Frontiers in Computer Science and Artificial Intelligence
ISSN: 2978-8048
DOI: 10.32996/fcsai
Journal Homepage: www.al-kindipublisher.com/index.php/fcsai

FCSAI
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Authoritative Data Stores for Treasury Deposits: Sourcing, Certification, and Provisioning Standards for Enterprise Downstreams

Ravi Kumar Vallemoni
Senior Data Architect, USA
Corresponding Author: Ravi Kumar Vallemoni**, E-mail**: vallemoni.ravikumar@gmail.com

| ABSTRACT

The rapid growth of enterprise data volumes and the increasing demand for timely, accurate, and auditable financial reporting have compelled large organizations to rethink traditional extract–transform–load (ETL) architectures supporting treasury deposit systems. Between 2018 and 2019, enterprises—particularly in banking and financial services—faced mounting limitations with legacy SAS, mainframe COBOL, Informatica, and Oracle PL/SQL-based batch processing platforms. These systems were costly, vertically scaled, and constrained by overnight batch windows that could no longer meet modern regulatory, operational, and analytical requirements. This paper presents a comprehensive architectural and methodological framework for building Authoritative Data Stores (ADS) for treasury deposits using Apache Spark as the core distributed processing engine. The framework addresses data sourcing, certification, governance, and standardized provisioning to downstream consumers such as regulatory reporting, risk analytics, finance, and real-time dashboards. It elaborates on enterprise migration drivers, architectural comparisons between legacy ETL systems and Spark-based ecosystems, and the key challenges encountered during large-scale modernization initiatives. A structured migration methodology is proposed, encompassing baseline legacy assessment, code translation, data quality reconciliation, performance optimization, orchestration redesign, and phased cutover strategies. The paper further illustrates best practices through representative case studies demonstrating significant performance gains, cost reduction, and improved scalability. The findings highlight Spark's role as a unifying compute platform capable of supporting batch, streaming, and advanced analytics workloads while maintaining the integrity and auditability required for treasury deposit data. The study concludes that Spark-enabled Authoritative Data Stores provide a resilient foundation for enterprise financial data modernization, enabling near real-time insights, improved governance, and long-term adaptability in evolving regulatory and business environments.

| KEYWORDS

Authoritative Data Store, Treasury Deposits, Apache Spark, Legacy ETL Migration, Data Certification, Enterprise Data Architecture, Distributed Processing

| ARTICLE INFORMATION

ACCEPTED: 01 February 2024 **PUBLISHED:** 28 February 2024 **DOI:** 10.32996/fcsai.2024.3.2.8

1. Introduction

1.1 Background and Enterprise Context (2018–2019)

In the 20182019 cycle, the need to modernize the data processing strategy of large organizations, especially financial services, was strongly experienced in the desire to support increasing volumes, [1-3] velocity, and regulatory attention. The monolithic ETL architectures had been supporting treasury deposit systems on which some of the most vital operations of the banking system, like aggregation of balances, tracking transactions, interest charge calculation and liquidity monitoring, are based. These legacy systems were based on an assortment of SAS batch programs, mainframe COBOL jobs, Informatica or Ab Initio mappings and

Oracle PL/SQL-based nightly pipelines. Historically sound and dependable, these architectures were progressively unable to cope with the demands of the modern finance of operations. With the increase of data to the terabyte and petabyte scale, overnight batch windows were increasing in length, processing failures had more significant consequences, and the cost of owning proprietary specialized infrastructure was increasing. Also, the inflexibility of these tools minimized flexibility, which reduced the possibility of incorporating new data sources, using powerful analytics, and responding quickly to changing business needs. At the same time, there were increased regulatory pressures, focusing on data lineage, full reconciliation and full auditability of data, with changing IFRS reporting requirements that required timely and accurate financial reporting. These operational and regulatory issues together made the inability of conventional ETL solutions quite clear and demonstrated a strong desire that scalable, flexible, and controlled data processing frameworks could provide trusted and high-quality data on treasury deposits in near real time.

1.2 Motivation for Migration

The drive behind the migration of legacy treasury deposit systems to new Spark-based systems was based on a cost, scale, flexibility and future-readiness factor. A constant need to reduce costs was among the most urgent factors because the enterprises were strongly dependent on proprietary systems like SAS and mainframes. These systems were also costly in terms of software licensing as well as specific skill sets and maintenance resources which added to high operation overhead. Moving to an open, distributed computing system such as Apache Spark may enable organizations to take advantage of commodity hardware and cloud computing infrastructure, and significantly lower licensing, hardware and operational expenses. Another important issue was scalability. The old architecture of ETL was not effective in processing terabyte- and petabyte-scale data and therefore, it had long batch windows leading to delays in the process of providing important treasury reports. The distributed processing capabilities of Spark have made it possible to have horizontal scaling on the clusters of commodity nodes, thus the ability of processing voluminous financial information in parallel. This will keep balances, transactions and liquidity metrics processed in a timely manner despite the increase in volumes of data.The advantage of the migration was also a single compute engine, which could support SQL-based analytics, machine learning (through MLlib), streaming, and graph processing on the same platform. This standardization made architecture easier, as well as simplified system complexity, and allowed enterprises to run sophisticated analytics and predictive modeling directly on treasury data without having to maintain separate processing engines. The migration, too, placed enterprises in the position of being cloud-ready, and as Spark can be connected to storage systems like Amazon S3, Azure Data Lake Storage (ADLS), and Google Cloud Storage (GCS) through built-in connectors, making it easy to deploy hybrids and multi-clouds as resources become more flexible. Lastly, the migration made it possible to process data almost in real time which was not possible before using Spark Structured Streaming instead of the old batch ETL processes. This capability enabled the treasury and the risk teams to have timely information and took shorter time to reconcile as well as responding to market or regulatory changes. All these made it an attractive business case to migrate legacy systems to a modern scalable and governed data architecture based on Spark.

1.3 Importance of Authoritative Data Stores for Treasury Deposits

Authoritative Data Stores (ADS) have been central in maintaining integrity, consistency and reliability of treasury deposit data in modern financial institutions. [4,5] Treasury deposits are complicated transactions, accrual of interest, liquidity monitoring and regulatory reporting all of which are based on quality and reconciled datasets. The legacy systems traditionally used to cause incomplete data replicas, manual reconciliations, and erroneous data in the downstream systems leading to operational lack of efficiency and compliance risks. An ADS implementation can help overcome these issues by providing one source of truthful all treasury-related datasets, integrating and standardizing information that comes from scattered legacy into a certified repository that is controlled.

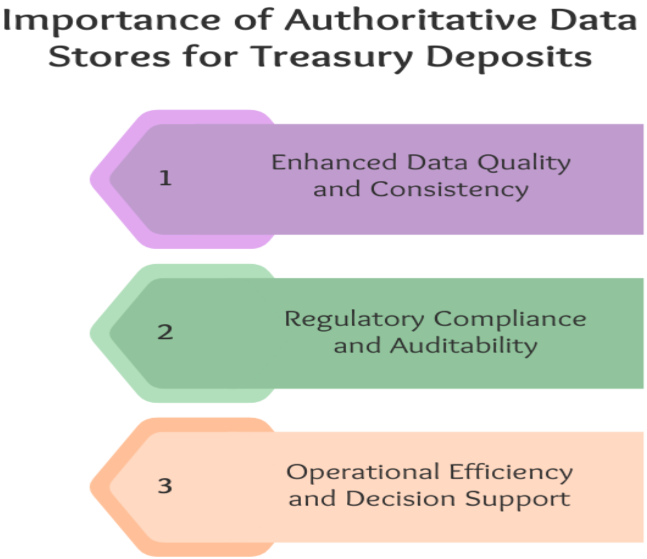
Importance of Authoritative Data Stores for Treasury Deposits
1
Enhanced Data Quality and Consistency
2
Regulatory Compliance and Auditability
3
Operational Efficiency and Decision Support

Figure 1 : Importance of Authoritative Data Stores for Treasury Deposits

- **Enhanced Data Quality and Consistency**

ADS has strict data quality rules such as validation, reconciliation and certification, which ensures that correct and consistent information is provided to all downstream consumers. This is vital in treasury activities where slight variations in balances, transactions or calculation of interest can have a serious financial and regulatory consequence. The ADS minimizes mistakes, manual processes, and reconciliations with the help of centralization of data and automated quality checks.

- **Regulatory Compliance and Auditability**

Treasury deposits are highly regulated under regulatory frameworks of BCBS 239 and IFRS standards which require transparency, traceability and lineage of data that is auditable. ADS fulfills these requirements inherently through the ability to capture metadata, the ability to keep accurate lineage, and imposes governance rules. This will make sure that every transformation and aggregation will be documented completely so that organizations can prove to be compliant when auditing or subjecting to regulators.

- **Operating Performance and Support**

The ADS enhances the operational teams and analysts to make more quicker and data-driven decisions by offering them a credible and certified source of treasury deposit information. It removes the necessity of having numerous reconciliations across silos and enables close real-time analytics, scenario modeling and reporting. Moreover, the ADS is an effective base of advanced functioning, such as machine learning, predictive analytics, and automated risk management, promoting overall treasury operations. Overall, Authoritative Data Stores are critical to treasury deposits because they promote quality, reconciled, and controlled data, regulatory compliance, decreased overheads, and a scalable basis on current financial analytic and decision-making.

2. Literature Survey

2.1 Evolution of Enterprise ETL Architectures

The initial generations of enterprise ETL architectures were mainly developed to handle structured, transactional data with known growth rates and fairly slow ingestion rates. [6-8] These systems were heavily based on centralized databases and batch processing where data consistency, reliability and easy maintenance were of greater concern as opposed to elastic scalability. Early writing by Kimball and Caserta included the use of dimensional modeling, star schemas and scheduled batch ETL pipelines as an ideal set of practices in enterprise data warehousing, especially in reporting and decision support. With the increasing size and heterogeneity of datasets stored by organizations, however, such conventional architectures became unable to support performance and scalability needs. With the development of distributed systems like Hadoop MapReduce, there was a real change in this sense because now processing could be done in parallel over the commodity hardware, however, when it came to complex, iterative or time-sensitive analysis programs, the introduction of disk based computation created some latency and inefficiencies.

2.2 Distributed Processing and Spark Adoption

The launch of Apache Spark was a significant improvement in distributed data processing because it has overcome most of the limitations presented in systems based on MapReduce. Zaharia et al. indicated that, in-memory computation model of Spark saved the execution time of iterative algorithms, interactive queries and streaming workloads by a significant margin. The unified programming model of Spark, which allows processing of batches, streaming, machine learning and graph analysis on one platform, further increased its adoption in enterprises. Following research and industry practice demonstrated that Spark can be used especially effectively in financial analytics, where it is necessary to make significant changes in data, combine things in various ways, and obtain insights in near-real time. Its fault tolerance, scalability and support of SQL-based analytics (through Spark SQL) made it an appealing platform to modernize the legacy ETL pipelines and support more advanced analytics use case.

2.3 Authoritative Data Stores and Data Governance

An Authoritative Data Store (ADS) is often placed as a trusted, reconciled, and controlled repository that becomes the authoritative source of the truth regarding consumption of enterprise data. An ADS is very important in promoting consistency, traceability, and compliance of data in the context of master data management and industries that are regulated like finance. The existing literature contains the emphasis that on top of storage and processing, an ADS should also include strong governance structures, such as metadata management, data lineage tracing, validation rules, and quality2.1 Evolution of Enterprise ETL Architectures. The initial enterprise ETL models were typically built to handle structured and transactional data, of predictable growth rates and relatively slow ingestion rates. They were very much based on centralized databases and batched processing putting data consistency, reliability and easy maintenance over elastic scalability. Early research by Kimball and Caserta had highlighted dimensional modelling, star schemas and planned batch ETL pipelines as the best practices in enterprise data warehousing, especially reporting and decision support. But as organizations started to grow in size and diversity of data, these conventional architectures became unable to support performance and scalability demands. The advent of distributed systems like Hadoop MapReduce was a huge change since it offered simultaneous processing of commodity hardware, but introduction of disk-based calculation resulted in latency and inefficient processing of complex, iterative, or time-sensitive analytic workloads.

2.4 Gaps in Existing Research

Despite a large body of research on the distributed data processing frameworks, and, independently, on the topic of data governance and authoritative data management, the research that combines the two areas into one common architectural solution is comparatively limited. Majority of the previous literature bases scalability and governance as two different issues, where people usually focus on the performance improvement, but they rarely pay adequate attention to the data certification, the lineage and even regulatory restrictions. The lack of this connection can be particularly observed within the framework of treasury and deposit data environments where high data volumes, high compliance requirements, and complicated reconciliation processes are present simultaneously. The absence of an integrated framework that is simultaneously responsive to the concepts of modernization of architecture, its efficiency in operations, and to its governance is an enormous gap in the literature. This paper aims to address this gap by suggesting a Spark-based Authoritative Data Store paradigm that integrates distributed processing capabilities with powerful data governance concepts adapted to the concept of modernization of treasury deposit data metrics. With these capabilities, organizational sustainability can be achieved to prove regulatory compliance, facilitate auditability and lessen operational risk. In the financial fields, especially where accuracy and timeliness of data have direct implications in the reporting and risk management processes, governance has never been a peripheral issue but an architectural prerequisite.

3. Methodology

3.1 Conceptual Flowchart

The concepts flowcharts show the entire pipeline of data movement and transformation pipeline of treasury deposit data modernization. [9,10] It illustrates the movement of data through non-homogenous source systems, ingestion and certification layers and then consumed by analytical and operational systems with a focus on scalability, governance and data reliability at each point in the system.

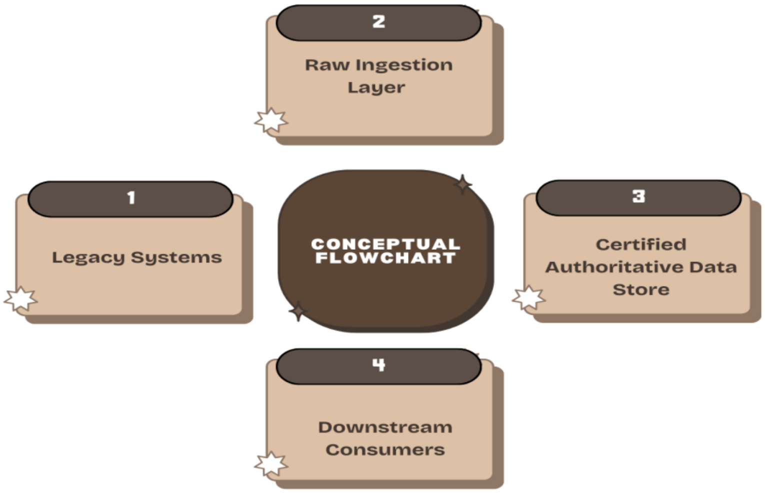

Figure 2 : Conceptual Flowchart

- **Legacy Systems**

The core banking platforms, transaction processing systems, and other complementary financial applications are the main sources of the treasury and deposit data. These systems usually produce formatted data (either in batch or near-real time), and are commonly based on inflexible schemas and proprietary technology. Legacy systems present a challenge in the form of data access, integration and scalability because of their age and design limitations, and require a strong downstream architecture to yield and standardize outputs in an efficient manner.

- **Raw Ingestion Layer**

The raw ingestion layer acts as the first place of landing of the data pulled out of the legacy systems. The key aim is to record the source data as it was originally with minimum transformation and they should be complete and traceable. This layer provides support to batch and streaming ingestion, and has historical snapshots to be reprocessed or audited or recover the errors. The raw ingestion layer makes systems more resilient by decoupling data acquisition and downstream processing to enable scalable data onboarding.

- **Certified Authoritative Data Store**

The heart of the architecture is the Certified Authoritative Data Store (ADS) whereby the data that is received is purged, reconciled, standardised and enriched based on specific business rules. Information in the ADS is subjected to data checks, quality controls as well as certification to guarantee accuracy and consistency in the domains. The ADS operates under metadata, lineage and access controls, making it a trusted single source of truth, which allows regulatory compliance and trusted downstream consumption.

- **Downstream Consumers**

Reporting platforms, risk and treasury analytics systems, regulatory reporting systems and external interfaces are some of the downstream consumers. These consumers will use the ADS to provide them with timely, consistent, and high-quality data, but with no extra reconciliation logic. The architecture allows many consumers to act autonomously and still adhere to enterprise data governance and performance needs by abstracting complexity, imposing standardized data contracts on the data.

3.2 Baseline Analysis and Mapping

Analysis and mapping stage will be addressed with the aim of determining an overview of the current data integration environment before modernization. [11,12] This step will entail a capturing of existing-state ETL processes, dependencies, transformation logic, and patterns of data utilization. Established through a comprehensive listing of the existing ETL assets, the organization achieves continuity of functionality, decreases the risk of migration, and facilitated mapping of the previous logic to the new Spark-based platform.

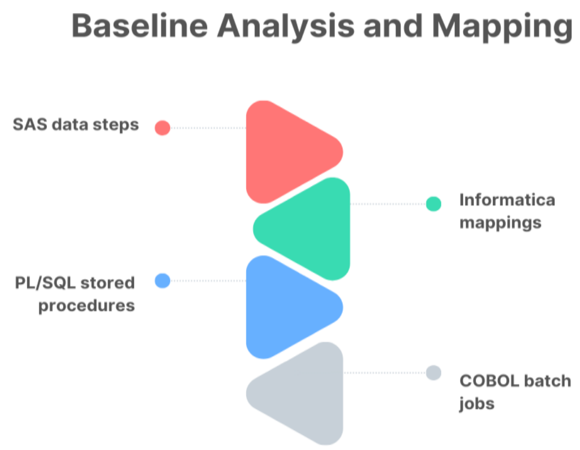

Figure 3 : Baseline Analysis and Mapping

- **SAS Data Steps**

Data extraction, transformation, and aggregation Data extraction, transformation, and aggregation The use of SAS data steps was common in the legacy environment, especially as a part of the analytical and regulatory reporting. In these scripts, intricate business policies and conditional logic in addition to data quality checks are usually entrenched over time. In the course of the baseline analysis, every SAS data step has been examined to determine transformation intent, input-output dependencies, and performance properties to provide the means to transcribe business logic into scalable Spark transformations.

- **Informatica Mappings**

Informatica mappings were a major component of the enterprise ETL layer, which coordinates data movement between the source systems and the downstream warehouses. Such mappings usually contain source qualifiers, transformations, lookups and workflow dependencies. The analysis process consisted of breaking down mappings to transformational units of logic and recording data flows, timing constraints and error-tolerating mechanisms. This mapping exercise enabled the re-implementation of the same logic using Spark SQL and Processing of DataFrames in addition to enhancing modularity and performance.

- **PL/SQL Stored Procedures**

Data validation, enrichment, and reconciliation were also done by using PL/SQL stored procedures that were used directly against relational databases. Although useful in the moderate data volume, these processes were frequently problematic on scalability as well as dependency on a particular database platform. Stored procedures were broken down as part of the baseline assessment to decouple data access logic and business rules so that computationally-intensive processing can be migrated to the distributed Spark environment without affecting the required transactional controls.

- **COBOL Batch Jobs**

The core of the old mainframe-based processing was made of the COBOL batch jobs especially those required at the end of the day and end of the year treasury. These are normally very stable and hard jobs where file formats are fixed and execution timeframes are strictly planned. The analysis at the baseline was aimed at learning the data dependencies, file structures, and processing sequences incorporated in COBOL programs. This experience was used to design similar batch processing workflows in the current architecture that provide continuity of the important treasury functionality and enhances the scalability and maintainability.

3.3 Data Ingestion and Layered Architecture

The data ingestion/layered architecture was developed to be scalable, auditable, and have robust data governance of treasury deposit data. [13,14] The architecture isolates issues of data capture, transformation, and certification by taking a multi-layered approach. This design allows the evolution of data to be controlled, it is easier to reprocess, and it is guaranteed that every layer has a clearly defined functional and governance objective.

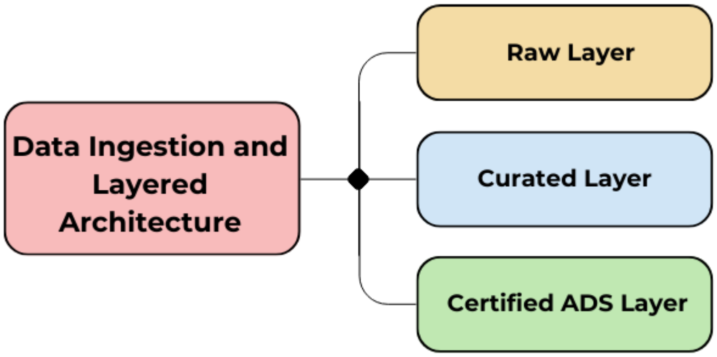

Figure 4 : Data Ingestion and Layered Architecture

- **Raw Layer**

The raw layer serves as the unchangeable landing point of all the source data that is fed by the legacy systems. Information is stored in the native format with little or no transformation and maintains the source fidelity and allows full traceability. This layer allows both batch and streaming ingestion patterns and historical data, which is critical to audit needs, error recovery, and downstream reprocess. The raw layer provides a reliable system of record of all the data of the treasury deposits that have been taken by maintaining immutability.

- **Curated Layer**

The curated layer standardizes transformations on raw data in order to fit enterprise data models and business definitions. This involves data cleansing, normalization, enriching and use of common reference data. The refined layer is also geared towards consistency and reuse guaranteeing that business rules are enforced uniformly across datasets. The architecture minimizes redundancy caused by the packaging of data by transformation logic at this layer and prepares data to be reliably reconciled and certified in later stages.

- **Certified ADS Layer**

Certified Authoritative Data Store (ADS) layer is the topmost data quality and control of the architecture. The input data in this layer has already been reconciled across the sources, validated against business and regulatory regulations, and checked in terms of completeness and accuracy. The certified ADS offers relied upon, consumption-readable treasury deposit data controlled by metadata, provenance, and access controls. This layer is the sole source of truth of the downstream consumers and it enables regulatory reporting, analytics and strategic decision-making with the assurance that the data is accurate.

3.4 Data Quality and Certification Controls

Data quality and certification controls constitute an essential part of the Authoritative Data Store as the data on treasury deposits is accurate, complete and ready to be used in the analysis and regulatory purposes. [15,16] The certification is done by a sequence of automated and repeatable controls implemented as a part of the data processing pipeline. These filters ensure integrity of data in every transformation process and they ensure a quantifiable level of assurance that only trusted data sets are advanced to the certified layer.

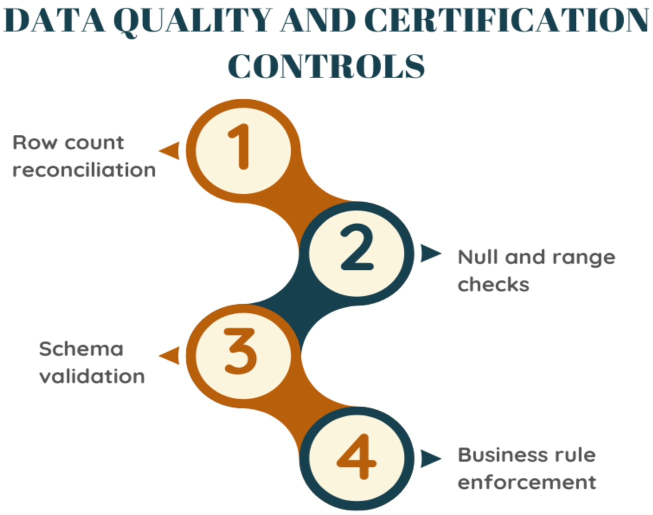

Figure 5 : Data Quality and Certification Controls

- **Row Count Reconciliation**

Row count reconciliation is used to confirm the completeness of the data and this can be done by comparing the number of records in the various source systems, layers and the certified ADS. This check assists in identifying any loss of data, duplication or partial ingestion of the content in ETL processing. Any discrepancies are identified as such to be investigated and all transactions of the treasury deposits and balances are properly accounted to prior to certification. Auditability and operational monitoring of data pipelines is also facilitated by row count measures.

- **Null and Range Checks**

Critical data elements undergo null and range checks to check the accuracy of data and business feasibility. Mandatory fields including account identifiers, balances, date, and transaction are verified against null or empty values and numeric fields are verified against acceptable ranges which may be based on business rules. These checks are used to detect the presence of data corruption, source system problems, or areas where the transformation has gone wrong at an early stage of the processing lifecycle before invalid data is sent to the end consumer.

- **Schema Validation**

Schema validation checks that the data received and converted follows the existing structural specifications such as the type of data, field names and relationships. This control identifies unanticipated schema modifications by the upstream systems including the addition, removal, or modification of columns that would otherwise cause downstream processing or reporting to fail. The architecture provides stability and consistency of data between treasury deposit datasets by the virtue that schema compliance is enforced.

- **Business Rule Enforcement**

Business rule implementation takes domain specific logic to test data and verify that it is semantically correct. Some of them are balance reconciliation rule, currency consistency rule, and product-specific rule. These regulations are the coded treasury and regulation information in the data line, which makes certified datasets truly represent real-life financial realities. The automated business rule implementation minimizes human operator intervention and leads to the increased confidence in the certified ADS as a credible source of truth.

3.5 Performance Optimization Methods

The design of the Spark-based Authoritative Data Store that required performance optimization crowded out other design alternatives due to the processing of treasury deposit data in large volumes and time constraints. [17,18] A number of Spark-specific optimization methods were systematically used to guarantee an efficient use of resources, minimized execution times, and predictable pipeline performance. The partition pruning was one of the key approaches that allowed Spark to scan only the data subsets that were relevant according to filter predicates like business dates or account ranges. Data partitioning was also done to match popular query patterns thus reducing unnecessary I/O which played a significant role in improving the performance of batch processing and analytical queries. The other important streamlining was that small reference and dimension tables, including product codes, currency mappings, and organizational hierarchies, were done by broadcast joins. By sending these datasets to all executor nodes, Spark did not have to incur expensive shuffle processes that were normally linked with huge joins. This method minimized the network overhead and provided better join execution time especially when large fact like treasury data had to be augmented with lookup tables that were relatively small. Shuffle tuning played a significant role in enhancing job stability and performance as well. The default Spark shuffle settings were changed to be optimal in regard to memory usage, parallelism, and spill behavior. The number of shuffle partitions, buffer sizes and compression parameters were adjusted according to the workload characteristics to minimize disk I/O and eliminate executor memory pressure. Correct shuffle tuning was used to overcome the performance bottlenecks due to skewed data distributions and massive aggregation functions typical of treasury computations. Lastly, the in-memory caching was planned to apply to data sets that are frequently accessed and intermediate outcomes that are utilized at the various processing phases. Retention of these datasets in memory by Spark removed any duplicative calculation and disk reads, thus, hastening iterative transformations and validation procedures. The size of the dataset, frequency of reuse and memory availability were used in caching decisions to trade off between performance gains and resource constraints. A combination of these optimization methods helped the architecture to achieve high-performance and scalability requirements, as well as ensuring data integrity and operational reliability.

3.6 Orchestration and Scheduling

Orchestration and scheduling was updated to enable the higher complexity, scalability and observability demands of the Spark-based data architecture. [19,20] Conventional enterprise schedulers like Control-M and Autosys, though capable of reliable operation on monolithic batch workflows, were not very flexible in supporting dynamic dependency, conditional logic and data-based execution patterns. To overcome these shortcomings, it has implemented Apache Airflow as the main orchestration engine alongside Spark-submit pipelines to run distributed processing work. Apache Airflow created the possibility of defining data pipelines as directed acyclic graphs (DAGs), giving explicit control over the task dependencies, order of execution, and failure behavior. This dependency execution model enhanced reliability because it meant that downstream Spark jobs could only run when their upstream ingestion, transformation, and validation jobs had completed successfully. The ability to use parameterization and templating in airflow also helped with environment-specific settings and date-driven processing, which are important to treasury deposit batch cycles and regulatory reporting schedules. Spark-submit together with Airflow gave a pure isolation between coordination and execution. Workflow scheduling, retries and alerting were handled by Airflow whereas resource allocation and job execution on the cluster was handled by Spark-submit. The separation enhanced the clarity of the operations and made it easier to troubleshoot failures could be traced to either orchestration logic or Spark processing. Also, Spark-submit settings were standardized and forced the same use of resources, logging and performance tuning were applied in pipelines. Better observability was one of the advantages of the new orchestration approach. The in-built monitoring, logging and visualization facilities of Airflow made it possible to see real-time the status of the pipeline, the duration of tasks and the point of failure. This provided increased visibility which minimized mean time to fix operational problems and allowed proactive performance tuning. In general, moving to the Airflow-based orchestration provided more agility, resiliency, and operational visibility than legacy scheduling tools and matched data operations with one of the current data engineering best practices.

4. Results and Discussion

4.1 Performance Improvements

The upgrade to a processing architecture based on Spark led to significant performance gains on key treasury data pipelines. The modernized pipelines had shortened end-to-end processing times by a significant margin as a result of distributed in-memory computation and optimized execution strategies that allowed quicker data availability and enhanced operational efficiency.

Pipeline Type	Runtime Reduction (%)
Treasury Balances	89.6%
Interest Accrual	88.3%

Table 1 : Performance Improvements

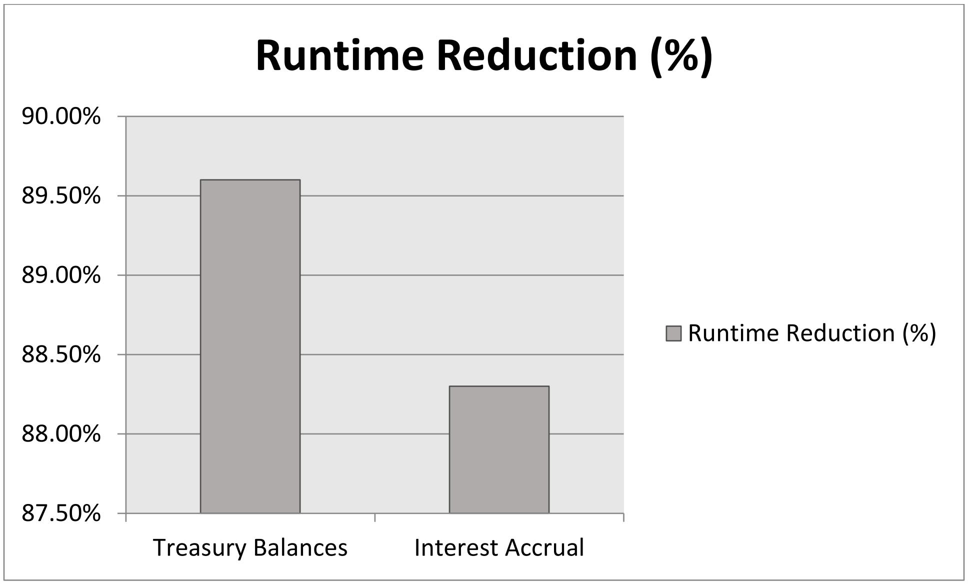

Figure 6 : Graph representing Performance Improvements

- **Treasury Balances**

The runtime of the treasury balances pipeline decreased by 89.6, which is a radically different result compared to the old system of pipeline processing. The most common aspects that contributed to this gain include parallel execution, partition-sensitive processing, and removal of sequential, disk-intensive processing that existed in the traditional ETL environments. The balance data processing was faster so that daily positions would be available earlier, reconciliation schedules would be improved, and downstream analytics and reporting would be more flexible.

- **Interest Accrual**

The accrual pipeline achieved a 88.3 factored interest decrease in the form of identical advantages of in-memory calculation and stream-lined joins contributed by Spark. Computations of interest are usually done in an iterative fashion on high levels of data and reference tables that in previous times led to periods of long batch windows. The modernized pipeline minimized the computation latency and enhanced the predictability of batch operations and also helped in meeting stricter regulatory and financial reporting deadlines, making the treasury operations generally better.

4.2 Cost and Scalability Benefits

Modernizing treasury data processing provided significant cost and scalability advantages by abandoning proprietary, closely coupled legacy systems in favor of an open, distributed and elastic computing architecture. The decommissioning of the SAS and mainframe-based ETL workloads was one of the biggest cost savings. Traditional analytics and batch processing systems are normally costly to license and operate in specific fields of expertise as well as very costly in terms of overhead costs. Moving transformation logic to Apache Spark helped organizations to remove the software licensing fee that was occurring regularly, minimize reliance on mainframe and SAS skills which was also scarce and also made it simpler to maintain over the long term. Moreover, the offloading of mainframe workloads lowered both infrastructure expenses in the form of special hardware, storage, and energy expenditures. Additional cost savings were accomplished through the use of commodity Hadoop clusters or cloud based infrastructure. Contrary to legacy systems, which are more vertically scaled and require capacity to be provisioned to peak loads, the distributed platforms can be scaled horizontally with inexpensive commodity servers or on-demand cloud solutions. This elasticity enabled organisations to match the compute usage with the actual processing demand and therefore scale up the clusters during peak batches and scale down during idle periods. The pay-as-you-go pricing models in cloud deployments also further optimized the costs because it did not require an initial capital investment and long-term capacity planning.Scalability-wise, the Spark-based architecture offered the possibility to not only support the increasing amounts of data, faster transactions, and more data sources without making major architecture changes. This could be expanded by increasing cluster capacity instead of remodeling pipelines to add new datasets and use cases. This was flexible and time-to-market new analytical and regulatory requirements was minimized. All in all, the three aspects of a license cost removal, infrastructure agility, and linear scalability allowed organizations to realize sustainable cost savings and future-proofed treasury data platforms as the organization continued to grow and become more complex.

4.3 Data Quality and Governance Outcomes

The deployment of Certified Authoritative Data Store (ADS) has provided significant benefits in improving data quality and governance in the treasury deposit operations which provided a measurably better trust, compliance and operation efficiencies. Through the application of automated validation, reconciliation and certification mechanisms, the ADS managed to make sure that all the data distributed to the downstream consumers was correct, complete, and consistent. Row count reconciliation, null and range checks, schema validation and business rule enforcement were critical controls that offered systematic verification in every step of transformation of data. Subsequently, the frequency of data inconsistencies and discrepancies was significantly important thus giving finance departments, risk management and reporting systems more confidence in the information they were using to make decisions. Among the most practical results was a decrease in the number of defects in the reconciliation. Downstream systems and reporting processes used to have to be manually adjusted or cross-system checked to clear discrepancies between various legacy sources. Having certified ADS data, these differences were reduced at the source, massively reducing the operational burden and saving staff to spend their time on more valuable analytical and strategic work. Auditability was also enhanced by the automatism and repeatability of the certification processes. Metadata tracking, data lineage, and detailed logging enabled regulators and internal auditors to check the integrity of data, logic of transformation and compliance adherence in a short period of time and mitigate regulatory risk and reduced the audit findings. In addition, there was enhanced decision-making and resilience in operations due to better governance. Reconciled and standardized data provided easier reporting, forecasting, and analytics which supported both day-to-day treasury activities and strategic initiatives. The ADS had governance structures that ensured that access controls, data stewardship duties, and quality checks were always implemented throughout all datasets. The centralization and advancement of reliably held data facilitated accountability and helped consolidate a single source of truth which enhanced teamwork between the business and technology teams. In general, the Certified ADS did not only enhance the reliability of the data, but also provided a pathway to the continued state of governance maturity, allowing data operations in the treasury area to be scalable, compliant, and of high quality.

4.4 Case Study Examples

Specifically, modernization of the old treasury and financial pipelines is one of the real-life examples of the advantages gained with the help of the Spark-based networks. An excellent case was a 20-year old SAS credit-risk pipeline that was being heavily burdened with slow batch processing and high operational complexity. The traditional pipeline was also based on the sequential processing of SAS data and the interactions with the database, which led to a slow runtime lasting many hours (or even nights) to report the risks and make decisions. The transfer of this workload to Apache Spark allowed parallel and distributed processing and in-memory computation and significantly speeded up the processing. The pipeline was 10x faster as legacy transformation logic was translated to optimized Spark jobs and a variety of techniques were employed (including partitioning, broadcast joins, and caching of previous results). It enabled real-time or close-to-real-time credit risk, increased responsiveness to market fluctuations and lower operations overheads incurred on manual monitoring and batch reprocessing. Another example was mainframe-based deposit ingestion which has traditionally been performed under a batch window throughout the night on COBOL jobs and hard-wired file transfer operations. These restrictions resulted in the inability to see account balances, deposits and transactional activity in time which influenced the treasury functions and reporting. The modernization initiative substituted the batch workflow based on the mainframe with a Spark and Kafka-driven ingestion pipeline thereby facilitating the real-time ingestion of deposit data. The processing, transformation and enrichment of incoming records was done by Spark and Kafka was a scalable fault-tolerant messaging layer used to send and receive real-time data. This architecture enabled downstream consumers to get near real-time deposit information, enabling quicker reconciliations, dynamic reporting and better operational flexibility. Also, it was the new system that entailed the use of automated verification, reconciliation and certification measures, and guaranteed the quality of data without compromising the compliance with regulatory and audit requirements. Collectively, these case studies explain how contemporary distributed processing systems are able to bust through the constraints of older systems, providing radical performance, scalability, data quality and operational visibilities. They point to the physical advantages of re-architecturing key financial pipelines to accommodate the needs of modern treasury functions.

5. Conclusion

The shift of the legacy ETL platform to Authoritative Data Stores (ADS) based on Spark was a turning point in digital transformation of companies, especially those in the treasury and deposit data management, in the 2018-2019. The previous ETL systems including SAS, Informatica, PL/SQL, and batch jobs run on a mainframe were severely limited by their scalability and performance capabilities, as well as their ability to turn applications on and off. They were based on serial processing, commercial software licenses, and fixed architectures that were hard to scale to increasing data volumes, more complex transactions and the changing regulatory demands. These challenges were met by the adoption of Spark-based architectures that exploited distributed, in-memory computation, allowed large datasets to be processed by using parallel computation, and could now take several minutes instead of several hours to run a batch. This performance enhancement has improved not only the availability of data faster, but also enabled the treasury and risk teams to make informed decisions on time to facilitate the operational efficiency and responsiveness of a business.

One of the most important steps in the modernization endeavor was the creation of a Certified Authoritative Data Store that was a single source of truth that was trusted by downstream consumers. The ADS reviewed the accuracy, completeness, and consistency of treasury deposit datasets by using automated data quality controls, such as checks on row counts, nulls and range, schema, and business rule. These mechanisms did not only minimize errors during reconciliation and the involvement of human hands, but also enhanced governance and auditability, and minimized the regulatory risk. Metadata control, lineage tracing, and access control ensured further the consistency and transparency of the information and allowed the enterprises to prove their compliance and keep the stakeholders trust in reporting and analytics.

The modernisation of orchestration and workflow management was also important. It was also possible to use the observability, dependency-based execution, and error handling of legacy schedulers like Control-M and Autosys, using Airflow and Spark-submit pipelines. Such a blend of scalable processing, effective data management and modern orchestration developed a resilient and sustainable structure capable of adapting to future increase in data volumes and analysis needs.

The effectiveness of such a transformation proves that, when migration planning is well practiced, when the logic of legacy ETL is carefully mapped out, when strict quality and governance controls are applied, the enterprises can realize significant operational and financial rewards. Authoritative Data Stores based on Spark offer not just better performance and cost-efficiency, but also a future-proof architecture that is able to deliver advanced analytics, regulatory compliance, and strategic decision-making. With the growing complexity and speed of data organizations encounter, ADS systems based on Spark are still a pillar of contemporary data management in enterprises, where the reliability, efficiency, and scalability of data-driven insights are promised.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1]. Kimball, R., & Caserta, J. (2004). The data warehouse ETL toolkit. John Wiley & Sons.

[2]. Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[3]. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. In 2nd USENIX workshop on hot topics in cloud computing (HotCloud 10).

[4]. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Stoica, I. (2016). Apache spark: a unified engine for big data processing. Communications of the ACM, 59(11), 56-65.

[5]. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). Database system concepts (Vol. 5). New York: Mcgraw-hill.

[6]. Loshin, D. (2010). Master data management. Morgan Kaufmann.

[7]. White, T. (2012). Hadoop: The definitive guide. " O'Reilly Media, Inc.".

[8]. Sakr, S., & Zomaya, A. Y. (Eds.). (2019). Encyclopedia of big data technologies. Berlin, Germany: Springer International Publishing.

[9]. Zikopoulos, P., & Eaton, C. (2011). Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media.

[10]. International, D. (2017). DAMA-DMBOK: Data management body of knowledge. Technics Publications, LLC.

[11]. Lenzerini, M. (2002, June). Data integration: A theoretical perspective. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 233-246).

[12]. Supervision, B. (2011). Basel committee on banking supervision. Principles for Sound Liquidity Risk Management and Supervision (September 2008).

[13]. Ladley, J. (2019). Data governance: How to design, deploy, and sustain an effective data governance program. Academic Press.

[14]. Polak, P., Nelischer, C., Guo, H., & Robertson, D. C. (2020). "Intelligent" finance and treasury management: what we can expect. Ai & Society, 35(3), 715-726.

[15]. Yaker, I. F., & Pattanayak, S. (2010). Treasury single account: concept, design and implementation issues. International Monetary Fund.

[16]. Mehmood, E., & Anees, T. (2022). Distributed real-time ETL architecture for unstructured big data. Knowledge and information systems, 64(12), 3419-3445.

[17]. Mishra, S., Komandla, V., & Bandi, S. (2022). Leveraging in-memory computing for speeding up Apache Spark and Hadoop distributed data processing. International Journal of Emerging Research in Engineering and Technology, 3(3), 74-86.

[18]. Gopalani, S., & Arora, R. (2015). Comparing apache spark and map reduce with performance analysis using k-means. International journal of computer applications, 113(1), 8-11.

[19]. Dreibelbis, A. (2008). Enterprise master data management: an SOA approach to managing core information. Pearson Education India.

[20]. Durelli, R. S., Santibánez, D. S., Marinho, B., Honda, R., Delamaro, M. E., Anquetil, N., & de Camargo, V. V. (2014, August). A mapping study on architecture-driven modernization. In Proceedings of the 2014 IEEE 15th international conference on information reuse and integration (IEEE IRI 2014) (pp. 577-584). IEEE.

[21]. Karkošková, S. (2023). Data governance model to enhance data quality in financial institutions. Information Systems Management, 40(1), 90-110.

[22]. Tang, S., He, B., Yu, C., Li, Y., & Li, K. (2020). A survey on spark ecosystem: Big data processing infrastructure, machine learning, and applications. IEEE Transactions on Knowledge and Data Engineering, 34(1), 71-91.