World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

Distributed data engineering: The backbone of modern data ecosystems

Srinivas Lakkireddy *

Sri Venkateswara University, India.

World Journal of Advanced Research and Reviews, 2025, 26(02), 3288-3295

Publication history: Received on 04 April 2025; revised on 20 May 2025; accepted on 22 May 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.2.2002

Abstract

This article examines the evolving landscape of distributed data engineering and its critical role in modern enterprise data architectures. As organizations face unprecedented challenges in processing escalating volumes of data across diverse sources, traditional centralized approaches have proven insufficient. Distributed data engineering has emerged as a foundational discipline that enables scalable, fault-tolerant data processing across multiple interconnected computing resources. The article explores how parallel computing frameworks like Apache Spark, Flink, and Dask provide the technical foundation for this paradigm shift, enabling high availability, resilience, and optimized resource utilization. It traces the evolution from batch processing to real-time streaming architectures and examines key technical challenges including data consistency, latency optimization, workflow orchestration, and cost management. The article further investigates emerging paradigms shaping the future of distributed data engineering, including data mesh architectures, AI/ML integration, edge computing, and serverless data processing. These converging trends are creating new possibilities for distributed intelligence that span from edge devices to cloud infrastructure, fundamentally transforming how organizations derive value from their data assets while requiring significant organizational and technological adaptations.

Keywords: Distributed Data Processing; Data Mesh; Edge Computing; Real-Time Analytics; Serverless Architectures

1. Introduction

In today's data-driven world, organizations face unprecedented challenges in processing and analyzing massive volumes of information. The traditional centralized data processing approaches are no longer sufficient to handle the scale, velocity, and variety of data generated across industries. Research published in the Scientific Research Publishing journal indicates that data volume worldwide is increasing at a staggering rate daily, with unstructured data accounting for the majority of all enterprise data [1]. This exponential growth has created an urgent need for more sophisticated approaches to data management and processing. The emergence of Industry 4.0 has further accelerated this trend, with Internet of Things (IoT) deployments generating continuous streams of data that require real-time processing capabilities. Companies utilizing advanced distributed processing frameworks have demonstrated significant ability to reduce time-to-insight compared to traditional data processing methods [1]. Distributed Data Engineering has emerged as a critical discipline enabling businesses to harness the full potential of their data assets in this challenging landscape.

2. Understanding Distributed Data Engineering

Distributed Data Engineering is the practice of designing, developing, and maintaining scalable and fault-tolerant data processing systems that operate across multiple interconnected computing resources. These resources span a wide spectrum, including on-premise servers, virtual machines, Kubernetes clusters, and cloud-based instances. According to research on big data architectures, the proliferation of cloud computing has transformed the distributed computing

* Corresponding author: Srinivas Lakkireddy

landscape, with enterprise cloud adoption reaching near-ubiquity in recent years [1]. This transformation has been driven by the need for elastic computing resources that can scale with fluctuating data processing demands.

Table 1 Distributed Computing Frameworks [1]

Framework	Processing Model	Key Strengths	Primary Applications
Hadoop	Batch	Scalability, Fault tolerance	Large-scale data processing
Spark	Batch, Micro-batch	In-memory processing	ML, Interactive queries
Flink	Stream	True streaming, Stateful processing	Event processing
Dask	Batch	Python ecosystem integration	Scientific computing, ML

At its foundation, this discipline leverages parallel computing frameworks such as Apache Spark, Flink, and Dask to execute computations across multiple nodes simultaneously. Research on data lake implementations has shown that organizations implementing modern distributed architectures experience substantial reduction in total cost of ownership compared to traditional data warehousing approaches [2]. The concept of Fast Data has emerged alongside Big Data, with studies demonstrating that real-time processing systems can now handle substantial data velocities in production environments. The implementation of data lake architectures provides significant advantages, including the ability to store and process both structured and unstructured data in their native formats, eliminating the overhead typically associated with ETL processes in traditional data warehouses [2].

These distributed approaches ensure high availability, with leading distributed systems achieving impressive uptime metrics. They also provide resilience, with workloads able to recover from failures without data loss through advanced fault-tolerance mechanisms that implement redundancy measures reducing data loss probability. Furthermore, they optimize resource utilization, with research demonstrating that proper implementation of distributed data lakes can improve CPU utilization compared to traditional architectures [2].

3. The Evolution from Batch to Real-Time Processing

Historically, data processing systems operated in batch mode, where data was collected over a period and processed at scheduled intervals. While effective for certain use cases, this approach created significant latency between data generation and insight delivery. Studies of traditional batch processing systems reveal lengthy average processing cycles, with some complex batch workloads requiring substantial time in extreme cases [3]. This latency proved increasingly problematic as business decision cycles accelerated in the digital economy.

Modern distributed data architectures have evolved to support real-time streaming, enabling organizations to process and analyze data as it arrives. Apache Spark, a leading distributed processing framework, has demonstrated significant performance improvements over traditional Hadoop MapReduce for in-memory operations, with documented benchmarks showing the ability to process substantial volumes of data across multi-node clusters [3]. The framework's micro-batch architecture allows for minimal processing latencies while maintaining exact-once processing guarantees. This technological advancement has enabled entirely new classes of applications that were previously impossible with batch processing. Research across numerous organizations implementing Spark revealed that most achieved positive ROI within a year of deployment, with average data processing costs decreasing compared to previous solutions [3].

Table 2 Evolution of Data Processing Paradigms [3]

Paradigm	Typical Latency	Key Technologies	Primary Use Cases
Batch	Hours to days	ETL tools, Data warehouses	Reporting, Historical analysis
Micro-batch	Minutes to hours	Hadoop, MapReduce	Big data analytics
Real-time	Seconds	Apache Spark, Flink	Fraud detection, Personalization
Streaming	Milliseconds	Kafka Streams, Flink	IoT analytics, Algorithmic trading

This shift to real-time processing has been transformative across numerous industries. In the financial sector, real-time fraud detection systems leveraging distributed stream processing have demonstrated the ability to identify suspicious transactions with high accuracy while processing many transactions per second, resulting in a significant reduction in fraud losses. Healthcare organizations implementing real-time analytics for patient monitoring have documented improved patient outcomes in critical care environments and reduced hospital readmission rates. E-commerce platforms utilizing distributed recommendation engines have increased conversion rates through personalization that adapts within seconds to changing customer behavior. Manufacturing facilities employing IoT sensor networks with real-time analytics capabilities have reduced unplanned downtime and lowered maintenance costs through predictive maintenance capabilities [4].

4. Data Consistency and Integrity Challenges

Ensuring ACID (Atomicity, Consistency, Isolation, Durability) compliance across distributed systems presents significant technical challenges. When data is processed across multiple nodes, maintaining a consistent state requires sophisticated coordination mechanisms. Research indicates that implementing strong consistency guarantees can substantially increase latency compared to eventual consistency models, creating a fundamental trade-off between consistency and performance [4]. This trade-off has led to the development of various consistency models such as eventual consistency, causal consistency, and session consistency, each offering different guarantees suitable for different application requirements.

The challenge of handling out-of-order event processing in streaming systems represents another critical concern. Studies of production streaming systems show that a non-trivial percentage of events arrive out of order, with delays ranging from milliseconds to minutes depending on source systems and network conditions [1]. The implementation of event-time processing with watermarking techniques has emerged as a standard approach to address this challenge, though it introduces additional complexity in pipeline design and monitoring. Analysis of large-scale distributed data systems found that data consistency issues accounted for a significant proportion of production incidents, highlighting the critical importance of robust consistency mechanisms [1].

Table 3 Data Consistency Models [4]

Model	Description	Trade-offs	When to Use
Strong	All nodes see same data simultaneously	High latency, Low availability	Financial transactions
Eventual	System becomes consistent over time	Low latency, High availability	Content delivery
Causal	Related operations appear in same order	Moderate latency	Messaging applications
Session	Guarantees within user session only	Good performance	E-commerce carts

5. Latency Optimization Strategies

Real-time applications demand minimal processing delays throughout the data pipeline. Comprehensive research across many distributed systems implementations has documented that optimized distributed architectures can achieve low end-to-end latencies for simple analytics and somewhat higher but still rapid latencies for complex processing workloads [3]. Achieving these performance levels requires systematic optimization at multiple layers of the data architecture.

Strategic data partitioning represents a foundational optimization technique, with research demonstrating that well-designed partitioning strategies can significantly reduce query latency by minimizing data movement across the network and enabling higher parallelism. Memory optimization through techniques such as columnar storage formats and data compression has been shown to substantially reduce storage requirements while simultaneously improving query performance for analytical workloads. Network topology improvements, particularly the implementation of data locality principles that place computation near data, have demonstrated the ability to lower data transfer times in geographically distributed systems. Query optimization through techniques such as predicate pushdown, cost-based optimization, and query rewriting has yielded documented performance improvements for complex analytical workloads [3].

6. Orchestration of Distributed Workflows

Managing complex workflows across distributed environments requires sophisticated orchestration tools and methodologies. The distributed nature of modern data pipelines introduces numerous failure points and complex dependencies that must be carefully managed. Analysis of data pipeline failures across numerous organizations revealed that a majority of incidents were directly attributable to orchestration issues, including dependency management failures, resource contention, and timing problems [4].

Platforms like Apache Airflow, Dagster, and Prefect have emerged as industry standards for workflow orchestration, allowing engineers to define, schedule, and monitor data pipelines as directed acyclic graphs (DAGs). These tools ensure proper execution dependencies and error handling through declarative pipeline definitions. The adoption of workflow orchestration tools has grown dramatically in recent years [4].

Advanced orchestration techniques implemented through these platforms have been shown to reduce pipeline failures and decrease end-to-end execution times through improved parallelization and resource allocation. Organizations implementing mature orchestration practices report significant reduction in operational costs associated with data pipeline management [1]. These improvements stem from capabilities such as automatic retry mechanisms, dynamic resource allocation, and sophisticated monitoring that enables proactive intervention before failures impact downstream systems.

7. Cost Optimization in Distributed Environments

Cloud-native distributed systems introduce complex cost considerations that must be carefully managed to ensure economic sustainability. Research across many enterprises implementing cloud-based data platforms revealed substantial cost overruns in the first year of operation, primarily attributed to inefficient resource provisioning and data management practices [4]. Effective cost optimization requires a multi-faceted approach addressing compute, storage, and operational dimensions.

Automated scaling based on workload represents a primary cost optimization strategy, with case studies demonstrating potential compute cost reductions compared to static provisioning. Data lifecycle management implementing tiered storage strategies has been shown to cut storage expenses in large-scale deployments by automatically moving infrequently accessed data to lower-cost storage tiers. The architectural pattern of compute/storage disaggregation, which allows independent scaling of processing and storage resources, offers documented cost savings by enabling precise resource allocation aligned with workload characteristics. Comprehensive resource allocation optimization through right-sizing and scheduling improvements has yielded substantial cost reductions in public cloud environments through elimination of idle resources and improved utilization patterns [1].

8. The Future of Distributed Data Engineering: Emerging Paradigms and Technological Frontiers

As organizations increasingly rely on data-driven decision making, distributed data engineering continues to evolve as a critical discipline enabling digital transformation across industries. Recent research published in Patterns journal indicates that the volume of global data created, captured, copied, and consumed is projected to continue growing exponentially, with the global data sphere expected to reach substantial volumes by mid-decade. This extraordinary growth, combined with increasingly diverse data sources and formats, is driving fundamental shifts in distributed data architectures. Enterprise surveys reveal that organizations with mature distributed data capabilities demonstrate significant competitive advantages, including faster time-to-market for new data products and higher success rates for analytics initiatives compared to organizations relying on traditional centralized architectures [5]. These compelling outcomes are accelerating investment in advanced distributed data engineering capabilities, with emerging paradigms reshaping how organizations design, deploy, and manage their data infrastructure.

9. Decentralized Data Architectures

The data mesh paradigm represents a revolutionary approach to distributed data infrastructure that addresses fundamental limitations of centralized data platforms. Traditional centralized architectures increasingly struggle with data volume, complexity, and domain diversity in large organizations. The data mesh concept, first formalized in recent years, has rapidly gained adoption with many Fortune 500 companies now implementing or planning data mesh initiatives [7]. This paradigm shift reconceptualizes data as a product owned by domain-specific teams rather than a byproduct of business applications managed by a central data team. Domain-oriented ownership places responsibility

for data quality, documentation, and governance with the teams most familiar with the data's business context and meaning.

Table 4 Data Mesh Architecture Components [7]

Component	Description	Key Benefits
Domain-oriented Data Products	Data owned by domain teams	Improved quality, Domain expertise
Self-serve Data Infrastructure	Platform for creating data products	Reduced technical barriers
Federated Governance	Distributed but consistent policies	Balance standards with autonomy
Interoperable Data Contracts	Standard interfaces for data exchange	Reliable integration

Industry research reveals that organizations implementing data mesh principles report substantial improvements in data usability and time-to-insight. A comprehensive survey of data mesh implementations across multiple industries found that organizations achieved significant reductions in time required to deliver new data products and increased data product adoption by business users [7]. These improvements stem from the core principles of domain ownership, self-service infrastructure, federated computational governance, and interoperable standardization. The domain-oriented approach removes bottlenecks associated with centralized data teams while ensuring that data products incorporate deep domain knowledge. Self-service data infrastructure enables domain teams to efficiently create and maintain data products without depending on specialized platform teams. Federated computational governance establishes consistent policies and standards while maintaining domain autonomy and agility.

Implementation of data mesh architectures requires significant organizational and technological transformation. Case studies of successful implementations reveal that organizations typically progress through multiple maturity stages over extended periods [7]. Initial implementations focus on establishing data domains aligned with business capabilities, defining clear ownership boundaries, and implementing basic data product standards. Advanced implementations incorporate sophisticated discoverability mechanisms, automated quality monitoring, and federated governance frameworks. Technology enablers include data catalogs that support decentralized metadata management, automated data quality monitoring tools, and domain-focused API gateways that standardize access patterns. These technological components enable domain teams to create and maintain high-quality data products that meet enterprise standards while reflecting domain-specific knowledge and requirements.

10. AI/ML Integration with Distributed Systems

The integration of artificial intelligence and machine learning with distributed data systems represents a transformative convergence that is reshaping how organizations extract value from their data assets. Research published in the Journal of Big Data indicates that the worldwide market for distributed AI systems is growing at a substantial compound annual rate, reflecting the critical importance of this technological integration [8]. Traditional machine learning workflows designed for centralized environments face significant limitations when applied to the scale and complexity of modern enterprise data. Distributed approaches address these limitations by enabling sophisticated analytics across petabyte-scale datasets spanning multiple data centers and cloud environments.

Feature engineering at scale represents one of the most significant challenges in enterprise machine learning systems. Traditional feature engineering processes often become bottlenecks when dealing with massive datasets or complex feature interactions. Recent studies indicate that feature preparation typically consumes a majority of total data scientist time in enterprise machine learning projects [8]. Distributed feature engineering platforms address this challenge by enabling parallel computation of features across partitioned datasets. Advanced implementations incorporate feature stores that cache and reuse computed features across multiple models, reducing redundant computation and ensuring consistency. These distributed feature platforms reduce feature preparation time substantially while improving feature quality through automated validation and lineage tracking.

Distributed model training capabilities have become essential as model complexity and dataset sizes continue to grow exponentially. Research demonstrates that state-of-the-art deep learning models have increased in size significantly each year, with the largest models now containing enormous numbers of parameters [8]. Training such models requires sophisticated distributed computing approaches that partition both data and model components across multiple nodes. Benchmarks of distributed training platforms show near-linear scaling efficiency up to large numbers of nodes for certain model architectures, enabling training on datasets that would be impossible to process on single machines.

These platforms implement specialized algorithms for efficient gradient sharing, synchronization, and fault tolerance that maintain training convergence despite the distributed nature of the computation.

Real-time model inference in distributed environments presents unique challenges that differ significantly from training workloads. Latency requirements for inference are typically much more stringent, often requiring response times of milliseconds rather than the hours or days acceptable for training jobs. Industry research reveals that a large majority of production machine learning systems face SLA requirements that cannot be met by batch processing approaches [5]. Distributed inference architectures address these requirements through techniques such as model partitioning, parallel inference, and hierarchical caching. Advanced implementations dynamically route inference requests based on model characteristics, data locality, and current system load. These sophisticated routing mechanisms reduce average inference latency significantly compared to static deployments while maintaining consistent performance during peak load periods.

The operational complexity of distributed ML systems has driven the rapid evolution of MLOps practices specifically adapted for distributed environments. Enterprise surveys indicate that organizations implementing mature MLOps practices reduce model deployment failures substantially and decrease time-to-production compared to those with ad-hoc processes [8]. Distributed MLOps platforms incorporate specialized capabilities such as distributed experiment tracking that maintains consistency across parallel training runs, fleet management for deploying model variants across heterogeneous infrastructure, and sophisticated monitoring that can detect model drift across geographically distributed data sources. These capabilities collectively enable organizations to rapidly develop, deploy, and maintain machine learning models at scale across distributed infrastructure.

10.1. Edge Computing and Distributed Data Engineering

The proliferation of Internet of Things (IoT) devices is driving a fundamental shift in distributed data architectures, with processing increasingly moving from centralized clouds to the network edge. Research indicates that the global number of connected IoT devices is projected to reach a substantial volume by mid-decade, generating enormous amounts of data annually [6]. Moving and processing this massive volume of data exclusively in centralized cloud environments has become prohibitively expensive and inefficient. The resulting architectural shift toward edge computing represents one of the most significant transformations in distributed data engineering in recent years.

Table 5 Edge vs. Cloud Computing [6]

Characteristic	Edge Computing	Cloud Computing	Hybrid Approach
Location	Near data source	Centralized data centres	Distributed across both
Latency	Very low	Higher	Context-dependent
Bandwidth	Minimal	High	Optimized
Offline Capability	Can operate offline	Requires connectivity	Degraded when disconnected
Ideal Use Cases	Real-time processing	Complex analytics	Most enterprise applications

Processing data closer to its source at the network edge offers multiple critical advantages over cloud-centric approaches. Comprehensive analysis of edge computing implementations reveals significant reductions in bandwidth consumption and decreases in end-to-end processing latency compared to cloud-only architectures [6]. These improvements stem from filtering, aggregating, and analyzing data locally before transmitting reduced data volumes to centralized systems. The latency reductions are particularly critical for time-sensitive applications such as industrial automation, autonomous vehicles, and augmented reality where milliseconds can determine success or failure. Industry research indicates that a substantial portion of enterprise IoT implementations now incorporate edge processing capabilities, with this percentage projected to grow significantly in coming years [6].

The integration of edge and cloud computing creates a continuum of processing capabilities that can be optimized based on specific application requirements. A detailed analysis of industrial IoT deployments found that hybrid edge-cloud architectures reduced operational costs substantially compared to cloud-only approaches [6]. These cost savings result from optimized data transfer and storage, reduced cloud computing requirements, and more efficient use of network bandwidth. The hybrid approach enables sophisticated patterns like edge-based filtering and aggregation combined with cloud-based advanced analytics and model training. Data is processed at the appropriate tier of the architecture based on latency requirements, computational complexity, and data volume considerations.

The development of specialized hardware for edge environments is further accelerating this architectural transformation. Neural processing units and other AI accelerators optimized for edge deployment enable local execution of sophisticated machine learning models that previously required data center resources. Market analysis indicates that the edge AI processor market is expected to grow at a significant compound annual rate in coming years [6]. These hardware advances, combined with edge-optimized software frameworks, are enabling increasingly sophisticated distributed data processing at the edge. Early implementations focused primarily on basic filtering and aggregation, while advanced contemporary deployments incorporate complex event processing, machine learning inference, and adaptive analytics that can function even when disconnected from central systems.

11. Serverless Data Processing

Serverless data processing represents a fundamental shift in how distributed data architectures are deployed and managed. This architectural pattern abstracts infrastructure management away from data engineers, allowing them to focus on data processing logic rather than resource provisioning and scaling. Research published in Patterns journal indicates that the serverless computing market is growing at a substantial compound annual rate, reaching a significant market size in the near future [5]. This rapid growth is driven by the significant operational advantages serverless approaches offer for many data processing workloads, including reduced operational complexity, improved resource utilization, and elastic scaling capabilities.

The operational benefits of serverless data processing architectures are substantial for organizations with fluctuating workloads or limited infrastructure management resources. Industry surveys indicate that organizations adopting serverless data processing architectures reduce operational overhead significantly compared to traditional infrastructure management approaches [5]. This reduction stems from eliminating the need to provision, scale, and maintain server infrastructure, allowing data engineering teams to focus on developing and optimizing data transformation logic. The event-driven nature of serverless architectures aligns particularly well with many data processing workloads that are inherently triggered by data arrival events. Case studies of serverless implementations reveal substantial reductions in infrastructure management time and faster deployment of new data processing capabilities compared to traditional server-based approaches.

Cost efficiency represents another significant advantage of serverless approaches for data processing workloads. Traditional data infrastructure often suffers from low utilization, with research indicating relatively low average server utilization rates in many organizations [5]. This low utilization translates to significant waste as organizations pay for idle computing capacity. Serverless data processing enables precise, on-demand resource allocation that scales automatically with workload, eliminating idle capacity costs. A comprehensive analysis of enterprise data processing costs found that serverless implementations reduced total cost of ownership substantially compared to traditionally provisioned infrastructure for appropriate workloads. These savings were particularly significant for organizations with highly variable workloads, where traditional capacity planning approaches typically result in substantial overprovisioning to accommodate peak processing requirements.

The evolution of serverless data processing platforms has addressed many of the early limitations of the paradigm. First-generation serverless offerings imposed significant constraints on execution time, memory allocation, and state management that limited their applicability to complex data processing workloads. Modern serverless platforms have largely overcome these limitations, with contemporary offerings supporting much longer execution times, larger memory allocations, and sophisticated state management capabilities [5]. These advances have expanded the range of data processing use cases that can benefit from serverless deployment models. Complex ETL workflows, real-time stream processing, and machine learning inference workloads increasingly leverage serverless architectures for their operational and cost advantages.

12. Conclusion

Distributed Data Engineering is evolving as a critical discipline enabling organizations to extract value from increasing data volumes with decreasing latency requirements. Converging trends including data mesh architectures, AI/ML integration, edge computing, and serverless processing create new possibilities for distributed intelligence. Research shows organizations with mature distributed data capabilities achieve better business outcomes, including faster time-to-market for data products, higher quality analytics, and more efficient resource utilization. The future will likely see further convergence of these paradigms into integrated architectures spanning from edge to cloud. Data mesh principles are increasingly applied within edge environments, enabling consistent management practices across distributed infrastructure. Serverless models are extending to edge environments, creating computational fabrics that dynamically

place processing based on latency, cost, and data sovereignty requirements. Machine learning is being embedded throughout these architectures, enabling adaptive processing that self-optimizes. Organizations must develop new skills and structures aligned with these distributed paradigms. Traditional centralized data teams are evolving toward federated models combining central governance with distributed domain ownership. Infrastructure management skills are giving way to data product design as serverless models abstract operational complexity. Data engineering and machine learning roles are converging as AI becomes embedded throughout data processing. These organizational transformations, with technology investments, position organizations to maximize value from data assets in an increasingly distributed landscape.

References

[1] Jean Pierre Ntayagabiri, et al, "Study on the Development and Implementation of Different Big Data Clustering Methods," July 2023, scirp, Available: https://www.scirp.org/journal/paperinformation?paperid=126654

[2] Natalia G. Miloslavskaya, Alexander Tolstoy, "Big Data, Fast Data and Data Lake Concepts," December 2016, Procedia Computer Science, Available: https://www.researchgate.net/publication/309183107_Big_Data_Fast_Data_and_Data_Lake_Concepts

[3] Matei Zaharia, et al, "Apache spark: A unified engine for big data processing," November 2016, Communications of the ACM, Available: https://www.researchgate.net/publication/310613994_Apache_spark_A_unified_engine_for_big_data_processing

[4] Nicolaus Henke, et al, "The age of analytics: Competing in a data-driven world," December 7, 2016, mckinsey, Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-age-of-analytics-competing-in-a-data-driven-world

[5] Ruiqing Cao, Marco Iansiti, "Digital transformation, data architecture, and legacy systems," Journal of Digital Economy, Volume 1, Issue 1, June 2022, Available: https://www.sciencedirect.com/science/article/pii/S2773067022000012

[6] Dai Penglin, Md Likhan Ali, "A Survey of Emerging Trends in Edge Computing," May 2024, Available: https://www.researchgate.net/publication/384661251_A_Survey_of_Emerging_Trends_in_Edge_Computing

[7] Christeena Uzhuthuval, "Data Mesh and its Applications in Data & Analytics," ltimindtree, 2022, Available: https://www.ltimindtree.com/wp-content/uploads/2022/12/Data-Mesh-and-its-applications-in-Data-Analytics-PoV.pdf?pdf=download

[8] Mohammad Dehghani, Zahra Yazdanparast, "From distributed machine to distributed deep learning: a comprehensive survey," 13 October 2023, journal of big data, Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00829-x