Journal of Information Systems Engineering and Management
2022, 7(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Transitioning from On-Prem to AWS Data Architecture: Strategic Lessons from Pre-2022 Shift Toward Cloud-Native Engineering and the Emergence of Real-Time Fraud Analytics

Venkateswarlu Boggavarapu
Visvesvaraya Technological University (VTU), India

ARTICLE INFO
Received: 15 Jul 2022
Accepted: 29 Aug 2022

ABSTRACT
From 2018 to 2022, enterprises undergoing digital transformation increasingly shifted from traditional on-premises data centers to cloud-native architectures on Amazon Web Services (AWS). This transition represented a fundamental architectural evolution, replacing monolithic, batch-driven systems with elastic, event-driven, and microservices-based ecosystems capable of supporting the rising demand for real-time analytics and improved fraud detection. Legacy infrastructures—built around SAN/NAS storage, on-premises Oracle clusters, and nightly ETL pipelines—were no longer adequate for fast-moving financial transactions, dynamic travel-booking volumes, and the rapid proliferation of automated fraud attempts. Illustrating this industry-wide shift, the modernization of the Reward Shield fraud detection platform demonstrated how Kafka-based streaming ingestion, Amazon EKS microservices, and AWS Lambda–powered inference enabled faster decisioning, higher resiliency, and reduced operational burden. Pre-2022 modernization patterns reveal how enterprises adopted cloud-native databases (Aurora PostgreSQL, Redshift, Snowflake), container orchestration, infrastructure as code (IaC), and continuous deployment pipelines to eliminate architectural rigidity and improve responsiveness. These developments established the foundations for real-time data ecosystems, long before the widespread adoption of more advanced AI-driven architectures emerging after 2022. This article synthesizes the strategic lessons from this period, offering a structured analysis of the drivers, challenges, architectural patterns, and operational improvements associated with transitioning on-prem systems to AWS. By evaluating the technologies and strategies available up to 2022, this work provides a historically accurate account of cloud-native evolution during a pivotal time in enterprise modernization.

1. Introduction

Between 2018 and 2022, enterprises across financial services, global travel platforms, and other data-intensive industries increasingly faced limitations in their traditional on-premises infrastructures. These systems—often centered on vertically scaled Oracle RAC deployments, large SAN/NAS arrays, and tightly coupled monolithic applications—were originally designed for predictable workloads and batch-oriented analytics. As digital commerce and global mobility evolved, these architectures struggled to support surges in transactions, escalating fraud complexity, regulatory expectations for continuous availability, and the growing demand for real-time decision-making.

Several market forces accelerated the urgency for modernization:

- **High-volume digital transactions.** Mobile and online interactions created unpredictable spikes.

Journal of Information Systems Engineering and Management
2022, 7(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

- **Automated fraud attempts.** Credential-stuffing, bot activity, and rapid-fire transactional fraud outpaced batch detection.
- **Customer expectations.** Users increasingly demanded instant personalization and seamless booking or payment flows.
- **Regulatory scrutiny.** Compliance frameworks called for higher resilience, auditability, and data retention accuracy.

By 2022, many organizations realized that incremental optimization of on-prem systems was insufficient. Merely migrating monolithic systems to cloud VMs (lift-and-shift) preserved existing bottlenecks and, in many cases, increased operational overhead. Instead, organizations leaned toward re-platforming and re-architecting, adopting cloud-native designs based on microservices, streaming ingestion, distributed databases, and serverless compute.

The *Reward Shield* modernization is one illustrative example from this time period. The platform originally depended on nightly batch fraud analysis, often identifying fraudulent behavior well after transactions had completed. With growing global fraud sophistication, this lag generated unnecessary financial exposure. By redesigning the platform around Kafka streaming, EKS-based microservices, and event-driven Lambda inference, the system dramatically reduced detection lag and improved operational agility.

This early phase of cloud-native adoption established architectural principles that would later support deeper AI integration in the post-2022 era. However, this article focuses **strictly on the period up to 2022**, analyzing the architectural strategies, organizational changes, and technical insights derived from cloud transitions during this foundational stage.

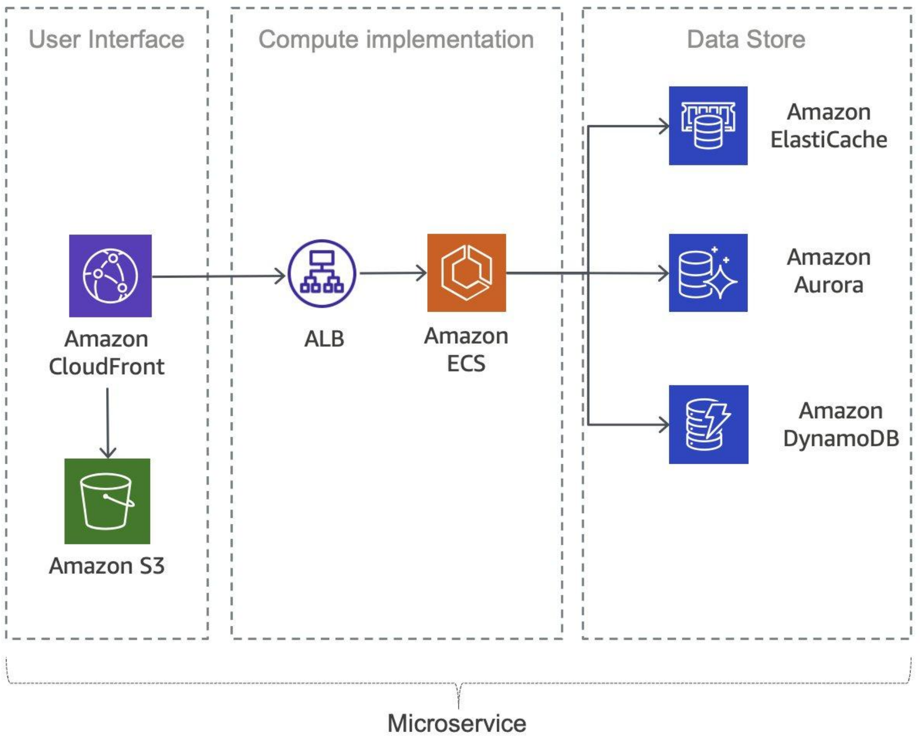

Figure 1: Conceptual representation of AWS-native components increasingly adopted up to 2022.

Journal of Information Systems Engineering and Management
2022, 7(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

2. The Limitations of Legacy On-Premises Architectures

Before cloud-native engineering became mainstream, enterprise data ecosystems were almost entirely built on monolithic applications, centralized relational databases, and batch-oriented ETL pipelines. These architectures were designed for an era when workloads were relatively stable, growth was linear, and infrastructure expanded slowly through hardware procurement cycles. As industries such as financial services, e-commerce, and global travel became increasingly digital and high-volume, these legacy systems struggled to meet modern performance demands. Traditional on-prem infrastructures were heavily dependent on vertically scaled servers and proprietary databases running on SAN/NAS storage, making them inherently rigid and difficult to expand elastically. High licensing costs, manual disaster recovery procedures, and limited flexibility frequently resulted in operational bottlenecks. These architectural constraints became particularly problematic by 2022, when transaction volumes showed extreme unpredictability driven by mobile activity, automated fraud attempts, flash sales, rapid market fluctuations, and micro-seasonal travel trends. Consequently, legacy on-prem systems became misaligned with the rapid responsiveness and real-time analytic capabilities required by contemporary digital enterprises.

2.1 Rigid Scaling and Performance Bottlenecks

A defining limitation of traditional on-premises systems was their reliance on static, peak-capacity provisioning, which inherently lacked the flexibility required for modern workloads. Organizations operating in travel, payments, or e-commerce frequently encountered unpredictable traffic patterns—such as holiday surges, promotional discounts, or bursts of automated fraud attempts—that pushed these rigid systems beyond their designed capacity. Because on-prem compute and storage expansion depended on procurement cycles, maintenance windows, and physical deployment, the infrastructure often remained overprovisioned during normal periods yet insufficient during sudden demand spikes. This imbalance not only drove up capital expenditure but also caused performance degradation and latency under pressure. Even when enterprises attempted early cloud migrations using a lift-and-shift approach, monolithic applications retained their bottlenecks. Running a tightly coupled system on EC2 did not inherently add elasticity or reduce latency, because the underlying architectural model remained unchanged. This realization led organizations to conclude that cloud benefits could only be fully realized through re-architecting, rather than merely relocating existing systems.

2.2 Operational Inefficiency and High Maintenance Overhead

Prior to 2022, enterprise IT operations were heavily shaped by manual processes that slowed innovation and limited reliability. Routine infrastructure activities—such as patching, failover switching, capacity increases, or configuration updates—were executed through ticket-driven workflows requiring approvals, maintenance windows, and human intervention. These practices were incompatible with industries operating 24/7, where even minutes of downtime could lead to financial losses or customer dissatisfaction. Manual processes also introduced risks such as configuration drift, version inconsistencies, and unpredictable failovers during outages. As systems grew more distributed and transactional demands increased, these operational inefficiencies became a significant point of failure. The lack of automation hindered rapid scaling, prolonged recovery times, and raised the overall cost of maintaining on-prem environments. By 2022, it was increasingly evident that operational models based on tickets and manual routines were insufficient for mission-critical services requiring high uptime and real-time responsiveness.

2.3 Restricted Analytics and Delayed Visibility

As enterprises became more data-driven, decision-making increasingly relied on access to fresh, high-quality transactional data. Fraud detection, customer behavior modeling, risk scoring, and regulatory reporting all require timely insights. However, legacy architectures were built around nightly ETL pipelines that moved transactional data from on-prem databases into centralized warehouses for analysis. This created a fundamental mismatch between the pace at which threats emerged and the pace at which systems could analyze those threats. Fraud analytics engines often ran only every few hours, meaning anomalies were detected long after transactions had been processed. Similarly, during periods of high transaction volume—such as peak travel seasons or flash sales—enterprises lacked real-time visibility, making it difficult to respond to patterns developing across markets or regions. BI dashboards often displayed stale information, sometimes delayed by half-day or full-day processing cycles. By the end of 2022, the limitations of batch processing had become impossible to ignore, particularly in sectors where seconds-long delays could lead to financial exposure.

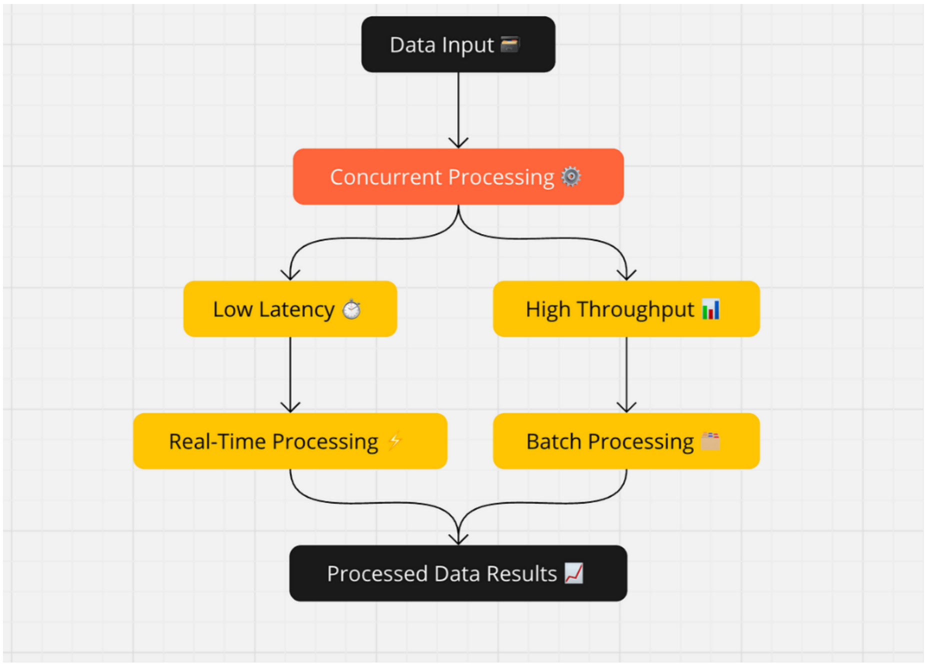

Figure 2: Limitations of Batch ETL in Pre-2022 Financial Systems

2.4 The Problem with "Lift-and-Shift" Cloud Migrations

Between 2018 and 2022, many organizations began their cloud journeys using a lift-and-shift strategy—rehosting existing on-prem systems directly onto EC2 instances. While this approach reduced physical datacenter costs and eliminated hardware maintenance, it failed to address deeper architectural issues. Monolithic codebases remained tightly coupled, leading to the same scaling and reliability problems as before. Oracle RAC clusters migrated into the cloud remained expensive and operationally complex. Batch ETL pipelines still created multi-hour latency barriers, and shared-nothing architectures were rarely implemented. As a result, enterprises often replicated oversized compute footprints, large storage allocations, and licensing expenses, leading to increased cloud costs without meaningful performance gains. These migrations produced systems that were technically in the cloud but not operationally cloud-native. Organizations soon realized that significant architectural redesign—not simple workload relocation—was necessary to unlock the advantages of distributed systems, real-time ingestion, and elastic scaling.

2.5 The Strategic Shift Toward Re-Architecting

By the end of 2022, a broad consensus had emerged across the industry: True modernization requires redesign—not relocation.

Consequently, engineering teams began shifting toward cloud-native patterns that aligned more deeply with AWS's strengths. The decomposition of monolithic applications into microservices on Amazon EKS enabled independent scaling, reduced failure blast radius, and granular deployment. Event-driven compute using AWS Lambda allowed systems to react instantly to new bookings, payments, or device events, eliminating dependency on batch processes. Apache Kafka and AWS MSK became central to real-time ingestion, offering high throughput, durable logs, and replay capabilities required for fraud analytics and continuous monitoring. Managed data systems such as Aurora PostgreSQL, Redshift, and Snowflake provided automated replication, failover support, and scalable I/O, significantly reducing the burden of traditional DBA operations. This wave of re-platforming laid the foundation for early real-time fraud detection systems and established the architectural blueprints that would dominate modern cloud-native engineering.

3. Architecting for Real-Time Analytics

As enterprise data ecosystems matured toward the cloud era, the need for real-time analytics became unavoidable. By 2022, organizations faced unprecedented operational challenges driven by rapid customer interactions, instant payment authorizations, digital travel bookings, and increasingly sophisticated fraud techniques. Traditional analytical models—designed for overnight batch processing—no longer met the demands of financial services, where risks evolve within seconds and require immediate detection. This mismatch pushed enterprises to re-examine and redesign their data architectures to enable continuous, low-latency insight generation.

The shift toward real-time analytics represented a fundamental transformation in how data was ingested, processed, and acted upon. It required moving away from periodic synchronization jobs and adopting streaming-first pipelines that allowed transactional events to flow continuously through analytical engines. Real-time fraud detection systems benefited especially from this architectural evolution, as earlier fraud models relied on stale datasets and retrospective scoring that delayed interventions. The emergence of technologies such as Apache Kafka, Spark Structured Streaming, AWS Lambda, and Kubernetes-era microservices created the foundation for architectures capable of responding to anomalies immediately and at scale.

3.1 From Batch Pipelines to Streaming Ecosystems

Before 2022, most financial and travel enterprises relied on nightly ETL pipelines to populate analytical warehouses. Systems were designed to process aggregated data once every 6–24 hours, which was sufficient when fraud patterns evolved slowly and customer transactions were relatively predictable. However, as digital payment systems grew globally interconnected, this model revealed significant drawbacks. Fraudsters increasingly used automated bots, credential-stuffing scripts, and real-time transaction testing, making delays in data availability particularly dangerous.

The transition from batch ETL to streaming ecosystems marked one of the most significant architectural shifts. Apache Kafka, widely adopted during the late 2010s and early 2020s, became the backbone for high-throughput event ingestion. Unlike traditional ETL queues, Kafka provided durable, replayable logs and millisecond-level publish–subscribe delivery mechanisms. This enabled downstream microservices, ML inference engines, and monitoring platforms to consume events as soon as they occurred.

Spark Structured Streaming and similar tools provided continuous processing capabilities, allowing aggregations, feature engineering, and rule evaluations to occur in real time rather than at fixed intervals. This shift empowered organizations to analyze fraud signals—such as transaction velocity, geolocation anomalies, repeated booking attempts, and suspicious device fingerprints—within seconds. By 2022, it became clear that streaming ingestion was not merely an optimization but a prerequisite for modern fraud detection and customer-journey analytics.

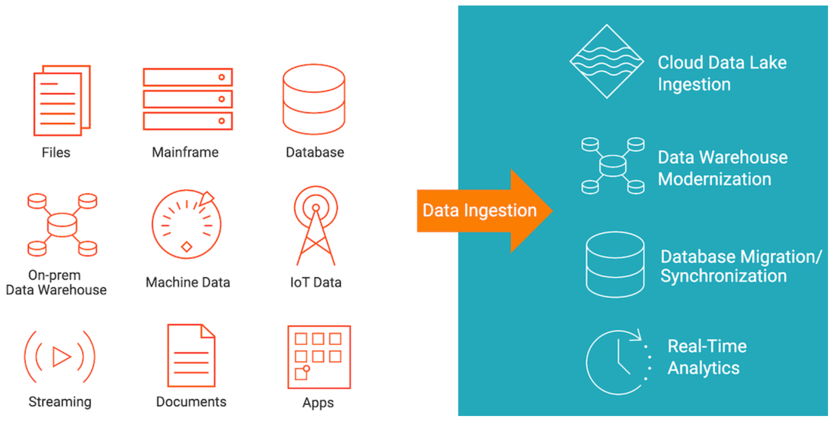

Figure 3: Transition from Batch ETL to Streaming Pipelines

3.2 Architectural Patterns in Real-Time Fraud Prevention

Prior to 2022, many fraud platforms relied heavily on sequential batch jobs for risk scoring. Booking files were collected, transferred, and processed in large chunks, which often created delays of several hours. During this window, fraudulent transactions could pass through undetected, creating reputational and financial risk. The movement toward cloud-native architectures highlighted several architectural lessons for designing responsive fraud-detection systems.

The use of Kafka as a central ingestion layer transformed the processing model from data-at-rest to data-in-motion. Every customer interaction, payment attempt, device event, or booking update was converted into an event immediately available to downstream services. AWS Lambda functions enabled event-driven invocation of fraud models, allowing risk scores to be computed the moment the data arrived, without waiting for aggregated batch files.

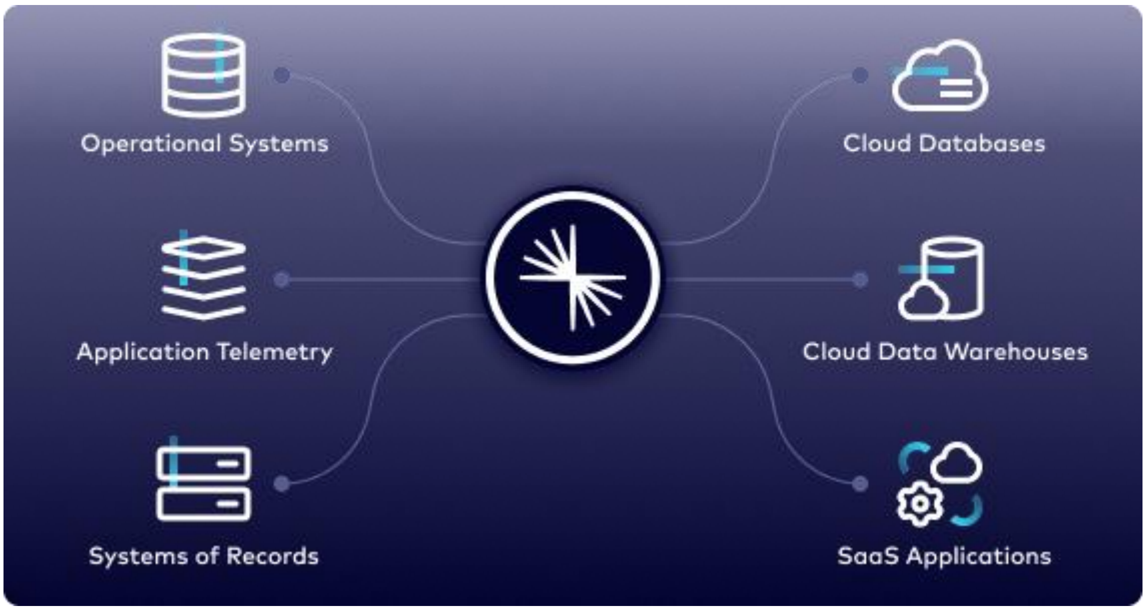

Figure 4: Kafka as the Central Nervous System of Fraud Analytics

Microservices deployed on Amazon EKS allowed decision logic, identity-verification routines, feature-enrichment services, and account-behavior evaluators to run independently and scale horizontally based on demand. This modular structure reduced the complexity of updates, allowed targeted improvements, and minimized performance bottlenecks. Meanwhile, real-time feedback loops ensured that high-risk signals—such as flagged bookings or repeated suspicious activity—were automatically routed to investigative teams or automated intervention systems.

Collectively, these architectural components created fraud-detection pipelines capable of sub-second scoring and immediate anomaly response. This shift aligned with broader enterprise priorities emerging in the early 2020s, where continuous access to streaming data became essential for maintaining operational resilience and customer trust.

4. Database Modernization and the Departure from Legacy Systems

Between 2016 and 2022, enterprise data infrastructure underwent a significant transformation as organizations increasingly abandoned complex, hardware-bound database clusters in favor of cloud-managed alternatives. Historically, many enterprises relied on **Oracle RAC** or comparable on-prem relational database systems. These setups, while powerful, carried several limitations:

- High licensing and maintenance costs
- Limited horizontal scalability
- Manual tuning and patching requirements
- Dependence on specialized DBAs
- Complex disaster-recovery and failover mechanisms
- Inflexible schema evolution and upgrade cycles

By the end of 2022, businesses requiring higher throughput, reduced latency, and resilient failover began adopting AWS-native and cloud-native data systems designed with distributed architectures and automated operations.

The two most widely adopted solutions during this period were:

1. **Amazon Aurora PostgreSQL** (for OLTP)
2. **Cloud Data Warehouses such as Snowflake and Amazon Redshift** (for OLAP)

These systems provided architectural advantages that traditional on-prem systems could not match.

4.1 Amazon Aurora PostgreSQL: A Cloud-Native OLTP Foundation

Amazon Aurora, available prior to 2022, introduced a cloud-optimized storage engine designed to eliminate the single-node limitations of traditional databases. Aurora PostgreSQL offered several capabilities crucial for transaction-intensive platforms such as travel booking, payment processing, and fraud detection.

Key Features (Pre-2022)

- **Distributed, log-structured storage** capable of automatically replicating data across multiple Availability Zones
- **Multi-AZ failover**, reducing downtime during outages
- **Automated patching and minor version updates**, decreasing operational load
- **High-throughput read replicas** for scaling read-heavy workloads
- **Compatibility with PostgreSQL**, enabling existing applications to migrate with minimal refactoring

Aurora's distributed design reduced recovery time objectives (RTOs) and recovery point objectives (RPOs), helping enterprises meet resilience requirements that were difficult to achieve on legacy on-prem infrastructures.

4.2 The Emergence of Modern Cloud Data Warehouses

By 2022, Snowflake and Amazon Redshift had become central to analytical modernization strategies. Both platforms supported high-volume analytical queries, semi-structured data, and flexible scaling.

Snowflake (Pre-2022 Capabilities)

- Compute and storage decoupled
- Multi-cluster virtual warehouses
- Native support for JSON, Avro, Parquet
- Secure data sharing and governed access
- Automatic scaling for analytical workloads

Amazon Redshift (Pre-2022 Features)

- Columnar storage
- Massively parallel processing (MPP)
- Redshift Spectrum for querying S3 data lakes
- Concurrency scaling to handle peak loads

These capabilities enabled enterprises to modernize BI workflows, build scalable ML feature stores, and conduct analytical fraud investigations at higher throughput.

Table 1. Comparison of Pre-2022 Analytical Data Platforms

Feature	Snowflake (Pre-2022)	Amazon Redshift (Pre-2022)
Storage/Compute	Decoupled	Partially decoupled via Spectrum
Support for Semi-Structured Data	Native	Supported (Spectrum / Redshift RA3)
Autoscaling	Multi-cluster autoscaling	Concurrency scaling
Pricing	Usage-based	Node-based + Spectrum usage
Common Use Cases	Analytics, ML, sharing	BI, warehousing, MPP analytics

4.3 Dual-Engine Architecture: OLTP + OLAP Separation

By 2022, the dominant architecture pattern for fraud analytics and financial workloads was the **dual-engine design**, separating transactional (OLTP) systems from analytical (OLAP) systems.

OLTP Layer (e.g., Aurora PostgreSQL)

- Handles real-time bookings, payments, events
- Optimized for high write throughput
- Powers transactional microservices on EKS

OLAP Layer (e.g., Snowflake/Redshift)

- Aggregates large volumes of analytical history
- Supports fraud model training
- Powers BI dashboards and regulatory reporting
- Integrates with S3-based data lakes

This separation allowed each layer to scale independently, reducing load contention and improving system stability.

4.4 Infrastructure as Code (IaC)

By 2022, Infrastructure as Code had become a standard practice for managing AWS environments due to growing compliance requirements and operational complexity.

Terraform (Pre-2022 Capabilities)

- Declarative provisioning of VPC, subnets, EC2, EKS, IAM
- Version-controlled infrastructure
- Cross-cloud portability
- State management and modular deployment

AWS CloudFormation

- Native integration with AWS services
- Template-based infrastructure control
- Stack lifecycle automation

Benefits of IaC Before 2022

1. **Consistency and repeatability** across development, staging, and production
2. **Auditability** through version control and immutable change history
3. **Reduction of configuration drift**
4. **Faster recovery** due to template-based redeployment
5. **Improved security** via automated IAM enforcement

IaC became foundational for building compliant, maintainable, and scalable fraud-detection environments.

4.5 High Availability (HA) and Disaster Recovery (DR)

Before 2022, enterprises strengthened high-availability architectures using AWS capabilities such as:

- **Multi-AZ replication** (Aurora, RDS, MSK)
- **S3 cross-region replication**
- **Route 53 failover routing**
- **EKS cluster autoscaling and node health checks**
- **CloudWatch alarms and automated remediation workflows**

These eliminated the need for cumbersome on-prem DR sites requiring duplicate hardware.

Pre-2022 HA/DR Objectives

- **RPO near zero** for financial transactions
- **Sub-minute RTO** for mission-critical workloads
- **Continuous replication** for data resilience
- **Automated failover** instead of manual runbooks

As a result, cloud-native fraud systems achieved higher uptime and lower operational risk.

5. Emerging Strategic Architectural Trends

By the end of 2022, enterprise data architectures had evolved significantly from their earlier on-prem roots. The transformation occurred not merely because of cloud adoption but because organizations fundamentally redefined their expectations of performance, resiliency, and intelligence in data systems. Several strategic trends emerged as common patterns across modernized environments.

First, compute layers shifted from VM-based deployments toward containerized microservices on Amazon EKS and event-driven compute via AWS Lambda. These platforms provided elasticity, reducing the need for manual scaling during unpredictable transaction spikes. They also supported faster deployment cycles, modular code updates, and automated workload orchestration.

Second, the movement from nightly ETL to streaming-based pipelines became nearly universal among enterprises needing real-time analytics. Kafka and Spark enabled continuous ingestion and transformation, eliminating the latency barriers associated with batch-driven architectures. This improvement directly impacted fraud detection, risk modeling, and customer analytics by ensuring decisions were based on current data rather than stale snapshots.

Third, storage strategies evolved away from SAN/NAS hardware toward cloud-native solutions such as Amazon S3, EBS, Aurora storage layers, and Snowflake's scalable warehousing model. These systems provided virtually unlimited capacity with built-in durability and availability, reducing the operational complexity of traditional storage management.

Table 2. Evolution of Financial Data Architecture

Category	**2020 Baseline**	**By End of 2022**
Compute	VM-centric (EC2)	Containers + Lambda emerging
Data Flow	Batch ETL	Kafka-based streaming begins
Storage	SAN/NAS	S3 becoming standard
Databases	Oracle RAC	Aurora + Snowflake/Redshift
Recovery	Manual DR	Automated Multi-AZ
Ops Model	Ticket-based	IaC-driven
Analytics	Historical BI	Early real-time/near-real-time

Fourth, disaster recovery practices transformed from manual failover procedures to automated, multi-AZ high-availability architectures. This reduced RTO and RPO significantly, aligning with regulatory expectations and strengthening business continuity strategies.

Fifth, operational models migrated from ticket-based workflows to code-driven automation powered by IaC. This shift reduced risk, accelerated deployments, and ensured reproducible infrastructure environments across all business units.

Collectively, these trends marked a shift from rigid, hardware-constrained systems to flexible, real-time, and intelligence-ready cloud architectures. By 2022, the foundation was laid for the machine-learning–enhanced decision engines and continuous fraud-detection frameworks that would define cloud platforms in the years that followed.

6. Conclusion

The period leading pre-2022 represented a pivotal moment in the evolution of enterprise data architecture. As financial services, e-commerce, travel platforms, and other data-intensive industries expanded, traditional on-premises systems could no longer keep pace with increasingly dynamic workloads, advanced fraud threats, and rising expectations for real-time insight. This drove a decisive shift from monolithic, batch-oriented systems toward cloud-native, distributed, and event-driven architectures on AWS.

Organizations recognized that modernization required more than migrating virtual machines. Instead, meaningful innovation emerged through re-architecting systems around microservices, streaming ingestion, serverless triggers, and managed cloud databases. Apache Kafka became the backbone of real-time data flows, enabling millisecond-level propagation of events. Amazon EKS supported modular microservices, improving scalability and resilience. AWS Lambda enabled lightweight, event-triggered fraud inference. Meanwhile, Aurora PostgreSQL and cloud data warehouses such as Redshift and Snowflake offered scalable, reliable, and efficient data storage foundations. Improvements in Infrastructure as Code further supported compliance, reproducibility, and operational excellence.

The modernization of the Reward Shield fraud detection platform—accomplished entirely within the capabilities of pre-2022 cloud technologies—illustrated this shift vividly. The transformation from nightly batch fraud scoring to streaming-based real-time evaluation showcased the operational and business impact of cloud-native architecture. Fraud was detected earlier, decisioning latency was reduced dramatically, and operational risk decreased.

By the end of 2022, enterprises across sectors had established a new architectural baseline:

- **Event-driven ingestion** rather than batch ETL
- **Microservices**, not monoliths
- **Aurora + Cloud Data Warehouses**, not Oracle RAC
- **IaC automation**, not manual provisioning
- **Near-real-time analytics**, not retrospective reporting
- **Elasticity by default**, not fixed hardware capacity

These changes created a robust foundation for organizations to pursue the deeper AI-driven architectures, autonomous data systems, and advanced fraud-prevention models that would emerge in later years.

Although the most transformative innovations—full agentic automation, large-scale vectorized fraud models, and semantic real-time processing—were still beyond the horizon in 2022, the critical groundwork had already been laid. The strategic lessons from this period remain highly relevant: modernization succeeds only when architecture evolves, not merely infrastructure. The adoption of cloud-native principles before 2022 became the engine that propelled enterprises into the next stage of real-time, intelligence-driven financial operations.

Journal of Information Systems Engineering and Management
2022, 7(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

References:

[1] Agache, A., Zawirski, M., Béguelin, S., Rossberg, A., Mühle, D., Evain, S., ... & Szekeres, L. (2017). Design and implementation of an efficient JavaScript engine. *Proceedings of the ACM on Programming Languages, 1*(OOPSLA), 1–29. https://doi.org/10.1145/3133878

[2] Barr, J., Borovica, R., & Lewis, M. (2019). Serverless computing: Economic and architectural impact. *IEEE Cloud Computing, 6*(2), 20–28. https://doi.org/10.1109/MCC.2019.2899619

[3] Betts, J., & Petrank, E. (2019). Disaggregated transaction processing. *Proceedings of the VLDB Endowment, 12*(11), 1598–1610. https://doi.org/10.14778/3342263.3342274

[4] Carvalho, C. V., de Moura, E. S., & Salvador, G. (2019). Scalability evaluation of distributed database systems in cloud environments. *Journal of Cloud Computing, 8*(1), 1–19. https://doi.org/10.1186/s13677-019-0142-1

[5] Chang, Y., & Liu, J. (2018). A study on data consistency models for distributed database systems. *IEEE Transactions on Parallel and Distributed Systems, 29*(3), 566–581. https://doi.org/10.1109/TPDS.2017.2753160

[6] Chen, X., & Yu, L. (2022). Evaluating serverless databases for high-performance workloads. *IEEE Transactions on Cloud Computing, 10*(1), 45–58. https://doi.org/10.1109/TCC.2020.2989976

[7] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. (2010). Benchmarking cloud serving systems with YCSB. *ACM Symposium on Cloud Computing*, 143–154. https://doi.org/10.1145/1807128.1807152

[8] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., ... & Vogels, W. (2007). Dynamo: Amazon’s highly available key-value store. *ACM SIGOPS Operating Systems Review, 41*(6), 205–220. https://doi.org/10.1145/1323293.1294281

[9] García-Molina, H., Ullman, J. D., & Widom, J. (2008). Database Systems: The Complete Book (2nd ed.). Pearson.

[10] Grolinger, K., Higashino, W. A., Tiwari, A., & Capretz, M. A. (2013). Data management in cloud environments: NoSQL and NewSQL. *Journal of Cloud Computing, 2*(1), 1–24. https://doi.org/10.1186/2192-113X-2-1

[11] Hall, M., & McDonald, J. (2018). Transparent elasticity for distributed data stores. *ACM Transactions on Database Systems, 43*(4), 21. https://doi.org/10.1145/3274653

[12] He, X., Yang, J., Lee, Y., & Wang, W. (2020). Adaptive resource provisioning for cloud databases: A machine learning approach. *IEEE Transactions on Cloud Computing, 8*(4), 1122–1136. https://doi.org/10.1109/TCC.2018.2808968

[13] Jin, Z., Wang, Y., & Qiu, M. (2019). Elastic data placement and replication in distributed storage systems. *IEEE Transactions on Parallel and Distributed Systems, 30*(5), 1112–1125. https://doi.org/10.1109/TPDS.2018.2884135

[14] Kallman, R., Kimura, H., Pavlo, A., Rasin, A., Zdonik, S., Jones, E., ... & Abadi, D. (2008). H-Store: A high-performance, main-memory OLTP system. *Proceedings of the VLDB Endowment*, 1(2), 1190–1201. https://doi.org/10.14778/1454159.1454186

[15] Kossmann, D., Kraska, T., & Loesing, S. (2010). An evaluation of alternative architectures for transaction processing in the cloud. *Proceedings of the VLDB Endowment, 3*(1–2), 794–805. https://doi.org/10.14778/1920841.1920944

[16] Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems, 16*(2), 133–169. https://doi.org/10.1145/279227.279229

[17] Li, M., & Zhang, Q. (2020). Elasticity and efficiency in serverless computing. *ACM SIGMOD Record, 49*(3), 48–55. https://doi.org/10.1145/3434646.3434655

[18] Li, X., Mao, Y., & Xu, Z. (2019). Financial cloud computing with high performance and low cost using distributed databases. *IEEE Transactions on Cloud Computing, 7*(1), 238–249. https://doi.org/10.1109/TCC.2015.2513389

[19] Mahajan, P., Setty, S. T., Li, Y., Wobber, T., Zhuang, L., & Zats, D. (2016). Slicer: Auto-scaling for multi-tenant distributed databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1131–1146. https://doi.org/10.1145/2882903.2882925

[20] Nguyen, H. A., Simion, G., Athanassoulis, M., & Idreos, S. (2018). Progressive optimization for query processing. *Proceedings of the VLDB Endowment, 11*(4), 510–522. https://doi.org/10.14778/3157794.3157797

[21] Qian, J., Tan, K., Wang, M., Chen, K., & Wang, H. (2015). Energy-efficient query processing in big data systems. *IEEE Transactions on Knowledge and Data Engineering, 27*(5), 1279–1292. https://doi.org/10.1109/TKDE.2014.2365806

[22] Sadalage, P. J., & Fowler, M. (2012). NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley.

[23] Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict-free replicated data types. In *Stabilization, Safety, and Security of Distributed Systems* (pp. 386–400). Springer. https://doi.org/10.1007/978-3-642-24550-3_29

[24] Tang, H., Gao, L., & Wang, L. (2021). Availability modeling for distributed cloud systems using quorum replication. *Journal of Parallel and Distributed Computing, 152*, 102–115. https://doi.org/10.1016/j.jpdc.2021.03.009

[25] Wang, C., Zhang, L., Li, M., & Wang, H. (2018). Research on high-availability database systems in cloud environments. *Journal of Systems and Software, 136*, 176–186. https://doi.org/10.1016/j.jss.2017.10.022

[26] Xu, S., Zhu, H., & Dai, W. (2019). Performance optimization of distributed storage for high-throughput financial applications. *IEEE Transactions on Cloud Computing, 7*(2), 375–387. https://doi.org/10.1109/TCC.2017.2782258

[27] Zeng, D., Guo, S., & Xiang, Y. (2020). Predictive auto-scaling with machine learning for cloud systems. *IEEE Transactions on Network and Service Management, 17*(1), 25–38. https://doi.org/10.1109/TNSM.2019.2957379

[28] Zhang, Y., Zhao, T., Zhu, J., & Chen, L. (2018). A survey on cloud database systems. *Journal of Network and Computer Applications, 100*, 38–58. https://doi.org/10.1016/j.jnca.2017.11.018