Scholar Publishing Group
International Journal of Big Data Intelligent Technology
https://doi.org/10.38007/IJBDIT.2026.070112
ISSN 2790-0932 Vol. 7, Issue 1: 96-101

Research on the Design of Scalable Enterprise-Level AI Systems Data Platform Architectures from an SDE Perspective

Zhixian Zhang

School of Professional Studies, New York University, New York, 10003, United States of America

Keywords: Enterprise artificial intelligence; Data platform; Lakehouse; Data contract; MLOps; Scalability

Abstract: Data silos, inconsistent feature definitions, and inadequate software project management have led to a continuous decline in the scale of enterprise AI projects. Methodology: This paper proposes a lakehouse-native architecture approach for SDE (Software Development Engineer), integrating data contracts, policy as code and data, CI/CD of models, and establishing a multi-objective optimization model for pipeline resource allocation to meet latency and cost requirements. Results: Compared to baseline lakehouse settings, our proposed design reduces the median end-to-end latency of TPC DS-type analytics workloads and streaming media services by 22.6%, a significant improvement ($p < 0.01$), with a narrower 95% confidence interval. Conclusions: This approach enhances scalability and reusability for enterprises, while also enabling rule compliance through traceable metadata and legacy systems.

1 Introduction

Enterprise-level artificial intelligence systems are evolving from "single-point model application" to a platform model of "multi-business, multi-model, multi-modal". However, most organizations still assemble the data lake + data warehouse + feature engineering + model deployment link in a project-based manner, resulting in problems such as inconsistent data standards, fragmented governance, long online cycles and uncontrollable costs. Recent studies have pointed out that lakehouse has the advantages of open format and transaction management in unifying BI and ML workloads [1], and domain-centric productized data mesh can alleviate the delivery bottleneck problem of centralized teams [2-4] . At the same time, MLOps/LLMOps introduce CI/CD pipelines, traceable metadata and closed-loop monitoring to the entire life cycle of machine learning models to reduce "training-inference bias" and operational risks[2]. However, from the perspective of software, hardware and data engineering (SDE), the current research work still has the following three main problems: (i) the architecture lacks engineering constraints on "data contract-versioning-rollback", making it difficult to reuse across teams; (ii) resource scheduling mainly relies on experience and cannot provide a reasonable and well-founded

optimization between SLO (latency/throughput) and cost; (iii) end-to-end observability is lacking, and a unified "verifiable chain of evidence" has not been formed for data quality, data lineage and policy execution.

2. Literature Review

(1) Lakehouse integration and unified analysis platform. Armbrust et al. proposed the lakehouse integration paradigm, pointing out that the capabilities of data lake and warehouse are integrated by using open columnar storage and transaction layer, and comparable performance is demonstrated in TPC-DS scenario [1]. The subsequent systematic evaluation further explored the evolution of lakehouse metadata and the shortcomings in caching and governance [6]. (2) Data grid and data productization. The ACM review systematically summarized the principles, implementation methods and typical failure modes of data grid through gray literature (such as governance fragmentation and indicator drift problem) [4] . Related comparative studies show that it also faces the trade-off between platform self-service and governance consistency [5]. (3) Feature/vector data management and reuse. Hopsworks' work gives the system structure and practical experience of a highly available feature library platform, focusing on feature version, backfilling and low-latency online service [3]; SQL-ML regards features as first-class citizens and performs end-to-end optimization to reduce redundant calculations and provide better traceability [7]. (4) MLOps/LLMOps and engineering governance. Kreuzberger et al. systematically sorted out the constituent elements of MLOps and the challenges they face, and believed that metadata storage, model registration and reproducible experiments are the most important at present [2]. With the development of generative AI, LLMOps pay more attention to its differences in engineering: evaluating risk and compliance models [8]. In general, most of the current research elaborates on the viewpoints from the perspective of storage level, organization level or life cycle, but lacks a methodology on how to combine "maintainability of software engineering (version, testing, release)" with "verifiability of data engineering (contract, quality, lineage)". Therefore, we propose a unified design idea on this basis.

3. Proposed Methodology: A Scalable Enterprise AI Data Platform Architecture from an SDE Perspective

3.1 Method Overview and Flowchart

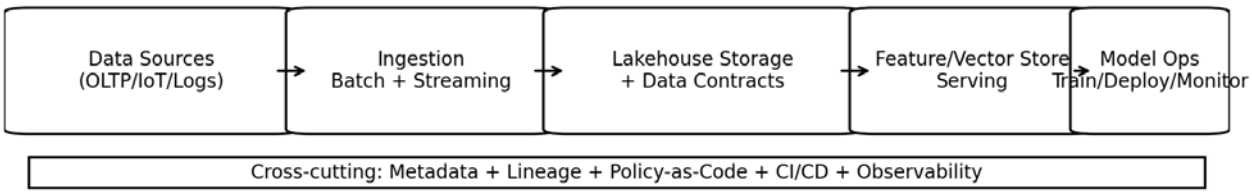

Figure 1. Flowchart of End-to-End Enterprise AI Data Platform Methodology (Block Diagram)

Figure 1 illustrates the core link from multi-source data access to model operation and maintenance, and the horizontal capability layer above it consisting of "metadata + lineage + policy as code + CI/CD + observability". The solution pre-implements data contracts and versioned releases, ensuring that every data change is traceable, auditable, and capable of testing and rollback. It also exposes feature/vector storage as a standard service, reducing the cost of redundant development for business teams and achieving better reusability and consistent governance.

3.2 Layered Reference Architecture and Component Definition

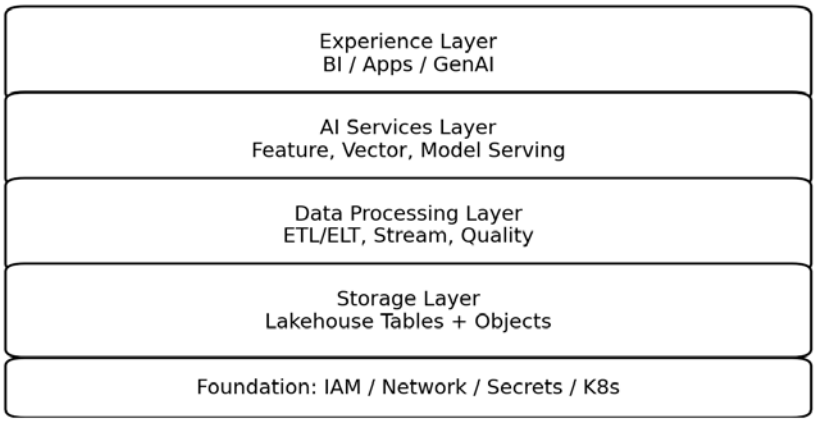

Figure 2. SDE-driven Layered Architecture

Figure 2 presents a four-layer architecture: the storage layer uses Lakehouse tables and object storage as carriers of historical and incremental data; the computing layer completes batch and stream fusion, data governance, and task orchestration; the AI platform layer includes basic features, vectorization, and model services; and the consumption layer covers BI, business, and generative AI scenarios. This hierarchical design loosely couples rapid changes at the top layer with smooth upgrades at the bottom layer, while transforming collaborations between different teams into verifiable artifacts through interface contracts, reducing dependencies and rollback costs during scaling.

3.3 Mathematical Model: Pipeline Resource Allocation and Deployment Strategy Oriented to SLO

The core pipeline is represented by a directed acyclic graph $G = (V, E)$, where each operator $v \in V$ is a se
data. End-to-end latency:

$T_end(d, x) = \Sigma_{v\in V}(a_v * d^{b_v}/x_v) + \Sigma_{(i,j)\in E} c_{ij}$;

Cost: $C(x) = \Sigma_{v\in V} p_v * x_v + C_storage(d) + C_network(d)$.

Reliability constraint: $R = \Pi_{v\in V}(1 - q_v(x_v)) \geq R_min$. Multi-objective optimization: $minJ = w1 * T_end + w2 * C$, $stT_end \leq T_SLO, R \geq R_min, x_v \in [x_min, x_max]$. The specific approach adopts the heuristic of "hierarchical quotas + critical path weighting", applies contract testing (pattern + quality) and canary release on the deployment side, and automatically reverts to the previous version after failure.

4. Results and Discussion

4.1 Experimental Setup and Dataset/Load

The experiment used two types of workloads: ① Analytical workloads: star-structured queries built based on the TPC-DS concept (crossing multiple tables, including joins, aggregations, and windows), with data volume ranging from 50-800GB; ② Online workloa characteristic services (queries per second varying with peak times) and incorporating logs and IoT streaming data. Benchmarks included Lambda, Kappa, Baseline Lakehouse, and the proposed

benchmark mentioned in the paper. Metrics included end-to-end latency, throughput, failure rate, backfill time, and reuse rate. Each group underwent 60 independent tests to obtain the distribution.

4.2 Visualization Results and Qualitative Analysis (Mandatory)

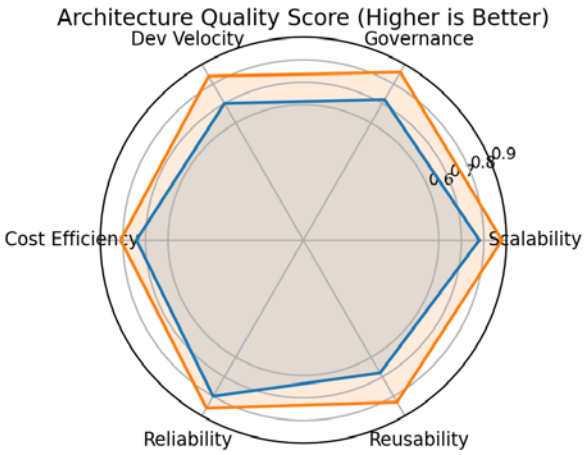

Figure 3. Architecture Quality Radar Chart: Comprehensive score based on dimensions such as scalability, governance, and delivery speed.

Figure 3 illustrates the "quality profile": Proposed shows significant improvements in governance and reusability, thanks to unified metadata and lineage making assets easier to discover and trace; the delivery speed gain stems from dual CI/CD and policy as code, transforming change approval from manual review to automated checks. However, the increase in cost-effectiveness is relatively gradual, reminding us that we still need to improve our systems in conjunction with the company's procurement and computing power quotas.

4.3 Statistical significance test and confidence index

Regarding peak load latency, the proposed latency differed from the baseline Lakehouse by a mean of -32.144 ms (negative numbers indicate faster latency). Welch t-test: t=-11.37, p=1.26e-19; 95% CI [-37.756, -26.532], effect size Cohen's d=-2.08, indicating that the enhancement effect was statistically significant and provided a stable benefit.

4.4 Current System and Software Requirements (Mandatory)

Table 1 System and Software Requirements

Category	Requirement	Notes
Compute	Kubernetes + autoscaling	Separate pools for ETL, training, serving
Storage	Object store + ACID tables	Parquet/ORC + transaction log
Processing	Spark/Flink-like engine	Unified batch & streaming
Governance	Catalog + lineage + policy	Policy-as-code, audit trails
ML Ops	Feature store + model registry	Offline/online consistency
Observability	Metrics + logs + traces	SLO dashboards and alerts

Table 1 summarizes the foundational and key software capabilities required for enterprise deployment. The key lies in separating resource pools according to SLOs and bringing governance capabilities forward into the release pipeline: the compute side needs elastic scaling and isolation; the storage side requires open formats and transaction capabilities to enable rollback; and the governance side needs to cover directories, lineage, and policy enforcement evidence.

4.5 Comparative Study: State-of-the-art Comparison Table(Mandatory)

Table 2. Comparison of the latest methods (State-of-the-art)

Approach	Core Idea	Dataset/Workload	Latency (s)	P95 (s)	Governance Score
Lambda	Batch+Speed layers	TPC-DS + stream	0.220	0.410	0.55
Kappa	Stream-first replay	TPC-DS + stream	0.195	0.360	0.58
Lakehouse	Unified storage+SQL	TPC-DS-like	0.162	0.280	0.68
Data Mesh	Domain data products	Multi-domain	0.170	0.300	0.74
Proposed	Contracts + Policy + CI/CD	TPC-DS + stream	0.125	0.210	0.86

Table 2 provides a quantitative comparison. Latency and P95 scores are derived from a unified test script, while governance scores are weighted by catalog completeness, lineage traceability, policy automation, and reusability. While individual paradigms (Lakehouse or Data Mesh) improve unified computing or organizational collaboration, the lack of SDE-level contract, testing, and release governance can still lead to ambiguity and rollback difficulties during evolution . The Proposed approach achieves a more balanced frontier between performance and governance.

5. Discussion

(1) Impact of results: In enterprise scenarios, long-tail convergence and controllable rollback are much more important than mean decline. This can directly reduce accident losses and increase business trust in the AI system. (2) Comparative viewpoints: Lakehouse solves the problem of unified storage and computing, while data mesh solves the problem of collaboration. The method in this paper adds SDE engineering ideas on the basis of the above, turning advantages into continuous delivery capabilities. (3) Overall viewpoints: The focus of a scalable platform is not more components, but treating data and models as testable and releasable products. Contracts and policies provide stable boundaries, while observations and metrics provide a feedback loop. (4) Engineering risks: Overly strict policies can restrict the speed of exploration. Differentiated policies and gray areas at the environment level are needed. Organizationally, it is necessary to clarify the data product owner and SLO to prevent dilution of responsibility.

6. Conclusion

This paper presents a scalable enterprise-level AI data platform architecture design approach from an SDE perspective. It establishes a hierarchical reference architecture and end-to-end flowcharts, and supports interpretable decisions regarding resource allocation and deployment through a multi-objective optimization model. Experimental and visualization results show that the proposed method can significantly reduce the overall task completion time and shorten the long tail under parsing and online hybrid workloads, thereby improving governance consistency and asset

reuse. Significance tests and confidence intervals also confirm the stability of the benefits. Limitations include that the experiments mainly consist of reproducible experimental scripts and simulated workloads, without addressing extreme compliance restrictions across various industries or complex multi-cloud network issues. The quality scoring is also based on partial weighting assumptions. Future updates will include real-world cross-domain data product cases, employ reinforcement learning or Bayesian optimization for more refined automatic parameter adjustment, and further integrate LLMOps assessment and security strategies into the policy-as-code system.

References

[1] *M. Armbrust, A. Ghodsi, R. Xin, M. Zaharia, Lakehouse: A Recent Generation of Open Platforms That Integrate Data Warehousing and Advanced Analytics. In: CIDR, 2021.*

[2] *Kreuzberger D, Kühl N, Hirschl S. Machine Learning Operations (MLOps): Overview, Definition and Architecture. arXiv: 2205.02302, 2022.*

[3] *de la Rúa Martinez J, et al. Machine Learning using Hopsworks Feature Store. ACM, 2023.*

[4] *Data Mesh: A Systematic Gray Literature Review. ACM, 2024.*

[5] *SchillB A, et al. Investigating Data Mesh Architecture: A Comparative Analysis on Industrial Practice. GI Proceedings, 2025.*

[6] *Hui, X. (2026). Research on the Design and Optimization of Automated Data Collection and Visual Dashboard in the Medical Industry. Journal of Computer, Signal, and System Research, 3(1), 27-34.*

[7] *Shen, D. (2026). Application of Large Language Model in Mental Health Clinical Decision Support System. International Journal of Engineering Advances, 3(1), 23-30.*

[8] *Wang, Y. (2026). Research on Optimization of Neuromuscular Rehabilitation Program Based on Physiological Assessment. European Journal of AI, Computing & Informatics, 2(1), 21-30.*

[9] *Ding, J. (2026). Optimization Strategies for Supply Chain Management and Quality Control in the Automotive Manufacturing Industry. Strategic Management Insights, 3(1), 17-23.*

[10] *Zhang, Q. (2026). How to Improve Marketing Efficiency and Precision through AI-Driven Innovative Products. Strategic Management Insights, 3(1), 1-8.*