Cloud Engineering Workshop, 193.

15. Anderson, J. H., & Devi, U. C. (2006). Soft real-time scheduling on multiprocessors.
16. Block, A., Brandenburg, B. B., Anderson, J. H., & Quint, S. (2008). An adaptive framework for multiprocessor real-time systems. Euromicro Conference on Real-Time Systems, 23–33.
17. Babcock, B., Babu, S., Datar, M., Motwani, R., & Thomas, D. (2004). Operator scheduling in data stream systems. The VLDB Journal, 13(4), 333–353.