International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.2 (2025) pp. 3820-3826
http://www.ijcesen.com

ISSN: 2149-9144

Copyright © IJCESEN

Research Article

Optimization Strategies for Cloud-Native Lakehouse Platforms Supporting Large-Scale Data Processing and Machine Learning Workloads

Lokeshkumar Madabathula*

Independent Researcher, San Antonio, Texas, USA
*** Corresponding Author Email:** lokeshkumar.madabathula@gmail.com - **ORCID:** 0009-0002-7104-4193

Article Info:

DOI: 10.22399/ijcesen.5560
Received : 05 April 2025
Revised : 25 May 2025
Accepted : 28 May 2025

Keywords

Cloud-native lakehouse,
Data processing;
Machine learning;
Query optimization;
Resource elasticity;
Cost optimisation

Abstract:

Cloud-native lakehouses are commonly used to handle large scale data processing and machine learning workloads. Nonetheless, increasing data size, varying computation needs, complexities of queries, costs associated with infrastructure and issues related to governance pose various optimization challenges. This paper seeks to understand how optimization can be done in cloud-native lakehouses in order to improve processing performance, scalability, machine learning efficiency, resource utilization, efficiency and security. The qualitative research approach was employed through secondary data from academic journals, technical articles, conference proceedings, reports and reliable technical sources. Inductive approach was used to detect various optimization trends from the secondary data. It is evident that separation of compute-storage, elasticity, workload scheduling, partitioning, columnar storage, caching, compression and optimization of queries result in processing efficiency. Machine learning workloads get benefits from resource elasticity, data locality, workload-based scheduling and optimization of GPUs/CPU. The results further demonstrate that automation of provision and dynamic scaling can lead to minimisation of underutilisation of infrastructure and operational efficiency improvements. In addition, metadata management, access control, encryption, audit trails, and data lineage add up to governance and security aspects in a distributed lakehouse architecture. From the study it is clear that optimization of systems should not only consider speed of execution of queries or infrastructure costs. Rather, performance, scalability, resource utilization, security, governance, and complexity of operations should all be taken into account. Thus, a workload-aware approach to system optimization can offer more sustainable architecture for large scale analytics and machine learning.

1. Introduction

Cloud-native lakehouses combine the agility of a data lake with the speed and management of a data warehouse. There are various ways in which optimizations can be done for scaling of data processing and machine learning operations efficiently. Some storage optimizations include column-based storage, which can be achieved through techniques like Parquet, data compression, data partitioning, and clustering. Some compute optimizations are elastic cloud compute resources, auto-scaling, load and job scheduling, and distribution by using cloud-native compute engines like Apache Spark. Cache and query optimizations will help in reducing repeated access to data and boost performance in analytics processing operations. A lakehouse infrastructure will also be able to benefit from Delta Lake capabilities such as ACID transactions, schema validation, time travel, and file optimizations. Some machine learning optimizations include optimized feature storage, parallelization of data processing, and scalable compute resources like GPUs or CPUs.

2. Method

In this study, qualitative research methodology using secondary data was applied. The reason for choosing this method is that the research looks into the existing knowledge of lakehouse optimization techniques in the cloud environment (Cheong *et al.,* 2023). Specifically, the research was concerned with large scale data processing and machine learning workloads. Qualitative methodology would be useful in explaining various optimization

techniques and their relevance. The secondary data were collected from academic journals, research papers, conference papers, industry reports, technical journals, and online sources. These sources have been used since they contain established technical knowledge and industry information. This research adopted the inductive approach in its design and analysis. The inductive approach was preferred in this case as the research intended to generate patterns out of existing literature (Roberts, Dowell & Nie, 2019). This approach does not involve having a preconceived hypothesis at the beginning. Instead, observations from various sources form the basis of generating findings in this case. The approach allowed the identification of the common optimization techniques in cloud-native lakehouses. The research looked into issues of data partitioning, query optimization, caching, workload management, elastic resource allocation, and storage optimization in order to establish how each contributes to the efficiency and effectiveness of processing and machine learning tasks (Shah & Gogineni, 2022). This approach facilitated comparison between the various approaches as well as their perceived advantages.

3. Results

3.1 Cloud-Native Lakehouse Performance

Cloud-native lakehouses have the potential to enhance performance in terms of data processing at scale. They combine flexible data storage with fast analytics. This enables them to be less dependent on data platforms and data movement (Hambardzumyan *et al.,* 2022). The performance increases due to separation of compute and storage. It makes it possible to scale processing power in accordance with the demand for resources. Serverless processing enhances query performance and flexibility of resource usage.

One of the identified performance optimization methods is caching. In a recently conducted benchmark test of a lakehouse, a 21% reduction in the execution time was recorded with cache implementation. The median query performance increased by about 12%, while the best query achieved a 77% improvement. Recent studies emphasize such performance optimization methods as storage tiering, columnar compression, and query routing.

3.2 Scalable Data Processing

The results show that scalable data processing is necessary for cloud-native lakehouses in light of the growing amount of data (Li & Avestimehr, 2020). Distributed processing ensures that workloads are distributed over several compute nodes, thereby minimizing processing problems and facilitating parallel execution of data. Scalability is critical when considering machine learning workloads that involve millions of data points and high dimensionality. Disaggregated compute and storage capabilities in cloud-native designs facilitate scalability. Compute nodes, hence, scale depending on workload demands (Weng *et al.,* 2022).

Recent studies of cloud-native database design show that scalability is enabled by this architecture separation. The experiment showed an up to 2.57× reduction in query latency, showing how well scalable architectures perform. Technical results further prove that scalability requires a parallel algorithm approach, efficient data structures, and load balancing. GRAPE is one such approach that uses parallelism and memory-efficient data structures to analyze very big graphs. It was able to improve its performance by two to three orders of magnitude through its approximate random walk approach for large graphs.

3.3 Storage and Query Optimization

From the results of the experiment, it is clear that the design of storage affects the performance of queries in a cloud-native lakehouse. The columnar format of storage like Parquet minimizes unnecessary data scans when performing analytics. Delta Lake enhances the storage management by introducing features such as transaction logging, metadata management, and ACID (Armbrust *et al*., 2020). This ensures the availability of data in large quantities. Partition pruning became an important method for query optimization.

By proper partitioning, query processors will not have to scan unnecessary data files. File level statistics allow data skipping. But too much partitioning may lead to creation of small files. Thus, the partitioning process must take into account the access patterns of the workload. The study highlights file compaction, clustering, caching, and adaptive query execution as significant optimization techniques. File compaction will decrease the problem of small-file fragmentation. The Z-order clustering technique will help in increasing data locality of often filtered columns (Lehmann *et al.,* 2020). On the other hand, caching will decrease storage accesses of repetitive workloads. In addition, recent studies have shown that access-based partitioning can greatly reduce the cost of cloud storage without affecting its

performance. One such study saved 50-83% in costs on large-scale workloads.

3.4 Machine Learning Workload Optimization

Machine learning workloads have various requirements on cloud-native lakehouse architecture. Training process needs high computational power, whereas inference process needs high speed and stable resource provision. The difference in requirements makes optimization for specific workload necessary. Shared lakehouse storage may provide support for training data sets, feature pipelines, and analysis without creating any duplicate data sets. Resource allocation becomes an important parameter in optimization. Allocation of CPU, GPU, memory, and storage resources becomes possible depending on the load of work. Dynamic allocation proves to be better compared to static allocation because machine learning workload varies significantly depending on different stages of processing (Wang *et al.,* 2019).Kubernetes is an open-source tool that is used to control containerized applications. It is a modular system that works very well with AI and ML-based applications. Essentially, Kubernetes uses a cluster system made up of several nodes, which in turn contain pods, which are the smallest deployable unit that can be managed and scheduled. The cluster system is controlled by a master node, which is responsible for deployment, scheduling, and scaling of workloads across different worker nodes. Machine learning workloads have many differences in their features that can be observed during the research process. Adaptive Resource Management & Intelligent Scheduling is further validated by literature. An application of deep reinforcement learning resulted in enhanced quality of experience and lower energy use via dynamic allocation of resources. Data locality, caching, autoscaling, GPU usage, and pipeline scheduling within the lakehouses could hence lead to lowered processing latencies and resource wastage (Han, 2021).

3.5 Resource Allocation and Elasticity

The resource allocation process influences how well a cloud-native lakehouse copes with varying workload intensity. In case of large-scale applications, there might be under-utilization or lack of processing power due to static infrastructure. However, elastic systems overcome this challenge through the ability to scale compute power depending on workload (Moreno-Vozmediano *et al.,* 2019). It is especially important in the cases when both analytics and machine learning tasks are performed at the same time. Thereby, the decoupling of compute from storage is vital to the concept of elasticity. Storage becomes persistent, while compute can be scaled up or down independently. Research on cloud-native database also shows that elasticity of resource management is another essential issue for performance maintenance under various loads. Resource allocation gets even more complicated in case different workloads use one common infrastructure. Workload prioritization, autoscaling, scheduling, and isolation can help to protect critical workloads from getting impacted by high-demand workloads (Rzadca *et al.,* 2020). Kubernetes based orchestration might additionally assist in automating scaling and resource management in containers. Elasticity is also related directly to cost-effectiveness because resources can be released when the demand goes down and increased when there is a need for peak processing.

3.6 Cost and Operational Efficiency

The cost efficiency of cloud-native lakehouses is reliant on efficient management of computational resources, storage resources, and processing resources. Elastic computing ensures that the resources scale upwards when there are high-intensity tasks and scale downwards when the demand is low. This helps cut down the cost of unneeded resources and improves resource efficiency (Kianian, Kurdve & Andersson, 2019). Techniques like compression, pruning, columnar storage, and lifecycle management would be helpful for decreasing storage expenses as well as unnecessary scanning of information. Automation improves efficiency due to allocation, monitoring, scheduling, and failure recovery. Dynamic allocation is useful in lowering costs associated with underutilization of CPU, GPU, memory, and storage resources. Automated scaling also has the ability to adapt to workload without necessarily having to constantly intervene. Automation on excess may be costly in terms of management. Cost optimization needs a balance between various factors (Margherita & Braccini, 2021).

3.7 Data Governance and Security

Data governance and security are critical components for ensuring lakehouse operations within the cloud-native environment. Access control, encryption, metadata management, auditing, and data lineage are required for the operation of large-scale systems. Access controls based on roles ensure that only the authorized user gains access to the data. Encryption provides

protection to data when it is stored and transferred (Yang, Xiong & Ren, 2020). The metadata helps in providing more visibility to the data's ownership, location, and usage. In case of distributed clouds, identity based security and continuous monitoring are required. For machine learning, data governance helps in ensuring the integrity of data and models.

4. Discussion

From the above discussion, it is clear that cloud-native lakehouse optimization necessitates a comprehensive approach toward optimization in terms of performance, scalability, storage, machine learning, resource management, costs, and governance. It can be seen from the above discussion that no single optimization technique can meet the requirements of large scale data processing. Rather, it depends upon the way various optimization techniques are inter-related and perform. Cloud-native lakehouses form a crucial basis in this context with the help of segregation of compute and storage capabilities (Hambardzumyan *et al*., 2022). This kind of architecture makes it possible for computing capacity to scale as per the workload. It is more beneficial in cases where there are varying requirements of workload. The results also reveal the fact that storage and query optimization contribute to process efficiency. Partitioning, compression, columnar storage, caching, and data skipping help in avoiding unnecessary data movement and processing of queries. But these strategies depend on workload-aware configuration. Inefficient partitioning or creation of many small files would increase metadata costs and lead to poor performance. Hence, the decisions regarding optimization must be workload-aware. The machine learning workloads present new complications since training and inference operations are computationally different (Gao *et al.,* 2022). Training may involve heavy use of GPUs and memory while inference may need fast latency. Consequently, dynamic scheduling and workload-aware allocation of resources can enhance efficiency. Access to data locality and feature access efficiently will help in reducing processing time delays. Elasticity has strong connection with cost and efficiency. Hence, automation of provision, monitoring, scaling, and workload scheduling can lead to reduced manual intervention. This can help in avoiding the use of inefficient resources. But aggressive scaling of resources will lead to increased complexities.

Data governance and security have to stay connected to the concept of optimization (Berlato *et al.,* 2020). Access control, encryption, metadata, lineage, and monitoring contribute to reliable data operations. Governance plays a vital role especially in cases where the use of lakehouses is for machine learning or sensitive enterprise data. On balance, from the discussion presented above, there is an idea that an efficient cloud-native lakehouse requires workload-based, elastic, and automatic optimization.



Figure 1: *Growth Curves of Data Warehouse and Data Lake (Source: Alibaba Cloud, 2022)*

Table 1: *Quantitative Performance Indicators for Scalable Cloud-Native Data Processing*

Study / Technique	Metric	Baseline / Reference	Reported Result	Improvement
Cloud-native database	Query latency	1.00×	0.39×	**2.57× faster**

architecture				
GRAPE	Graph processing performance	1×	100–1,000×	**2–3 orders of magnitude**
Distributed processing	Compute nodes	1 node	Multiple nodes	**Parallel scalability**
Machine-learning workload	Dataset scale	Thousands	Millions of data points	**1000×+ data-scale capability***
Cloud compute–storage separation	Compute allocation	Fixed	Elastic	**Dynamic scaling**
GRAPE	Memory efficiency	Conventional structures	Memory-efficient structures	**Reduced memory overhead**
Streaming analytics	Processing latency	Minutes–hours	Milliseconds–seconds	**Near-real-time processing**

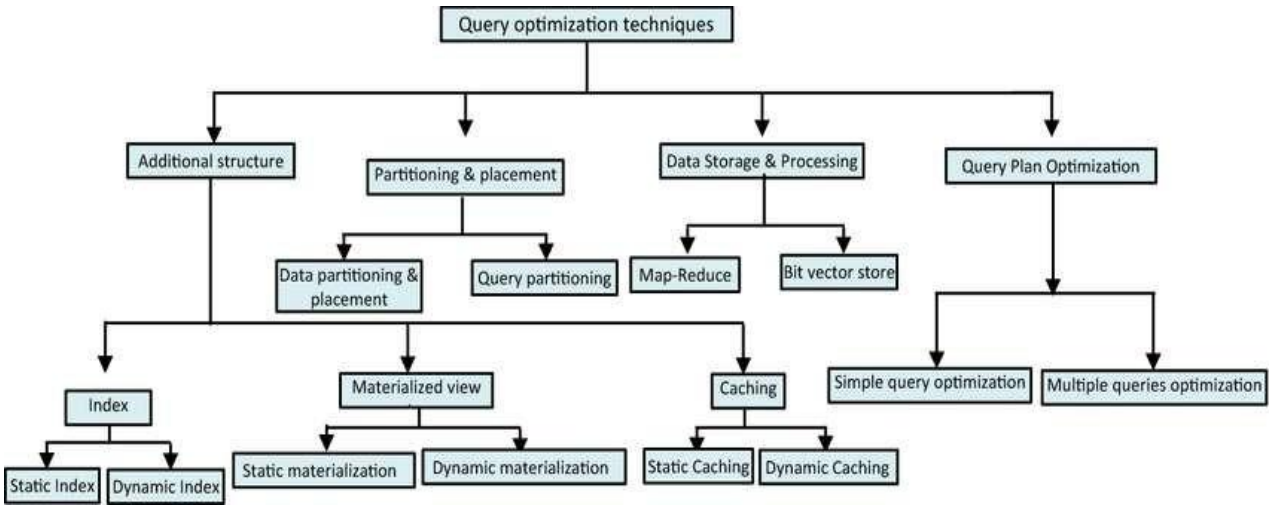


Figure 2: *Classification of query optimization techniques (Source: Sebaa & Tari, 2019)*

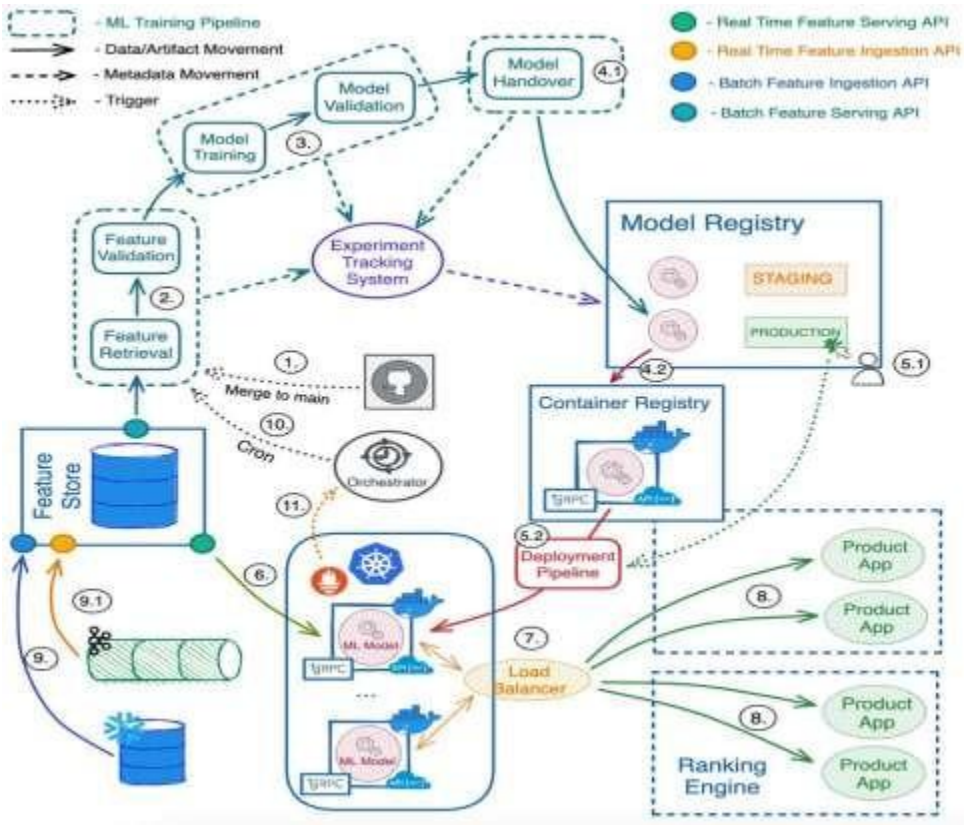


Figure 3: *Step-by-Step Breakdown of Real-Time ML Operational Deployment Pipeline(Source: Adams, 2023)*

Table 2: *Resource Elasticity and Workload Optimization*

Resource Parameter	Static Architecture	Elastic Architecture	Numeric Indicator	Technical Benefit
Compute Scaling	Fixed	Dynamic	**2 modes**	Scale-up/down with demand
Workload Types	Separate capacity	Shared infrastructure	**2+ types**	Supports analytics and ML
Resource Management	Manual	Automated	**3 mechanisms**	Scheduling, autoscaling, isolation
Compute–Storage Model	Coupled	Decoupled	**2 layers**	Independent scaling
Scaling Response	Fixed capacity	Demand-driven	**2 states**	Reduces idle-resource cost

4. Conclusions

Cloud native lakehouses offer a scalable basis for big data processing and machine learning tasks. As seen from the research, compute elasticity, storage optimization, query optimization, workload scheduling, and automatic resource allocation may increase performance and resource usage. Efficient storage management is helpful in avoiding unnecessary data movement and expenses associated with its processing. Workload-aware scheduling and infrastructure provisioning are beneficial for machine learning tasks. Governance and security contribute to efficient and ethical data management. Generally, optimization should be viewed from the point of view of performance, scalability, cost, and governance. A balanced approach based on workloads will allow companies to develop flexible and secure lakehouse architectures.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

1. Adams, F. (2023). *Step-by-step breakdown of real-time ML operational deployment pipeline*. Medium. Retrieved at https://medium.com/@FrankAdams7/step-by-step-breakdown-of-real-time-ml-operational-deployment-pipeline-848f290eb5d9
2. Alibaba Cloud. (2022, January 14). *Data warehouse and lake house: Continuous evolution of the cloud-native big data platform*. Alibaba Cloud
3. Armbrust, M., Das, T., Sun, L., Yavuz, B., Zhu, S., Murthy, M., ... & Zaharia, M. (2020). Delta lake: high-performance ACID table storage over cloud object stores. *Proceedings of the VLDB Endowment*, *13*(12), 3411-3424. https://www.eecs.umich.edu/courses/cse584/static_files/papers/p975-armbrust.pdf
4. Berlato, S., Carbone, R., Lee, A. J., & Ranise, S. (2020, October). Exploring architectures for cryptographic access control enforcement in the cloud for fun and optimization. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security* (pp. 208-221). https://dl.acm.org/doi/abs/10.1145/3320269.3384767
5. Cheong, H. I., Lyons, A., Houghton, R., & Majumdar, A. (2023). Secondary qualitative research methodology using online data within the context of social sciences. *International Journal of Qualitative Methods*, *22*, 16094069231180160. https://journals.sagepub.com/doi/abs/10.1177/16094069231180160
6. Gao, W., Hu, Q., Ye, Z., Sun, P., Wang, X., Luo, Y., & Wen, Y. (2022). Deep learning workload scheduling in gpu datacenters: Taxonomy, challenges and vision. *arXiv preprint arXiv:2205.11913*. https://arxiv.org/abs/2205.11913
7. Hambardzumyan, S., Tuli, A., Ghukasyan, L., Rahman, F., Topchyan, H., Isayan, D., & Buniatyan, D. (2022). Deep lake: A lakehouse for deep learning. *arXiv preprint arXiv:2209.10785*. https://arxiv.org/abs/2209.10785
8. Hambardzumyan, S., Tuli, A., Ghukasyan, L., Rahman, F., Topchyan, H., Isayan, D., &

Buniatyan, D. (2022). Deep lake: A lakehouse for deep learning. *arXiv preprint arXiv:2209.10785*. https://arxiv.org/abs/2209.10785

9. Han, Y. S. (2021). High-Performance Data Management Architectures for Scalable Machine Learning Pipelines in Cloud Ecosystems. *American International Journal of Computer Science and Technology*, *3*(2), 11-20. http://aijcst.org/index.php/aijcst/article/view/29
10. Kianian, B., Kurdve, M., & Andersson, C. (2019). Comparing life cycle costing and performance part costing in assessing acquisition and operational cost of new manufacturing technologies. *Procedia Cirp*, *80*, 428-433. https://www.sciencedirect.com/science/article/pii/S2212827119300277
11. Lehmann, C., Goren Huber, L., Horisberger, T., Scheiba, G., Sima, A. C., & Stockinger, K. (2020). Big Data architecture for intelligent maintenance: a focus on query processing and machine learning algorithms. *Journal of Big Data*, *7*(1), 61. https://link.springer.com/article/10.1186/s40537-020-00340-7
12. Li, S., & Avestimehr, S. (2020). Coded computing: Mitigating fundamental bottlenecks in large-scale distributed computing and machine learning. *Foundations and Trends in Communications and Information Theory*, *17*(1), 1-148. https://www.emerald.com/ftcit/article/17/1/1/1332643
13. Margherita, E. G., & Braccini, A. M. (2021). Managing industry 4.0 automation for fair ethical business development: A single case study. *Technological Forecasting and Social Change*, *172*, 121048. https://www.sciencedirect.com/science/article/pii/S0040162521004807
14. Moreno-Vozmediano, R., Montero, R. S., Huedo, E., & Llorente, I. M. (2019). Efficient resource provisioning for elastic cloud services based on machine learning techniques. *Journal of Cloud Computing*, *8*(1), 1-18. https://link.springer.com/article/10.1186/s13677-019-0128-9
15. Roberts, K., Dowell, A., & Nie, J. B. (2019). Attempting rigour and replicability in thematic analysis of qualitative research data; a case study of codebook development. *BMC medical research methodology*, *19*(1), 66. https://link.springer.com/chapter/10.1007/978-3-030-22113-3_9
16. Rzadca, K., Findeisen, P., Swiderski, J., Zych, P., Broniek, P., Kusmierek, J., & Wilkes, J. (2020, April). Autopilot: workload autoscaling at google. In *proceedings of the fifteenth european conference on computer systems* (pp. 1-16). https://dl.acm.org/doi/abs/10.1145/3342195.3387524
17. Sebaa, A., & Tari, A. (2019). Query optimization in cloud environments: Challenges, taxonomy, and techniques. *The Journal of Supercomputing, 75*(6), 5420–5450. https://doi.org/10.1007/s11227-019-02806-9
18. Shah, M., & Gogineni, A. (2022). Distributed Query Optimization for Petabyte-Scale Databases. *Int. J. Recent Innov. Trends Comput. Commun*, *10*(10), 223-231. https://openreview.net/pdf?id=NkGdG15swy
19. Wang, H., Wu, Y., Min, G., Xu, J., & Tang, P. (2019). Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach. *Information Sciences*, *498*, 106-116. https://www.sciencedirect.com/science/article/pii/S0020025519303986
20. Weng, Q., Xiao, W., Yu, Y., Wang, W., Wang, C., He, J., & Ding, Y. (2022). {MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters. In *19th USENIX symposium on networked systems design and implementation (NSDI 22)* (pp. 945-960). https://www.usenix.org/conference/nsdi22/presentation/weng
21. Yang, P., Xiong, N., & Ren, J. (2020). Data security and privacy protection for cloud storage: A survey. *Ieee Access*, *8*, 131723-131740. https://ieeexplore.ieee.org/abstract/document/9142202/