Data Engineering and Applications

Research Paper

Cloud-Native Framework for Large-Scale Big Data Governance

Zuzanna Chmielewska[1,*] and Teresa Zającowa[1]

[1] Faculty of Informatics and Information Technology, Silesian University of Technology, Gliwice, 44-100, Poland
*Corresponding author: zuzanna.c@polsl.pl

Abstract. The data-asset management system, data-quality assurance, lineage tracking, security compliance, and intelligent decision support now all depend on large-scale big data governance. Conventional governance systems are not appropriate for scaling with heterogeneous data sources, frequent metadata updates, and multi-tenant governance workloads since they often feature centralized metadata repositories and monolithic task engines. This study proposes a Cloud-Native Framework for Large-Scale Big Data Governance. Incorporate distributed metadata management, lineage graph coordination, policy-based access control, data quality rule execution, containerized governance services, and elastic governance job scheduling into the system. For dispersed data assets, a lightweight governance control plane will perform audit tasks, update lineage information, and manage metadata collection and quality evaluation. In comparison to the centralized governance deployment, the experimental analysis shows that the proposed framework has improved elastic scaling efficiency by 31.6%, decreased average metadata query latency by 47.3%, and increased governance task throughput from 8,600 to 18,900 tasks per minute. The study offers a workable, engineering-focused architecture for building flexible big data governance systems in cloud-native settings.

Keywords: *Cloud-Native Framework, Big Data Governance, Metadata Management, Data Quality, Elastic Scheduling*

Received on 09 October 2021, Accepted on 29 December 2022, Published on 04 January 2022

Introduction

Businesses, governments, and other organizations that use the vast volumes of data now have to set up data governance systems. Governance must now concurrently address asset registration, metadata consistency, data quality, lineage traceability, access control, and compliance auditing as the range of data sources grows from traditional databases to logs, data lakes, streaming sensors, public interfaces, and AI training repositories. Currently, the goal of data governance in many organizations is to guarantee the quality of generated data and encourage its exchange and reuse rather than building a single warehouse for all data. Therefore, the idea of governance has shifted from traditional document management to continuous platform engineering due to data-driven applications [1]. Service decomposition, container orchestration, automated deployment, and elastic resource allocation are all supported by cloud-native computing, which provides a new execution environment for this shift [2]. The capabilities are necessary in a large-scale setting to manage spikes in metadata expansion and governance tasks without compromising data service [3]. Semantic descriptions of heterogeneous assets must be provided by the governance system, according to research on metadata for data lakes [4]. Whether the data asset may be appropriately comprehended in the business domain is directly determined by the quality of the metadata [5]. Because the downstream data outputs rely on hidden transformation pathways, lineage management is also necessary [6]. Regularly evaluate the data quality following modifications to the data pipeline [7].

Despite the rapid development of the large-data platform, many governance systems still rely on an integrated rule engine and a central metadata store. The system works well for small-scale applications, but as data assets grow to millions, problems including insufficient tenant isolation, unstable rule execution, delayed metadata

synchronization, and lineage reconstruction bottlenecks may arise. The platform's operation and rule design should be integrated rather than carried out separately [8]. The business rules should be converted into implementable quality checks, according to good governance methods [9]. The core data collection must be managed since it serves as a benchmark for cross-system integration [10]. Governance is added to security categorization, authorization, and audit duty through data risk management [11]. Cloud data warehouse migration risk can be decreased with automated metadata governance [12]. According to enterprise governance studies, everyday data operations should incorporate governance tasks [13]. Trusted governance requires clear dependence tracking instead of isolated catalog entries, according to research on the data lineage of business data [14]. Improve the quality of metadata to increase data literacy and decision transparency on a large scale [15].

A cloud-native paradigm for large-scale big data governance is examined in this article. To distribute metadata services, manage metadata lineage, carry out policy-aware rules, and plan elastic governance duties, a governance framework has been presented. It aims to address three issues: the centralized governance service's restricted scalability, poor metadata and lineage update coordination, and erratic execution efficiency under a dynamic governance load. The governance module can function independently as a service that expands in accordance with the extent and stringency of the data-quality regulations and tenant-level governance needs, and the platform is cloud-compatible.

Related Work

Big Data Governance and Data Asset Management

Standards for data, data ownership, data quality, metadata management, data lineage, master data management, risk control, and data value evaluation are all included in research on big data governance. Rules for asset organization, data classification, data life cycle management, and audit accountability are typically included in the entire system of governance. Semantic descriptions and metadata blueprints can enhance discovery and governance automation, according to metadata enrichment strategies for data lakes [16]. In order to comply with regulatory obligations, governance teams must explain how the data is transferred, modified, and used in the enterprise data lake [17]. As a result, lineage and metadata management are closely linked to compliance. Quality indicators for metadata can enhance data literacy and decision-making openness, according to research on the data governance of public policy [18]. Governance services must enable distributed execution for geographically scattered data sources, according to research on edge-cloud big data architecture [19]. The framework for coordinating the dispersed governance components is provided by big data orchestration tools [20]. Workflow orchestration and containerized execution can be coupled, as demonstrated by cloud-native large data processing using Kubernetes and Airflow [21].

Metadata management is the first pillar of the internal Data Governance system. creates a single governance view by linking data asset description, schema version, ownership, storage location, access policy, quality rule, and lineage dependence. Workload elasticity can be enhanced and deployment coupling can be decreased by segmenting into services, according to serverless and microservices research [22]. The scaling mechanism should take security risks and workload factors into account, according to research on Kubernetes autoscaling [23]. The efficiency of cloud execution is impacted by service placement and network overhead, according to communication-aware microservice scheduling [24]. Latency, cost, and resource usage may all be balanced via dynamic multi-objective scheduling for microservices [25]. large metadata systems demonstrate that metadata can grow to the size of large data, necessitating distributed storage and query optimization [26].

Cloud-Native Data Platforms and Elastic Service Architectures

When building data platforms, cloud-native design incorporates microservices, container orchestration, service discovery, declarative deployment, observability, and autonomous scalability. Cloud-native data platforms can be broken down into discrete parts that can be deployed separately for tasks like metadata collecting, rule execution, lineage computation, audit verification, and notification. Service information can facilitate automatic integration in a dispersed context, as demonstrated via REST composition via linked metadata [27]. Cloud-native execution models must be assessed under real-world application workloads, according to research on microservices and serverless performance comparison [28]. The process of healthcare services has also been

studied; as a result, organizations are able to synchronize data from several specializations [29]. Reusable analytics capabilities can be supported by a private-cloud deployment, as shown by Big Data as a Service design [30]. The technical basis for creating a governance system that is both fully functional and scalable in a cloud-native way has been established by the investigations.

Cloud-Native Big Data Governance Framework

Overall Framework and Governance Control Plane

A governance control plane, distributed metadata services, lineage graph service quality rule services, policy enforcement services, audit services, and elastic scheduling services make up the suggested structure. The control plane does not do all governance calculations; instead, it organizes service registration, workflow triggering, policy distribution and execution monitoring, and failure recovery. The general architecture is depicted in Figure 1, where governance functions are implemented as independently scalable containerized services and heterogeneous data assets are connected via metadata adapters.

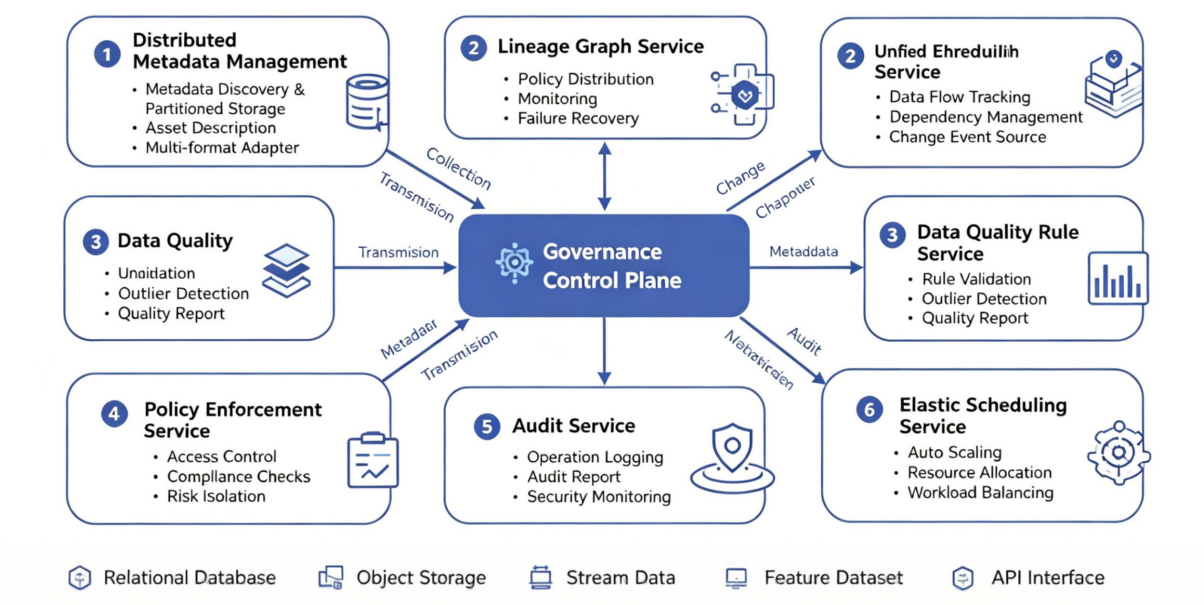

Figure 1. Cloud-native architecture of large-scale big data governance framework

The global governance state is represented as a time-dependent composite object:

$$G(t) = \langle A(t), M(t), L(t), Q(t), P(t), R(t) \rangle \quad \text{Eq.(1)}$$

In this expression, $A(t)$ denotes the active data asset set, $M(t)$ represents distributed metadata states, $L(t)$ descr lineage relationships, $Q(t)$ denotes data quality rule states, $P(t)$ represents governance and access policies, and F describes available runtime resources. This representation treats governance as a continuously changing operationa state instead of a static data catalog.

The control plane evaluates service coordination overhead and governance workload before performing resource adjustment. Its control cost is defined as:

$$C_g(t) = \sum_{i=1}^{n} \eta_i S_i(t) + \sum_{j=1}^{m} \mu_j W_j(t) \quad \text{Eq.(2)}$$

Here, $S_i(t)$ represents the operating state of governance service i, while $W_j(t)$ denotes the pressure generated by governance workload j. The coefficients η_i and μ_j quantify service coordination cost and workload execution cost, respectively. A rapid increase in metadata ingestion or quality inspection pressure therefore produces a measurable control signal for service scaling.

Service availability is evaluated through a governance health score:

$$H(t) = 1 - \frac{|F_m(t)|}{|F_m(t)| + |S_m(t)|} \quad \text{Eq.(3)}$$

The sets $F_m(t)$ and $S_m(t)$ contain failed and successful service interactions within the current monitoring interval. When $H(t)$ falls below 0.92, the control plane initiates replica expansion, isolates abnormal service instances, and reallocates unfinished governance tasks.

An event stream, not sporadic manual synchronization, is the foundation of the Control Plane. Quality services publish rule-execution events, lineage services post dependency events, and metadata adapters publish schema change events. When the data asset is inconsistent with other downstream applications, this event-driven approach can send warnings and won't have outdated information.

Distributed Metadata and Lineage Management Model

Distributed metadata management is based on domain-aware asset partitioning and lineage graph indexing. Each metadata object records its schema, owner, storage location, sensitivity classification, quality rule binding, lineage pointer, policy tag, and version information. Relational tables, object storage files, streaming topics, feature datasets and service interfaces can therefore be represented through a unified metadata structure.

A metadata object is formally expressed as:

$$m_k = \{id_k, \text{type}_k, \text{own}_k, \text{sch}_k, \text{tag}_k, \text{ver}_k\} \quad \text{Eq.(4)}$$

In this model, id_k is the globally unique asset identifier, $type_k$ denotes the asset type, own_k represents the responsible owner, sch_k describes the schema, tag_k contains governance labels, and v_k, records the current metadata version. Version control prevents governance decisions from being made using outdated schema descriptions.

The framework represents dependency between two assets through a directed lineage edge:

$$L_{ab}(t) = \phi(m_a, m_b, u_{ab}, \tau_{ab}) \quad \text{Eq.(5)}$$

In this expression, m_a and m_b denote upstream and downstream metadata objects, u_{ab} represents the transformation operation, and τ_{ab} is the latest update timestamp. A lineage edge is updated when the framework observes data ingestion, transformation, aggregation, feature construction, or service-consumption events.

Frequent lineage changes may indicate unstable processing logic or an incomplete release process. The lineage instability index is defined as:

$$I_l(t) = \sum_{(a,b)\in E} \omega_{ab} \mathbb{I}[L_{ab}(t) \neq L_{ab}(t-1)] \quad \text{Eq.(6)}$$

The coefficient ω_{ab} reflects the importance of each dependency, while the indicator function identifies lineage edges that changed between adjacent governance intervals. High-value reports and machine learning features are assigned larger weights because unexpected changes to their upstream dependencies have greater operational impact.

Metadata quality is evaluated by combining completeness, validity, semantic consistency, and freshness:

$$B_q(k) = \rho_1 c_k + \rho_2 v_k + \rho_3 s_k + \rho_4 r_k \quad \text{Eq.(7)}$$

Here, $c_k, v_k, s_k,$ and r_k represent metadata completeness, validity, semantic consistency, and freshness. The coefficients satisfy $\sum_{i=1}^{4} \rho_i = 1$. Assets with $B_q(k) < 0.80$ are automatically assigned to metadata enrichment or quality inspection workflows.

An incremental graph update method is called lineage management. After a transformation produces a new output asset, just the pertinent edges and downstream effect indices need to be recalculated. Store-level dependencies are recorded by physical lineage, whereas business indicators and semantic derivations are recorded by logical lineage. A business indicator built on several tables, streaming topics, and feature datasets can be made more traceable by using a two-tier structure.

Elastic Governance Task Scheduling Mechanism

Metadata collection, lineage construction, quality evaluation, policy enforcement, compliance inspection, and abnormality notice are the components of the scheduler's governance task. The governance control plane, elastic execution containers, metadata services, and lineage services are all coordinated in Figure 2.

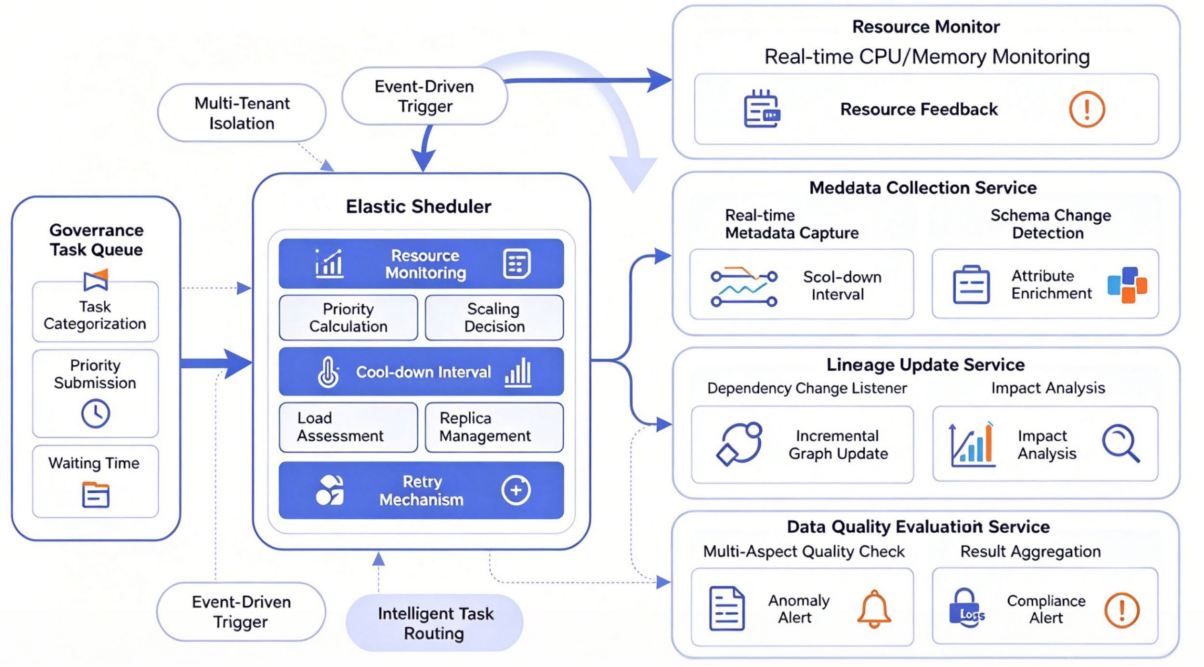

Figure 2. Metadata-lineage coordination and elastic governance task scheduling mechanism

The scheduler first evaluates normalized CPU and memory pressure:

$$U_r(t) = \frac{CPU(t)}{CPU_{\max}} + \frac{MEM(t)}{MEM_{\max}} \qquad \text{Eq.(8)}$$

The variables $\mathrm{CPU}(t)$ and $MEM(t)$ represent current processor and memory utilization, while $CPU_{\max}$ and $MEM_{\max}$ denote available resource limits. When $U_r(t)$ exceeds 1.45 for three consecutive monitoring windows, the scheduler identifies the most congested governance service and initiates replica expansion.

Governance task priority is calculated from asset importance, queue waiting time, and expected governance impact:

$$\Pr(T_i) = a_i d_i + b_i q_i + c_i e_i \qquad \text{Eq.(9)}$$

In this expression, d_i represents the business importance of the target data asset, q_i denotes normalized waiting time, and e_i measures the expected impact of the governance task. Sensitive-data audits, failed policy checks, and lineage reconstruction for high-value data products receive higher priority than routine metadata refresh operations.

The scheduler determines the required service replica count through an adaptive update rule:

$$N_s(t+1) = \lceil N_s(t)[1 + \delta(U_r(t) - \theta_s)] \rceil \qquad \text{Eq.(10)}$$

Here, $N_s(t)$ is the current replica count, θ_s is the service-specific scaling threshold, and δ controls scaling sensitivity. The resulting replica count is additionally restricted by tenant quotas, cluster capacity, and minimum service availability requirements.

After every scaling operation, a cool-down interval is included to prevent the creation and deletion of replicas repeatedly. During this period, the scheduler will continue to distribute the queued jobs, but it is unable to instantly alter the prior scaling choice. A specialized technique is required because metadata bursts are very brief; lineage reconstruction and quality rule execution can run for long periods of time.

The foundation of failure recovery is task semantics. Tasks related to quality inspection and metadata collecting are idempotent and can be safely transferred. Audit tasks have a distinct execution ID for retry traceability, while lineage update activities employ metadata version numbers to avoid adding duplicate edges.

Experimental Scheme and Quantitative Evaluation

Experimental Environment and Dataset Configuration

The experimental platform consists of 12 cloud worker nodes with 48 virtual CPU cores, 192 GB of memory, distributed object storage, a metadata graph service, and a container orchestration layer. 1.2 million data assets, 18.6 million metadata fields, 4.8 million lineage edges, 2,400 data quality rules, 36,000 access policies, and 120 tenant workspaces make up the governance workload.

A composite workload is displayed for the experimental scale:

$$D_s = N_a + N_f + N_l + N_q + N_p \quad \text{Eq.(11)}$$

In this expression, N_a denotes the number of data assets, N_f is the number of metadata fields, N_l represents lineage edges, N_q is the number of quality rules, and N_p denotes governance policies. This definition prevents experimental scale from being measured only by physical data volume.

Relational tables, JSON files, object storage partitions, streaming topics, feature datasets, and service interfaces are among the created assets. Every asset has a quality rule group, storage location, classification label, owner, schema version, and lineage relationship. Missing values, duplicate records, aberrant ranges, schema drift, staleness, and cross-table incompatibilities are examples of quality rules.

A comparison of three systems is made. There is only one rule execution pool and one metadata store in the central baseline. The microservice baseline employs fixed service replicas and divides the governance functions. Tenant quotas, elastic replica scaling, lineage-aware scheduling, and event-driven metadata coordination are all new features of the new framework.

The experiment has been carried out five times. At the start of each step, the cache state is cleared; the initial warm-up phase is not included. In this manner, container startup, initial metadata loading, and temporary cache won't have an impact.

Governance Efficiency and Scalability Metrics

Governance task throughput measures the number of tasks completed within a fixed interval:

$$T_g = \frac{N_{\text{done}}}{\Delta t} \quad \text{Eq.(12)}$$

Here, N_{done} represents successfully completed metadata, quality, lineage, and audit tasks during interval Δt. Failed retries are not counted as completed tasks. This metric reflects whether the governance platform can keep pace with metadata growth and rule execution demand.

Average metadata query latency is calculated as:

$$L_m = \frac{1}{N_{\text{query}}} \sum_{i=1}^{N_{\text{query}}} t_i^{\text{query}} \quad \text{Eq.(13)}$$

The variable t_i^{query} denotes the response time of query i, while N_{query} is the total number of metadata requests. The evaluation also records 95th percentile latency because a low average value may conceal unstable tail latency during workload bursts.

Elastic scaling efficiency measures the throughput improvement obtained per additional resource unit:

$$E_s = \frac{T_{\text{after}} - T_{\text{before}}}{R_{\text{after}} - R_{\text{before}}} \quad \text{Eq.(14)}$$

The variables T_{before} and T_{after} denote task throughput before and after scaling, while R_{before} and R_{after} represent allocated resource units. A high value indicates that additional replicas produce meaningful governance capacity rather than merely consuming more resources.

Other evaluation indicators include lineage update delay, quality rule completion rate, anomaly detection recall, tenant-level governance delay, quota violation frequency, replica occupation ratio, CPU utilization, and memory utilization. These metrics jointly evaluate governance efficiency, elasticity, accuracy, and multi-tenant stability.

Comparative Experimental Design

Below are the three stages of the workload. Steady metadata queries, lineage retrieval, and quality rule execution under typical governance demands are tested in the first step. The second is brought on by a metadata explosion brought on by the extensive asset movement and schema registration. The third stage involves the simultaneous generation of quality inspection, lineage update, and compliance audit tasks for several tenants.

Approximately 8,000 metadata requests and 300 concurrent quality rule executions occur every minute in the steady state. During the metadata burst stage, 240,000 schema objects are registered in 15 minutes. 120 tenant workspaces send 1,200 quality rules, 180,000 lineage updates, and 32,000 policy checks during the mixed burst stage.

The distribution of data assets, request order, task priority input, and cluster resource limit are all the same across all comparison systems. The microservice baseline makes use of the same functional services as the suggested framework, and the central system is not allowed to add additional hardware. Thus, distributed metadata organization, lineage-aware task dispatch, and elastic scheduling are the primary causes of the observed variations.

Result Interpretation and Comparative Analysis

Governance Task Throughput Analysis

The throughput comparison shows that with a stable workload, centralized governance deployment reaches 8,600 tasks per minute; after a quality rule surge, it falls to 5,200 tasks per minute. Although the microservice baseline raises the steady-state throughput to 12,700 tasks per minute, queue accumulation is still evident when lineage and quality jobs arrive simultaneously. Because the metadata collection, rule execution, and lineage update services are separately scalable, the framework can manage about 18,900 tasks per minute under a steady demand and even 16,400 tasks per minute during a peak load. Automatic metadata processing can lessen the amount of effort required for manual governance, according to studies on AI-assisted metadata conversion [31]. The governance task throughput and execution stability are compared in Figure 3. The average throughput under the three workload phases is shown in Figure 3(a); the fluctuation in queue length during governance bursts is shown in Figure 3(b); and the task completion stability for tenant workspaces is shown in Figure 3(c). An organized dependency update system is beneficial for warehouse administration, according to research on the dissemination of information lineage [32].

There are other factors contributing to the increased throughput besides the rise in service replicas. To cut down on unnecessary information reads, the framework caches commonly used asset descriptors close to the rule execution service. Additionally, it may separate the quick metadata validation effort from the lengthy lineage reconstruction; otherwise, the short task would have to wait for the costly graph operation. The central system's mean queue length during the burst period was 18,400 jobs; following scaling with the suggested framework, this figure dropped to less than 6,700. As a result, the extended schedule will boost processing power and expedite queue recovery.

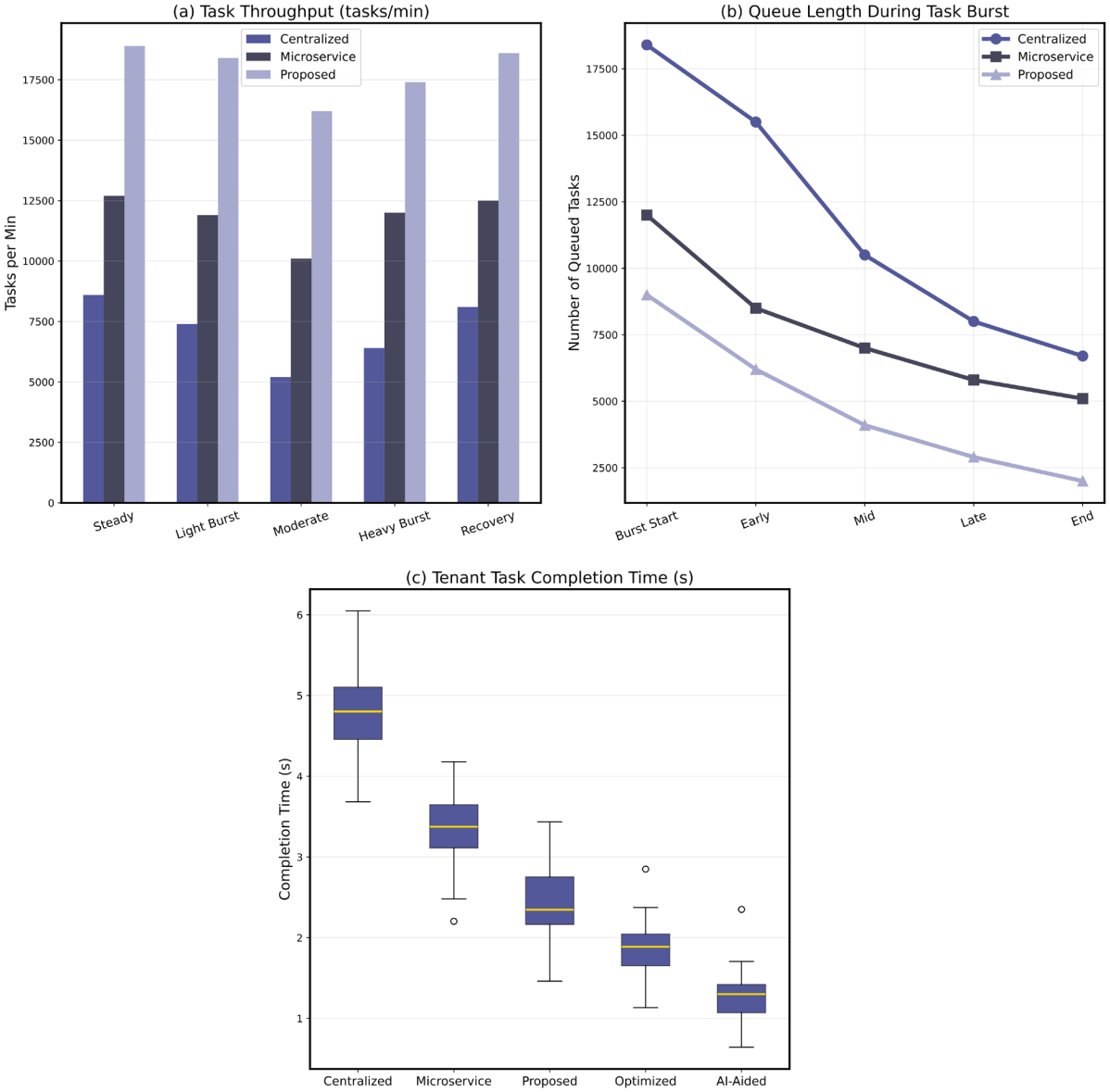

Figure 3. Governance task throughput and execution stability comparison. (a) Average governance task throughput under different workload stages. (b) Queue length variation during governance task bursts. (c) Task completion stability across tenant workspaces

Metadata Query and Lineage Tracking Performance

The latency of metadata queries is reduced from 146 MS in the centralized baseline to 77 MS in the microservice baseline and 53 MS in the suggested framework. Reduce latency by using freshness-based indexing, replica-aware routing, and information partitioning. The suggested approach has maintained the 95th percentile latency below 118 MS with less than 18.6 million metadata fields; the centralized baseline reached 312 Ms. The reliability of the subsequent governance is directly impacted by the quality assurance of intake, according to research on data integrity [33]. A comparison of lineage tracking efficiency and metadata query latency is shown in Figure 4. The 95th percentile latency of concurrent metadata requests is displayed in Figure 4(b), the average latency under various asset scales is shown in Figure 4(a), and the lineage update delay is compared as the number of lineage edges increases in Figure 4(c). The alignment of the Catalog Structure and Metadata Use Cases is also necessary, according to research on the Management of Data Lake Metadata [34].

The same pattern can be seen in lineage tracing. The centralized baseline indicates a delay increase of 0.84 s to 3.76 s for medium-depth dependency traversal as the number of lineage edges rises from 1.2 million to 4.8 million. Because lineage partitions and incremental update indexes limit the scope of graph traversal, the suggested framework has grown from 0.42 s to 1.31 s. For the impact test, this will be more appropriate. The

governance system should quickly locate all relevant reports, machine-learning features, downstream tables, and service interfaces when the source schema is modified. Operational hazards may rise and release permission will be delayed if the lineage responds slowly.

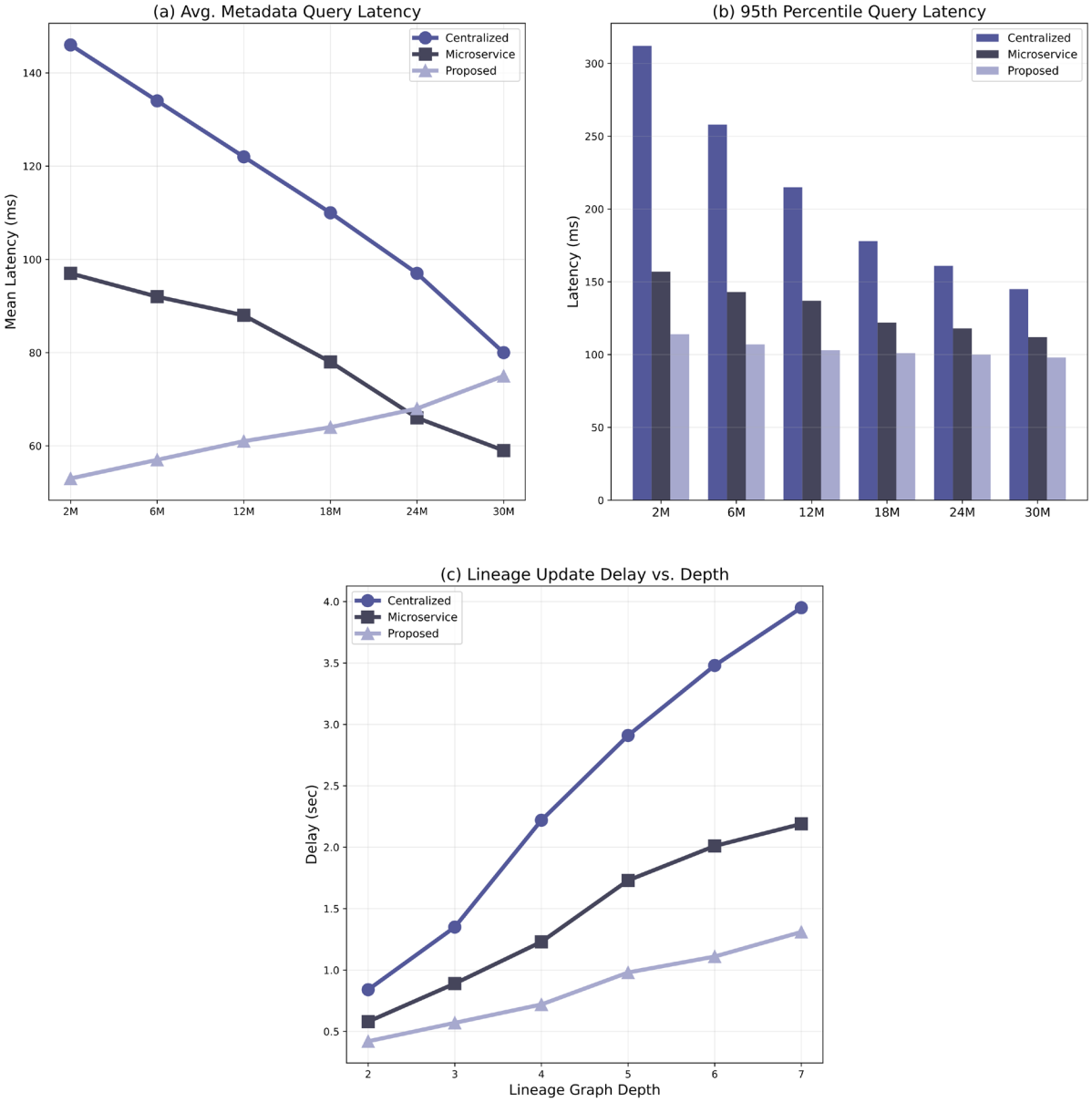

Figure 4. Metadata query latency and lineage tracking efficiency comparison. (a) Average metadata query latency under different asset scales. (b) 95th percentile latency under concurrent metadata requests. (c) Lineage update delay under different lineage edge counts

The suggested structure further enhances the execution of the Data Quality Rule. 62.8% of the 2,400 quality standards in the target window are covered by the centralized baseline, 78.5% by the microservice baseline, and 93.6% by the suggested framework. Metadata lineage coordination can minimize dependency traversals and redundant schema lookups, and it is more suited for cross-domain rules. Kubernetes-based services can adjust to shifts in workload by scaling in accordance with predefined rules, according to research on AI-driven autoscaling [35]. The data quality rule execution and anomaly detection performance are shown in Figure 5. The rule completion rate is shown in Figure 5(a), the anomaly detection recalls for cases including missing values, duplicate records, and schema drift is shown in Figure 5(b), and the rule execution time under tenant-level workload isolation is shown in Figure 5(c). Replica-oriented operation for cloud-native services is supported by Kubernetes management research [36].

The findings indicate a positive correlation between metadata completeness and governance quality. Due to missing key-field definitions, the recall of duplicate-record detection will decrease by 8.4 percentage points if the metadata quality baseline is less than 0.70. The recall can be raised from 86.1% to 94.7% by adding metadata and rebinding rules. Because lineage-aware execution enhances the effectiveness of schema drift detection, upstream modifications to the quality rules can spread to downstream inspection batches prior to the start of the subsequent batch. The research suggests that metadata freshness and lineage event processing should be added to the rule engine for data quality control.

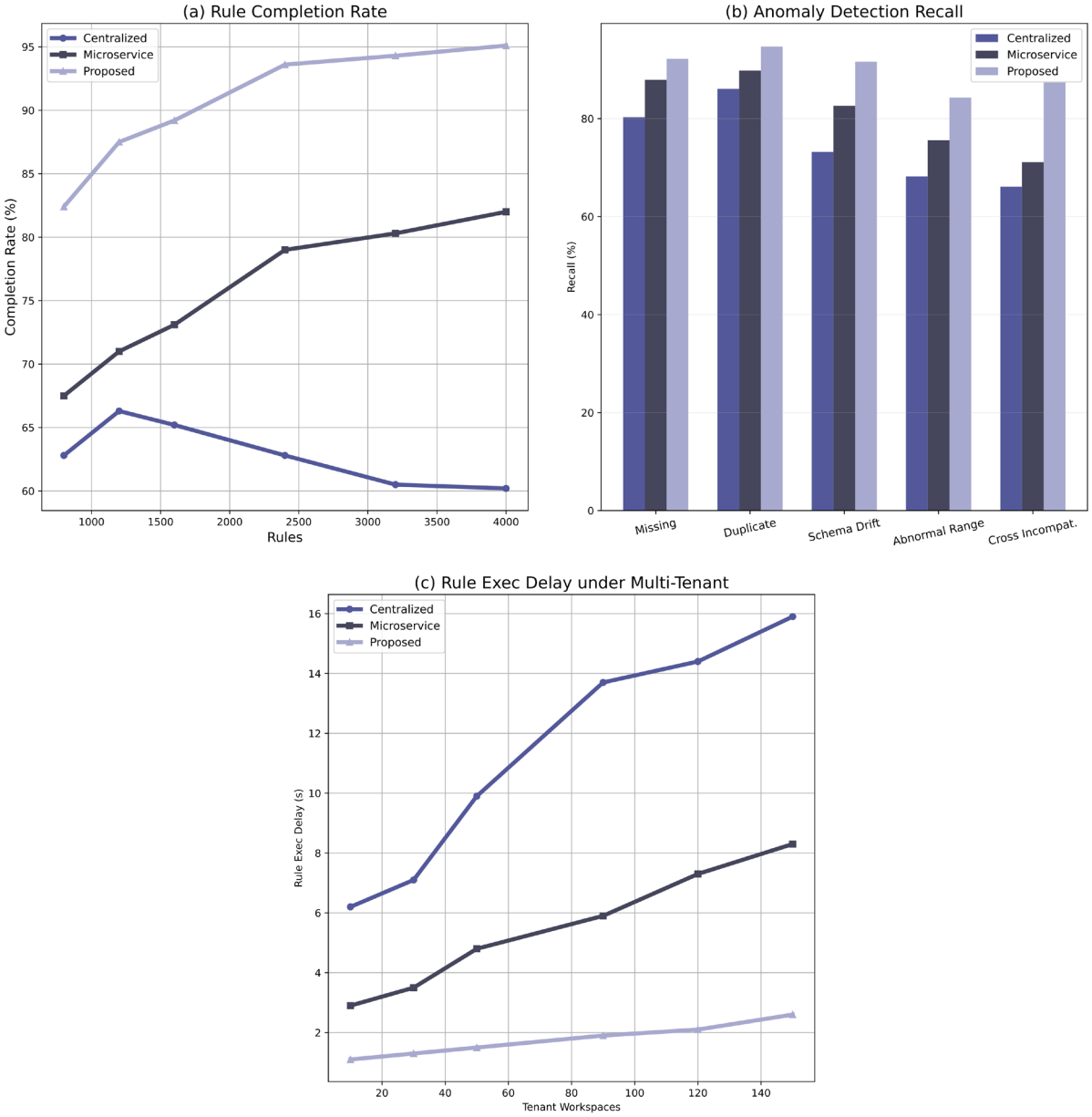

Figure 5. Data quality rule execution and anomaly detection performance. (a) Quality rule completion rate under different rule volumes. (b) Anomaly detection recall for missing-value, duplicate-record, and schema-drift cases. (c) Rule execution delay under tenant-level workload isolation

Elastic Scaling and Multi-Tenant Governance Discussion

The suggested framework raises metadata service replicas from 6 to 14 within 85 seconds when the workload pressure increases, and quality rule service replicas increase from 8 to 20 within 110 seconds, according to Elastic Scaling Experiments. After stabilization, the CPU utilization is now between 63.5% and 78.2%, which is comparatively balanced when compared to the microservice baseline of 42.1%–91.7%. According to research

on cloud microservices security, policy enforcement and governance-aware isolation are necessary for the distributed service [37]. The elastic scaling performance under various governance demands is shown in Figure 6. Replica changes during metadata bursts are depicted in Figure 6(a); CPU and memory utilization following scaling is compared in Figure 6(b); and the scaling efficiency of overlapping quality inspection and lineage updating jobs is assessed in Figure 6(c). Additionally, serverless and containerized components must be safeguarded by consistent policy management, according to cloud-native security research [38].

The scaling curve illustrates the tension between resource utilization and responsiveness. If the scaling threshold is set too low, there won't be any improvement in completion time and redundant copies will be made for brief bursts of metadata. The quality rule queue will build up and the audit findings will be delayed if the threshold is set too high. For the first two burst windows, the chosen threshold keeps scaling efficiency above 0.71 and ensures consistent replica production. As a result, the general web-service autoscaling strategy should not be applied to the governance workload; instead, it should be scaled independently.

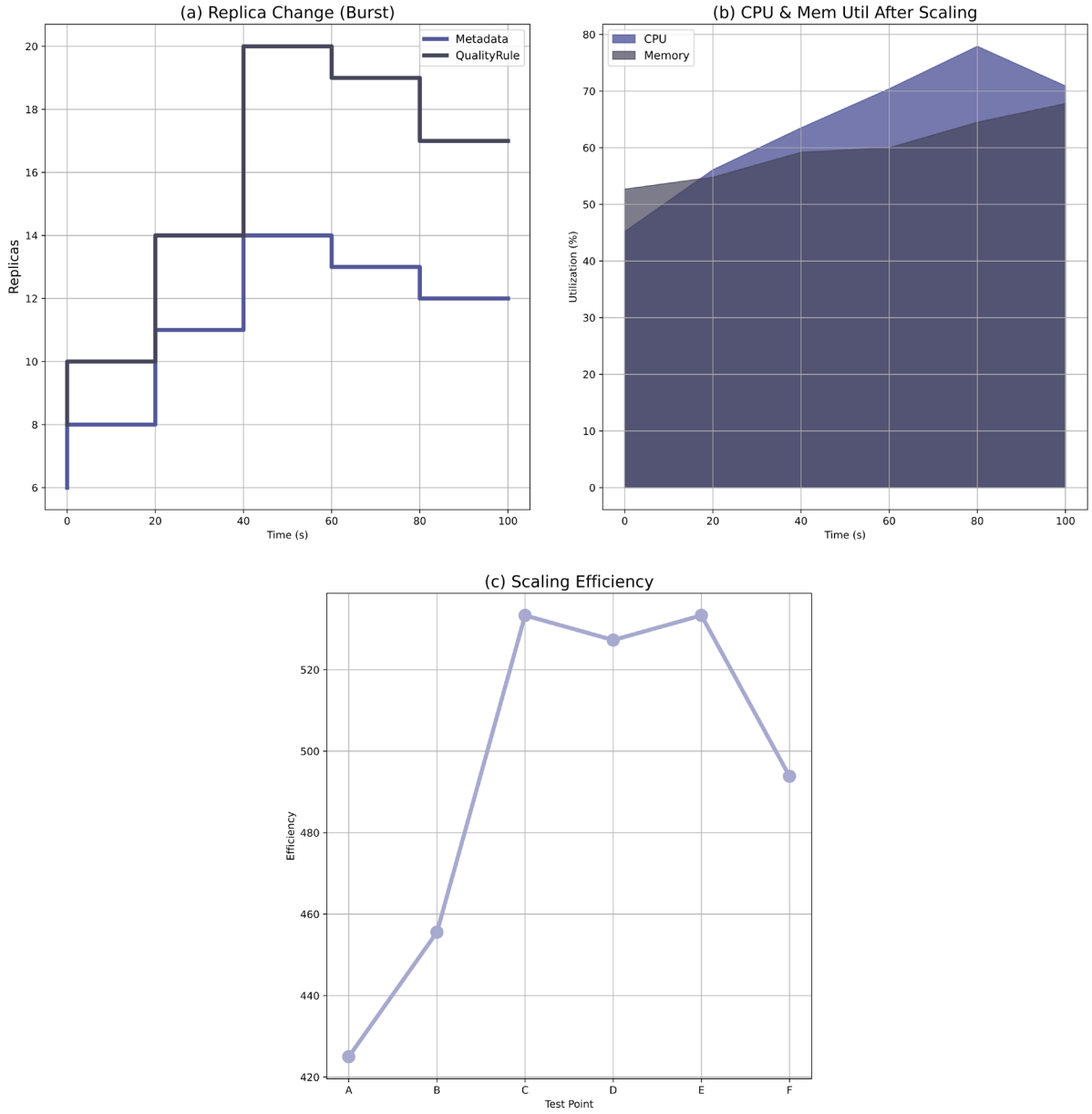

Figure 6. Elastic scaling performance under different governance workloads. (a) Replica change during metadata burst workloads. (b) CPU and memory utilization after elastic scaling. (c) Scaling efficiency under overlapping quality and lineage tasks

The findings of multi-tenant governance demonstrate that the suggested framework outperforms the baselines in maintaining tenant isolation. The average tenant-level governance delay under 120 tenant workspaces is 1.42

seconds, which is better than the microservice baseline's 2.36 seconds and the centralized baseline's 4.91 seconds. By using tenant quotas in task scheduling and setting aside protected execution windows for high-priority audit tasks, you can limit resource interference. The governance value is predicated on steady operation and quantifiable risk reduction, according to data governance business-case study [39]. Multi-tenant governance isolation and resource utilization comparison are displayed in Figure 7. Tenant delay distribution is shown in Figure 7(a), quota violation frequency is shown in Figure 7(b), and resource utilization fairness is compared in Figure 7(c). The need for reusable governance capabilities that may be used across various domains and tenants is supported by studies on data governance capability [40].

Tenant-level quota control seeks to lessen the influence of major tenants based on the fairness analysis. Metadata and audit requests from other domains won't be blocked in the same manner. Because all tenants use the same execution pool, the frequency of quota violations during concurrent governance workload in the centralized baseline is 7.8%. The suggested approach reduces the variance of tenant-level delay by 54.2% and the incidence of quota violations to 1.6%. This outcome can be used to support the governance platform of a company that has different service-level agreements and regulatory requirements but employs a single data infrastructure across all business divisions.

The architecture will be appropriate for a variety of governance tasks given the state of technology. Through indexing and caching, a central system can still satisfy the speed criteria if the sole type of work is a straightforward metadata lookup. This location offers the advantages of quality inspections, compliance audits, lineage updates, and metadata inquiries. A fixed deployment cannot satisfy service quality standards since different types of jobs have variable execution times, resource requirements, and priority due to the various loads. With elastic service capacity, a cloud-native architecture may effectively fulfill the demands of a governance assignment.

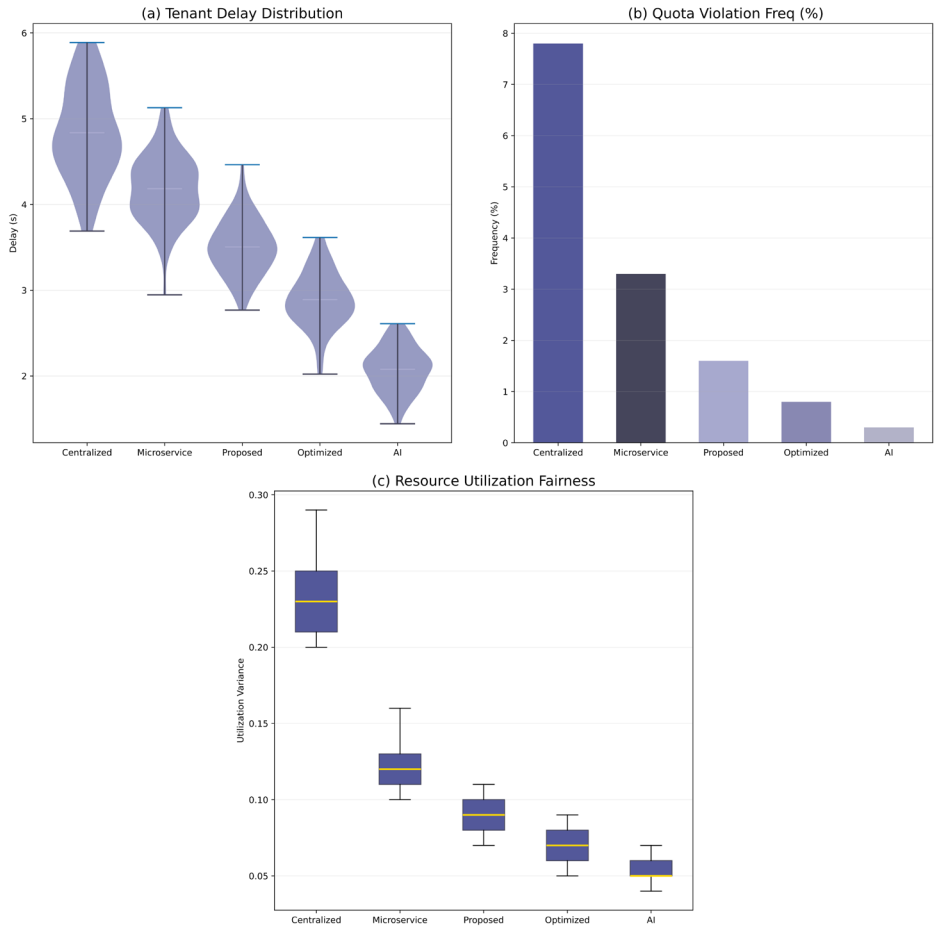

Figure 7. Multi-tenant governance isolation and resource utilization comparison. (a) Tenant-level governance delay distribution. (b) Quota violation frequency under concurrent workloads. (c) Resource utilization fairness across tenant workspaces

Conclusion

This study proposed a cloud-native framework for large-scale big data governance. A governance control plane, distributed metadata services, lineage graph coordination, data quality rule execution, policy-oriented governance services, and elastic task scheduling are the framework's constituent parts. In summary, the studies show that cloud-native decomposition can change governance from a centralized cataloging mode to a platform capability that is scalable, service-oriented, and always operational.

To improve governance performance, incorporate workload-aware scalability, quality checks, information partitioning, and lineage updates into the new framework. According to the experiments, the average metadata query latency has decreased by 47.3%, the governance task throughput has increased from 8,600 to 18,900 tasks per minute, and the elastic scaling efficiency is 31.6% greater than that of central deployment. The findings suggest that metadata and governance tasks should be handled as dynamic cloud workloads as opposed to fixed administrative data.

It is a great platform for a lot of data from various sources that may satisfy requirements for lineage tracking, multi-tenant governance separation, frequent metadata changes, and data-quality monitoring. Research can be done in the future to combine hardware-aware scheduling, automated policy formulation, knowledge-graph-based semantic governance, and privacy-preserving computing. Optimize the governance workload model and improve scheduling flexibility based on the real production data in the organization's data platform.

Author Contributions

Zuzanna Chmielewska contributes to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision. Teresa Zającowa contributes to data collection, draft preparation, manuscript editing. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

References

[1] Automated Metadata Governance Frameworks for Large-Scale Cloud Data Warehouse Migrations. (2020). International Journal of AI, BigData, Computational and Management Studies. https://doi.org/10.63282/3050-9416.ijaibdcms-v1i1p105

[2] Goniwada, S. R. (2021). Cloud native data architecture. In Cloud Native Architecture and Design. Apress. https://doi.org/10.1007/978-1-4842-7226-8_8

[3] Gutierrez, F. (2020). Cloud and big data. In Spring Cloud Data Flow. Apress. https://doi.org/10.1007/978-1-4842-1239-4_1

[4] Zgolli, H., Collet, C., & Madera, C. (2020). Metadata in data lake ecosystems. In Data Lakes. Wiley. https://doi.org/10.1002/9781119720430.ch4

[5] Elouataoui, W., El Alaoui, I., & Gahi, Y. (2021). Metadata quality dimensions for big data use cases. Proceedings of the 2nd International Conference on Big Data, Modelling and Machine Learning. https://doi.org/10.5220/0010737400003101

[6] Karkoskova, S., & Novotny, O. (2021). Design and application on business data lineage as a part of metadata management. 2021 International Conference on Computers and Automation. https://doi.org/10.1109/compauto54408.2021.00014

[7] Srivastava, D. (2020). Towards high-quality big data: Lessons from FIT. 2020 IEEE International Conference on Big Data. https://doi.org/10.1109/bigdata50022.2020.9378181

[8] Chapter 6 Big Data Governance Framework. (2020). In Big Data Management. De Gruyter. https://doi.org/10.1515/9783110664065-006

[9] Chapter 9 Big Data Governance Best Practices. (2020). In Big Data Management. De Gruyter. https://doi.org/10.1515/9783110664065-009

[10] Chapter 7 Master Data Management. (2020). In Big Data Management. De Gruyter. https://doi.org/10.1515/9783110664065-007

[11] Chapter 3 Data Risk Management. (2020). In Big Data Management. De Gruyter. https://doi.org/10.1515/9783110664065-003

[12] Kumar, A. (2022). An AI-driven framework for data governance, quality management, and metadata integration in enterprise systems. International Journal of Artificial Intelligence, Data Science, and Machine Learning. https://doi.org/10.63282/3050-9262.ijaidsml-v3i2p118

[13] Mahanti, R. (2021). Data governance and data management functions and initiatives. In Data Governance and Data Management. Springer. https://doi.org/10.1007/978-981-16-3583-0_3

[14] Neri, C. (2019). Lineage metadata as a critical component of data trustworthiness for subsurface and analytics applications. 2019 AAPG Annual Convention and Exhibition. https://doi.org/10.1306/42408neri2019

[15] Edara, P., & Pasumansky, M. (2021). Big metadata. Proceedings of the VLDB Endowment. https://doi.org/10.14778/3476311.3476385

[16] Pingos, S., & Andreou, A. S. (2022). A data lake metadata enrichment mechanism via semantic blueprints. Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering. https://doi.org/10.5220/0011080400003176

[17] Data Lineage and Metadata Management in Enterprise Data Lakes: Tools, Frameworks, and Compliance Implications. (2020). International Journal of Innovative Research and Creative Technology. https://doi.org/10.62970/ijirct.v6.i4.2601003

[18] Barbosa, J., Planas, C., & Francisco, M. (2022). Framework for public health policy indicators governance and metadata quality flags to promote data literacy. Proceedings of the 11th International Conference on Data Science, Technology and Applications. https://doi.org/10.5220/0011259000003269

[19] Inibhunu, C., & McGregor, C. (2020). Edge computing with big data cloud architecture: A case study in smart building. 2020 IEEE International Conference on Big Data. https://doi.org/10.1109/bigdata50022.2020.9377918

[20] Garg, R., Wang, H., & Ranjan, R. (2019). Orchestration tools for big data. In Encyclopedia of Big Data Technologies. Springer. https://doi.org/10.1007/978-3-319-77525-8_43

[21] Lekkala, S. (2021). Implementing cloud-native technologies for big data processing: A case study with Kubernetes and Airflow. International Journal of Science and Research. https://doi.org/10.21275/sr24430152128

[22] Kratzke, N. (2021). Microservice und serverless-architekturen. In Cloud-native Computing. Hanser. https://doi.org/10.3139/9783446472846.012

[23] Ben David, Y., & Barr, J. (2021). Kubernetes autoscaling: YoYo attack vulnerability and mitigation. Proceedings of the 11th International Conference on Cloud Computing and Services Science. https://doi.org/10.5220/0010397900340044

[24] Marchese, A., & Tomarchio, O. (2022). Communication aware scheduling of microservices-based applications on Kubernetes clusters. Proceedings of the 12th International Conference on Cloud Computing and Services Science. https://doi.org/10.5220/0011049300003200

[25] Fard, H. M., Prodan, R., & Wolf, F. (2020). Dynamic multi-objective scheduling of microservices in the cloud. 2020 IEEE/ACM International Conference on Utility and Cloud Computing. https://doi.org/10.1109/ucc48980.2020.00061

[26] Serrano, D., & Stroulia, E. (2021). The LRA Workbench: An IDE for efficient REST API composition through linked metadata. Journal of Big Data, 8, 117. https://doi.org/10.1186/s40537-021-00504-z

[27] Fan, C., Jindal, A., & Gerndt, M. (2020). Microservices vs serverless: A performance comparison on a cloud-native web application. Proceedings of the 10th International Conference on Cloud Computing and Services Science. https://doi.org/10.5220/0009792702040215

[28] Fiaidhi, J., Mohammed, S., & Fong, S. (2020). Orchestration of thick data analytics based on conversational workflows in healthcare community of practice. 2020 IEEE International Conference on Big Data. https://doi.org/10.1109/bigdata50022.2020.9377848

[29] Zheng, L. (2021). Technical architecture of big data cloud platform for intelligent washing factory. 2021 4th International Conference on Artificial Intelligence and Big Data. https://doi.org/10.1109/icaibd51990.2021.9459049

[30] Reis, J., & Favacho de Araújo, A. (2019). ArchaDIA: An architecture for big data as a service in private cloud. Proceedings of the 9th International Conference on Cloud Computing and Services Science. https://doi.org/10.5220/0007787801870197

[31] Proctor, P., & Marciano, R. (2021). An AI-assisted framework for rapid conversion of descriptive photo metadata into linked data. 2021 IEEE International Conference on Big Data. https://doi.org/10.1109/bigdata52589.2021.9671715

[32] Saputra, R., Purnomo, H., & Setiawan, N. (2022). Sistem propagasi anotasi pada metadata lineage untuk manajemen data warehouse. Jurnal Teknologi Informasi dan Ilmu Komputer. https://doi.org/10.25126/jtiik.2022976833

[33] Sandhya Kona, S. (2020). Ensuring data integrity in big data ingestion: Techniques and best practices for data quality assurance. International Journal of Science and Research. https://doi.org/10.21275/sr24522140238

[34] Megdiche, I., Ravat, F., & Zhao, Y. (2020). A use case of data lake metadata management. In Data Lakes. Wiley. https://doi.org/10.1002/9781119720430.ch5

[35] Panyala, A. (2022). Integrating AI-driven autoscaling mechanisms in Kubernetes-based microservices architectures. International Journal of Advanced Research in Education. https://doi.org/10.34218/ijare_05_01_002

[36] Bartlett, J. (2022). Basic Kubernetes management. In Cloud Native Applications with Docker and Kubernetes. Apress. https://doi.org/10.1007/978-1-4842-8876-4_9

[37] Rajasekharaiah, K. M. (2020). Securing microservices on cloud. In Cloud-Based Microservices. Apress. https://doi.org/10.1007/978-1-4842-6564-2_9

[38] Jitendrakumar Kanani, P., & Sridhar, R. (2020). Cloud-native security: Securing serverless architectures. International Journal of Science and Research. https://doi.org/10.21275/ms2008134043

[39] Ladley, J. (2020). The data governance business case. In Data Governance. Academic Press. https://doi.org/10.1016/b978-0-12-815831-9.00005-9

[40] Data governance capabilities and functions. (2020). In Data Governance. Academic Press. https://doi.org/10.1016/b978-0-12-815831-9.09972-0