International Journal of AI, Big Data, Computational and Management Studies
Noble Scholar Research Group | ICAIDSCT26-Conference Proceeding
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.ICAIDSCT26-144

Cloud Data Optimization: Performance Tuning for Batch Processing and Real-Time Streaming

Subhasis Kundu
Solution Architecture & Design, Roswell, GA, USA.

Abstract: This paper explores performance tuning techniques for cloud data workflows, focusing on both batch processing and real-time streaming. It addresses key challenges in scalability, efficiency, and latency reduction to optimize data handling in cloud environments. Various strategies for resource allocation, load balancing, and data partitioning are analyzed to enhance throughput and minimize processing delays. The study evaluates the impact of tuning parameters on system performance through experimental results and case studies. Emphasis is placed on balancing cost-effectiveness with computational demands. Insights into adaptive optimization approaches for dynamic workloads are also provided. The findings demonstrate significant improvements in processing speed and resource utilization. This work contributes practical guidelines for optimizing cloud-based data pipelines in diverse operational contexts.

Keywords: Cloud Computing, Data Optimization, Performance Tuning, Batch Processing, Real-Time Streaming, Scalability, Latency Reduction.

1. Introduction

1.1. Background and Motivation

Cloud data workflows have become integral to modern data-driven applications, handling vast volumes of information through batch processing and real-time streaming. The increasing demand for scalable, efficient, and low-latency data processing in cloud environments motivates the need for performance tuning techniques tailored to these workflows. Challenges such as resource constraints, fluctuating workloads, and cost considerations necessitate optimized strategies to maintain system responsiveness and throughput. This paper addresses these challenges by exploring methods to enhance scalability, improve efficiency, and reduce latency in cloud-based data pipelines.[1] The motivation stems from the critical role that optimized cloud data workflows play in supporting timely decision-making and operational agility across industries. By focusing on both batch and streaming paradigms, the study aims to provide a comprehensive framework for performance enhancement. The background highlights the evolving complexity of cloud data systems and the importance of adaptive tuning to meet dynamic workload demands. Ultimately, the goal is to deliver practical insights that enable more effective utilization of cloud resources while balancing performance and cost.[2]

1.2. Objectives and Scope

The objectives of this paper are to systematically investigate and present performance tuning techniques that enhance scalability, efficiency, and latency reduction in cloud data workflows involving both batch processing and real-time streaming. It aims to identify key optimization strategies such as resource allocation, load balancing, and data partitioning that directly impact processing speed and system responsiveness. The scope encompasses a comprehensive analysis of tuning parameters and adaptive approaches suited for dynamic workloads in diverse cloud environments. By integrating experimental evaluations and case studies, the study seeks to validate the effectiveness of these techniques in practical scenarios. The paper also intends to balance computational demands with cost-effectiveness, providing actionable guidelines for optimizing cloud-based data pipelines. Ultimately, the work targets researchers and practitioners looking to improve cloud data processing performance across various operational contexts while addressing evolving challenges in scalability and latency.[3], [4]

1.3. Paper Overview

The paper is structured to provide a clear and logical progression of topics related to cloud data optimization and performance tuning. It begins with an introduction outlining the background, motivation, objectives, and scope. This is followed by an overview of cloud data workflows, covering both batch processing and real-time streaming, along with their comparative analysis. Next, the paper addresses the key performance challenges in cloud environments, focusing on scalability, efficiency, and latency. Subsequent sections delve into specific optimization techniques for batch processing and real-time streaming, detailing resource allocation, data partitioning, load balancing, adaptive tuning, and latency minimization. An experimental evaluation section presents the methodology, performance metrics, results, and a discussion of findings through case studies. The paper concludes with a summary of insights and practical guidelines for enhancing cloud-based data workflows.[5]

2. Cloud Data Workflows Overview

2.1. Batch Processing Fundamentals

Batch processing fundamentals involve the systematic handling of large volumes of data collected over a period, processed as a single unit or batch. This approach is well-suited for tasks that do not require immediate results but benefit from high

throughput and efficient resource utilization. Key characteristics include scheduled execution, fault tolerance, and the ability to process data in parallel across distributed cloud environments. Batch processing workflows typically involve stages such as data ingestion, transformation, aggregation, and storage, optimized to maximize throughput while minimizing processing time. Scalability is achieved by dynamically allocating cloud resources based on workload demands, ensuring efficient handling of varying data sizes. Effective batch processing depends on tuning parameters like partitioning schemes, resource allocation, and load balancing to reduce bottlenecks and latency. This paradigm contrasts with real-time streaming by prioritizing volume and completeness over immediacy. Understanding these fundamentals provides a foundation for exploring performance optimization techniques tailored to batch workloads in cloud environments.[6], [7]

2.2. Real-Time Streaming Concepts

Real-time streaming concepts focus on the continuous processing of data as it is generated, enabling immediate analysis and response. Unlike batch processing, streaming workflows prioritize low latency and near-instantaneous data handling to support time-sensitive applications. Key features include event-driven architectures, real-time data ingestion, and incremental processing using stream processing frameworks. These workflows require robust mechanisms for fault tolerance, state management, and dynamic scaling to handle fluctuating data rates. Adaptive tuning parameters such as window size, checkpoint intervals, and parallelism levels are critical to maintaining system responsiveness and minimizing latency. Real-time streaming supports use cases like monitoring, alerting, and live analytics, where timely insights are essential. The complexity of streaming systems demands optimization techniques that balance throughput with latency while ensuring resource efficiency. Understanding these concepts is vital for developing performance tuning strategies tailored to real-time cloud data workflows.[8], [9]

2.3. Comparative Analysis

Batch processing and real-time streaming represent two distinct paradigms in cloud data workflows, each with unique characteristics and optimization needs. Batch processing excels in handling large volumes of accumulated data with a focus on throughput and resource efficiency, operating on scheduled, discrete data sets. In contrast, real-time streaming emphasizes continuous, low-latency data processing to enable immediate insights and responsiveness for time-sensitive applications. While batch workflows prioritize completeness and scalability through parallel processing, streaming workflows require adaptive tuning to manage fluctuating data rates and maintain system responsiveness. Both paradigms face challenges related to scalability, efficiency, and latency but differ in their operational priorities and tuning parameters. Batch processing benefits from strategies like data partitioning and load balancing to optimize resource use, whereas streaming relies on dynamic scaling, windowing, and checkpointing to minimize delays.[2], [10] Understanding these differences is crucial for developing tailored performance tuning techniques that address the specific demands of each workflow type within cloud environments. Same depicted in Fig. 1.

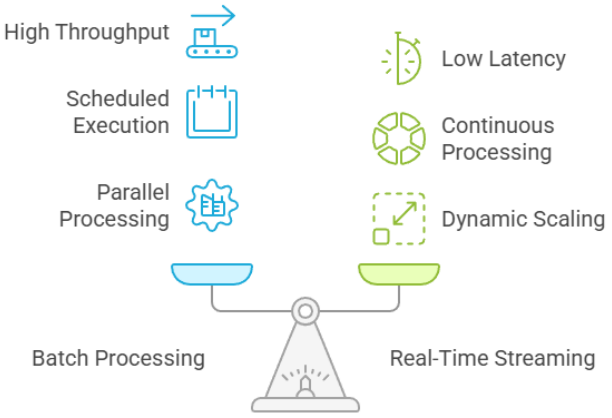

Figure 1: Comparing Cloud Data Workflow Paradigms

3. Performance Challenges in Cloud Environments

3.1. Scalability Constraints

Scalability constraints in cloud environments pose significant challenges to optimizing data workflows for batch processing and real-time streaming. These constraints arise from limitations in resource availability, network bandwidth, and the overhead associated with managing distributed systems at scale. As workloads grow unpredictably, static resource allocation can lead to bottlenecks, reduced throughput, and increased latency. Cloud systems must dynamically scale compute, storage, and network resources to handle varying data volumes without compromising performance.[11] However, scaling introduces complexity in maintaining load balancing and data consistency across nodes. Additionally, cost considerations often limit the extent of scaling, requiring efficient tuning to maximize resource utilization. Addressing scalability constraints involves designing adaptive mechanisms that respond to workload fluctuations while ensuring system stability and responsiveness. Effective scalability solutions must balance elasticity with operational overhead to support both high-throughput batch jobs and low-latency

streaming processes. Understanding these constraints is essential for developing tuning strategies that enhance cloud data workflow performance.[12]

3.2. Efficiency Bottlenecks

Efficiency bottlenecks in cloud data workflows arise from suboptimal resource utilization, data movement overheads, and processing delays that impede throughput in both batch and real-time streaming environments. Inefficient allocation of compute, storage, and network resources can lead to underperformance, where some components remain idle while others are overloaded. Data skew, improper partitioning, and serialization overhead exacerbate these bottlenecks by causing uneven workload distribution and increased processing times. Additionally, the overhead of fault tolerance mechanisms and synchronization in distributed systems can reduce efficiency.[13] In streaming workflows, frequent state updates and checkpointing add latency and resource consumption. Overcoming these bottlenecks requires fine-tuning of resource allocation, load balancing, and data partitioning strategies to ensure balanced workload distribution and minimize idle times. Optimizing data serialization and compression techniques further enhances throughput. Addressing efficiency bottlenecks is critical to maximizing cloud resource utilization and maintaining high-performance data processing pipelines. Understanding these challenges informs the development of targeted tuning techniques that improve overall system responsiveness and cost-effectiveness.[14]

3.3. Latency Issues

Latency issues in cloud data workflows present critical challenges for both batch processing and real-time streaming, directly affecting system responsiveness and user experience. Latency arises from factors such as network delays, data serialization overhead, and processing inefficiencies within distributed cloud environments. In batch processing, latency can increase due to large data volumes and complex transformations, leading to delayed job completion times. For real-time streaming, maintaining low latency is essential to support immediate data analysis and decision-making, but it is often hindered by checkpointing, state management, and fluctuating input rates. Additionally, contention for shared resources and synchronization overhead among distributed nodes contribute to latency spikes. Effective latency reduction requires fine-tuning of processing pipelines, optimizing data flow paths, and employing adaptive buffering and windowing techniques. Balancing latency minimization with throughput and cost constraints is vital for sustaining high-performance cloud data systems. Understanding these latency challenges guides the development of targeted tuning strategies to enhance overall workflow efficiency and timeliness in cloud environments.[15], [16], [17]

4. Optimization Techniques for Batch Processing

4.1. Resource Allocation Strategies

Resource allocation strategies for batch processing in cloud environments focus on efficiently distributing compute, storage, and network resources to optimize throughput and minimize processing time. These strategies involve dynamically provisioning resources based on workload demands, enabling scalable handling of large data volumes while avoiding underutilization or bottlenecks. Key approaches include workload-aware scheduling, prioritizing tasks to match resource availability, and leveraging cloud elasticity to scale resources up or down as needed. Effective resource allocation also incorporates partitioning data to parallelize processing and balance loads across distributed nodes.[18] Additionally, cost considerations are integrated to ensure that resource provisioning aligns with budget constraints without compromising performance. By tuning allocation parameters, such as CPU cores, memory size, and I/O bandwidth, batch workflows achieve improved efficiency and reduced job completion times. Adaptive mechanisms that respond to workload fluctuations further enhance system responsiveness and stability. These strategies collectively contribute to maximizing resource utilization while supporting the high-throughput demands of batch processing in cloud data workflows.[19]

4.2. Data Partitioning Methods

Data partitioning methods in batch processing involve dividing large datasets into smaller, manageable segments to enable parallel processing and improve throughput in cloud environments. Effective partitioning ensures balanced workload distribution across compute nodes, reducing bottlenecks and preventing resource contention. Common approaches include range partitioning, hash partitioning, and round-robin techniques, each suited to different data characteristics and query patterns. Proper partitioning minimizes data skew, which can cause uneven processing times and degrade overall performance. Additionally, partitioning facilitates efficient data locality, reducing network overhead and I/O latency during processing. [20] Tuning partition size and granularity is critical to optimizing resource utilization and minimizing job completion times. Adaptive partitioning strategies that respond to workload changes further enhance system scalability and efficiency. These methods collectively support high-throughput batch workflows by enabling fine-grained control over data distribution and processing parallelism.[6]

4.3. Load Balancing Approaches

Load balancing approaches in batch processing focus on evenly distributing workloads across cloud compute nodes to prevent resource contention and optimize processing efficiency. Effective load balancing reduces bottlenecks by ensuring that no single node becomes a performance limiter due to disproportionate task assignment. Techniques include static load balancing, where workloads are pre-assigned based on estimated capacity, and dynamic load balancing, which adapts task distribution in

real-time according to node performance and workload fluctuations. Strategies such as task queuing, workload redistribution, and feedback-driven scheduling help maintain balanced processing loads.[21] Proper integration with data partitioning methods enhances load balancing by aligning data segments with compute resources to minimize data movement and latency. Additionally, load balancing must consider fault tolerance and recovery to sustain workflow stability during node failures. Tuning load balancing parameters improves throughput and reduces job completion times, contributing to scalable and efficient batch processing in cloud environments. These approaches collectively support robust resource utilization and system responsiveness under varying workload conditions.[22] Same depicted in Fig. 2.

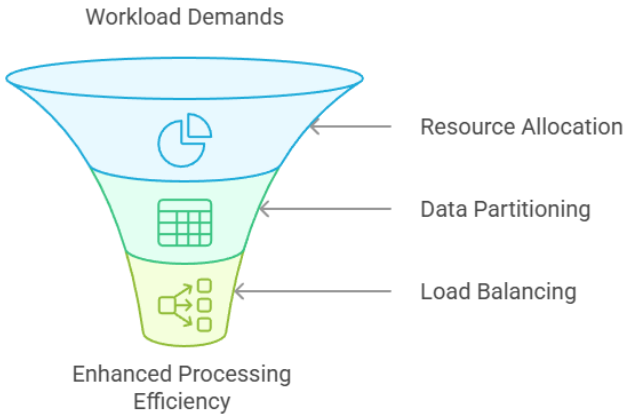

Figure 2: Optimizing Batch Processing in Cloud

5. Optimization Techniques for Real-Time Streaming

5.1. Stream Processing Frameworks

Stream processing frameworks provide the foundational infrastructure for real-time data streaming workflows in cloud environments, enabling continuous and incremental data processing with low latency. These frameworks support event-driven architectures that handle high-velocity data streams by offering features such as stateful processing, fault tolerance, and dynamic scaling. Popular frameworks incorporate mechanisms for windowing, checkpointing, and exactly-once processing semantics to ensure data accuracy and consistency despite failures. They facilitate parallelism and distributed execution, allowing workloads to be balanced across multiple compute nodes efficiently.[23] The choice and configuration of a stream processing framework significantly impact system responsiveness and resource utilization. Effective tuning of framework parameters, such as parallelism level, checkpoint intervals, and buffer sizes, optimizes throughput while minimizing latency. By leveraging these frameworks, cloud-based streaming workflows can adapt to fluctuating data rates and maintain high availability. Understanding the capabilities and limitations of stream processing frameworks is essential for implementing performance tuning strategies tailored to real-time cloud data workflows.[24]

5.2. Adaptive Tuning Parameters

Adaptive tuning parameters in real-time streaming workflows enable dynamic adjustment of system settings to maintain optimal performance amid fluctuating data rates and workload variability. Key parameters include window size, which determines the scope of data processed at once; checkpoint intervals, balancing fault tolerance with processing overhead; and the level of parallelism, influencing resource utilization and throughput. Fine-tuning these parameters helps minimize latency while ensuring data consistency and system stability.[25] Adaptive mechanisms monitor runtime metrics and adjust configurations in real time, allowing the streaming system to respond efficiently to changes without manual intervention. This approach supports maintaining low-latency processing and high availability, even under unpredictable load conditions. By integrating adaptive tuning with stream processing frameworks, cloud-based workflows achieve enhanced responsiveness and resource efficiency. Understanding and implementing these adaptive strategies is crucial for optimizing performance in real-time cloud data streaming environments.[26]

5.3. Latency Minimization Techniques

Latency minimization techniques in real-time streaming focus on reducing the delay between data generation and processing to ensure timely insights and responsiveness. These techniques include optimizing windowing strategies to balance the amount of data processed and the frequency of computation, thereby controlling processing intervals and buffering delays. Fine-tuning checkpoint intervals reduces overhead while maintaining fault tolerance, preventing excessive latency caused by frequent state snapshots. Efficient state management and incremental processing help minimize the time spent on data updates and recovery.[27] Load shedding and backpressure mechanisms regulate data flow during peak loads to avoid congestion and processing bottlenecks. Additionally, employing adaptive buffering techniques adjusts data batching dynamically based on workload and network conditions to sustain low latency. Network optimization, such as minimizing serialization overhead and reducing data transfer times, also contributes to latency reduction. Together, these techniques ensure real-time streaming

workflows maintain high throughput with minimal delay, supporting critical time-sensitive applications in cloud environments.[28] Same depicted in Fig. 3.

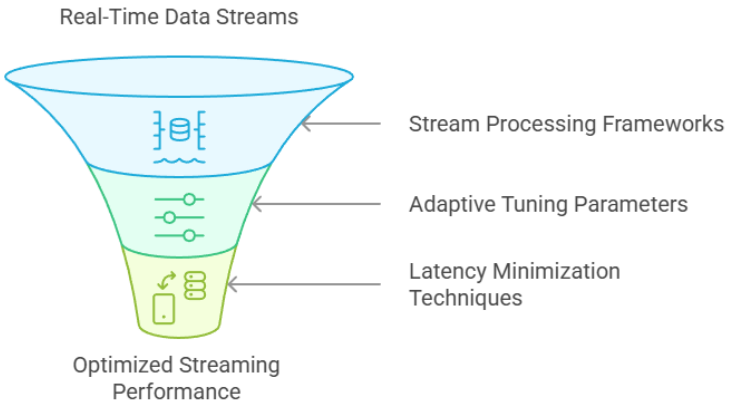

Figure 3: Optimizing Real-Time Streaming in Cloud

6. Experimental Evaluation and Case Studies

6.1. Methodology and Setup

The methodology and setup for the experimental evaluation involve systematically testing the proposed performance tuning techniques in cloud data workflows for both batch processing and real-time streaming. The evaluation framework includes defining representative workloads, selecting appropriate cloud platforms, and configuring tuning parameters such as resource allocation, data partitioning, and adaptive streaming controls. Experiments are designed to measure key performance metrics including throughput, latency, resource utilization, and cost efficiency under varying workload conditions. Case studies are incorporated to demonstrate practical applications and validate the effectiveness of optimization strategies in real-world scenarios. The setup ensures reproducibility by detailing the hardware and software environments, data sources, and monitoring tools used. Both controlled experiments and dynamic workload simulations are employed to capture system behavior and responsiveness. This comprehensive approach enables a robust assessment of tuning impacts on scalability, efficiency, and latency reduction in cloud-based data pipelines.[29], [30]

6.2. Performance Metrics and Results

The performance metrics and results section evaluates the effectiveness of the proposed tuning techniques by measuring key indicators such as throughput, latency, resource utilization, and cost efficiency across both batch processing and real-time streaming workflows. Experimental results demonstrate how different tuning parameters impact system responsiveness and scalability under varying workload conditions. Through quantitative analysis, improvements in processing speed and reductions in latency are highlighted, validating the benefits of adaptive resource allocation, load balancing, and data partitioning strategies. Case studies further illustrate practical gains in diverse operational scenarios, showing enhanced stability and cost-performance balance. Performance trends reveal the trade-offs between maximizing throughput and minimizing latency, emphasizing the importance of fine-tuning for specific cloud environments. [31]The results confirm that well-calibrated tuning can significantly optimize cloud data workflows, supporting both high-volume batch jobs and low-latency streaming applications. This empirical evidence underpins the practical guidelines proposed for cloud data optimization.

6.3. Discussion of Findings

The discussion of findings highlights that the applied performance tuning techniques significantly improve scalability, efficiency, and latency in both batch processing and real-time streaming cloud workflows. Experimental results confirm that adaptive resource allocation, load balancing, and data partitioning effectively address bottlenecks and optimize throughput under varying workloads. The findings emphasize the trade-offs between maximizing throughput and minimizing latency, demonstrating the need for careful parameter tuning based on specific workflow requirements. [27] Case studies illustrate practical benefits such as enhanced system stability, cost-effectiveness, and responsiveness in diverse operational contexts. The evaluation underscores the importance of dynamic tuning mechanisms to handle workload fluctuations and maintain consistent performance. Additionally, the results validate that integrating tuning strategies with stream processing frameworks and batch processing architectures leads to substantial gains in processing speed and resource utilization. These insights provide a robust foundation for the proposed guidelines aimed at optimizing cloud-based data pipelines. Overall, the study confirms that targeted performance tuning is critical for achieving efficient, scalable, and low-latency cloud data workflows.[32]

7. Conclusion

This paper has systematically examined performance tuning techniques essential for optimizing cloud data workflows in both batch processing and real-time streaming contexts. By addressing key challenges related to scalability, efficiency, and

latency, the study demonstrated how adaptive resource allocation, data partitioning, and load balancing can significantly enhance throughput and responsiveness. Experimental evaluations and case studies validated the effectiveness of these strategies, highlighting the necessary trade-offs between maximizing processing speed and minimizing latency while maintaining cost-effectiveness. The integration of tuning approaches with appropriate stream processing frameworks and batch architectures further reinforces the potential for dynamic optimization in diverse cloud environments. These findings provide practical guidelines that empower researchers and practitioners to achieve scalable, efficient, and low-latency cloud data pipelines, ultimately supporting timely decision-making and operational agility across various applications.

References

1. Q. Lin, B. C. Ooi, Z. Wang, and C. Yu, "Scalable Distributed Stream Join Processing," Association for Computing Machinery, May 2015, pp. 811–825. doi: 10.1145/2723372.2746485.
2. R. Alsurdeh, R. N. Calheiros, K. M. Matawie, and B. Javadi, "Hybrid Workflow Scheduling on Edge Cloud Computing Systems," IEEE Access, vol. 9, pp. 134783–134799, Jan. 2021, doi: 10.1109/access.2021.3116716.
3. D. Cheng, Y. Wang, X. Zhou, and C. Jiang, "Adaptive Scheduling Parallel Jobs with Dynamic Batching in Spark Streaming," IEEE Trans. Parallel Distrib. Syst., vol. 29, no. 12, pp. 2672–2685, Dec. 2018, doi: 10.1109/tpds.2018.2846234.
4. D. Cheng, D. Milojicic, Y. Chen, D. Gmach, and X. Zhou, "Adaptive scheduling of parallel jobs in spark streaming," Institute Of Electrical Electronics Engineers, May 2017. doi: 10.1109/infocom.2017.8057206.
5. Q. Zhang, W. Shi, Y. Song, and R. R. Routray, "Adaptive Block and Batch Sizing for Batched Stream Processing System," Institute Of Electrical Electronics Engineers, July 2016, pp. 35–44. doi: 10.1109/icac.2016.27.
6. M. Khaldi, M. Rebbah, B. Meftah, and O. Smail, "Fault tolerance for a scientific workflow system in a Cloud computing environment," International Journal of Computers and Applications, vol. 42, no. 7, pp. 705–714, July 2019, doi: 10.1080/1206212x.2019.1647651.
7. Z. Chen, X. Chen, C. Rong, B. Lin, K. Lin, and X. Zheng, "Adaptive Resource Allocation and Consolidation for Scientific Workflow Scheduling in Multi-Cloud Environments," IEEE Access, vol. 8, pp. 190173–190183, Jan. 2020, doi: 10.1109/access.2020.3032545.
8. H. Liu, W. Zhu, Y. Lu, and S. Fu, "A Trend Detection-Based Auto-Scaling Method for Containers in High-Concurrency Scenarios," IEEE Access, vol. 12, pp. 71821–71834, Jan. 2024, doi: 10.1109/access.2024.3403451.
9. Y. Li, L. Zou, M. T. Ozsu, and D. Zhao, "Time Constrained Continuous Subgraph Search Over Streaming Graphs," Institute Of Electrical Electronics Engineers, Apr. 2019. doi: 10.1109/icde.2019.00100.
10. J. K. Konjaang and L. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing," J Cloud Comp, vol. 10, no. 1, Jan. 2021, doi: 10.1186/s13677-020-00219-1.
11. J. Cervino, E. Kalyvianaki, P. Pietzuch, and J. Salvachua, "Adaptive Provisioning of Stream Processing Systems in the Cloud," Institute Of Electrical Electronics Engineers, Apr. 2012, pp. 295–301. doi: 10.1109/icdew.2012.40.
12. A. S. Rajawat, S. B. Goyal, V. Malik, and M. Kumar, "Adaptive resource allocation and optimization in cloud environments: Leveraging machine learning for efficient computing," Crc, 2024, pp. 499–508. doi: 10.1201/9781003471059-64.
13. C. Xu, M. Kaul, V. Markl, and M. Holzemer, "Efficient fault-tolerance for iterative graph processing on distributed dataflow systems," Institute Of Electrical Electronics Engineers, May 2016, pp. 613–624. doi: 10.1109/icde.2016.7498275.
14. M. Junaid et al., "Modeling an Optimized Approach for Load Balancing in Cloud," IEEE Access, vol. 8, pp. 173208–173226, Jan. 2020, doi: 10.1109/access.2020.3024113.
15. M. Mudassar, L. Lejian, and Y. Zhai, "Adaptive Fault-Tolerant Strategy for Latency-Aware IoT Application Executing in Edge Computing Environment," IEEE Internet Things J., vol. 9, no. 15, pp. 13250–13262, Aug. 2022, doi: 10.1109/jiot.2022.3144026.
16. K. Ren, A. Thomson, T. Diamond, and D. J. Abadi, "Low-Overhead Asynchronous Checkpointing in Main-Memory Database Systems," Association for Computing Machinery, June 2016, pp. 1539–1551. doi: 10.1145/2882903.2915966.
17. N. Jain, J. Naor, I. Menache, and J. Yaniv, "A Truthful Mechanism for Value-Based Scheduling in Cloud Computing," Theory Comput Syst, vol. 54, no. 3, pp. 388–406, Feb. 2013, doi: 10.1007/s00224-013-9449-0.
18. H. Zheng, M. Zhang, H. Li, H. Tan, and K. Xu, "Efficient resource allocation in cloud computing environments using AI-driven predictive analytics," ACE, vol. 82, no. 1, pp. 17–23, Sept. 2024, doi: 10.54254/2755-2721/82/2024glg0055.
19. Z. Chen, C. Luo, J. Hu, T. El-Ghazawi, and G. Min, "Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement Learning," IEEE Trans. Parallel Distrib. Syst., vol. 33, no. 8, pp. 1911–1923, Aug. 2022, doi: 10.1109/tpds.2021.3132422.
20. K. Lee and L. Liu, "Scaling queries over big RDF graphs with semantic hash partitioning," Proc. VLDB Endow., vol. 6, no. 14, pp. 1894–1905, Sept. 2013, doi: 10.14778/2556549.2556571.
21. Y. Mao, X. Li, and X. Chen, "Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing," Springer India, 2014, pp. 457–465. doi: 10.1007/978-81-322-1759-6_53.

22. M. O. Oyediran, O. Aiyeniko, M. O. Adigun, P. Chima Obuzor, O. S. Ojo, and S. A. Ajagbe, "Comprehensive review of load balancing in cloud computing system," IJECE, vol. 14, no. 3, p. 3244, June 2024, doi: 10.11591/ijece.v14i3.pp3244-3255.
23. L. Amini, A. Sehgal, J. Silber, O. Verscheure, and N. Jain, "Adaptive Control of Extreme-scale Stream Processing Systems," Institute Of Electrical Electronics Engineers, Jan. 2017, p. 71. doi: 10.1109/icdcs.2006.13.
24. J.-H. Hwang, S. Zdonik, A. Rasin, M. Balazinska, M. Stonebraker, and U. Cetintemel, "High-Availability Algorithms for Distributed Stream Processing," Institute Of Electrical Electronics Engineers, Apr. 2005, pp. 779–790. doi: 10.1109/icde.2005.72.
25. T. Das, I. Stoica, Y. Zhong, and S. Shenker, "Adaptive Stream Processing using Dynamic Batch Sizing," Association for Computing Machinery, Nov. 2014, pp. 1–13. doi: 10.1145/2670979.2670995.
26. J. Rane, Ö. Kaya, N. L. Rane, and S. K. Mallick, "Scalable and adaptive deep learning algorithms for large-scale machine learning systems," Deep Science, 2024. doi: 10.70593/978-81-981271-0-5_2.
27. G. Van Dongen and D. Van Den Poel, "Evaluation of Stream Processing Frameworks," IEEE Trans. Parallel Distrib. Syst., vol. 31, no. 8, pp. 1845–1858, Aug. 2020, doi: 10.1109/tpds.2020.2978480.
28. M. Ghasemi, A. Mansy, P. Kanuparthy, J. Rexford, and T. Benson, "Performance Characterization of a Commercial Video Streaming Service," Association for Computing Machinery, Nov. 2016, pp. 499–511. doi: 10.1145/2987443.2987481.
29. R. Han et al., "Workload-Adaptive Configuration Tuning for Hierarchical Cloud Schedulers," IEEE Trans. Parallel Distrib. Syst., vol. 30, no. 12, pp. 2879–2895, Dec. 2019, doi: 10.1109/tpds.2019.2923197.
30. M. Zhang, B. Yuan, K. Xu, and H. Li, "LLM-Cloud Complete: Leveraging Cloud Computing for Efficient Large Language Model-based Code Completion," JAIGS, vol. 5, no. 1, pp. 295–326, Aug. 2024, doi: 10.60087/jaigs.v5i1.200.
31. V. Bhimanapati, S. Jain, and O. Goel, "Cloud-Based Solutions for Video Streaming and Big Data Testing," URR, vol. 10, no. 4, pp. 329–345, Dec. 2023, doi: 10.36676/urr.v10.i4.1333.
32. M. Bilal and M. Canini, "Towards automatic parameter tuning of stream processing systems," Association for Computing Machinery, Sept. 2017, pp. 189–200. doi: 10.1145/3127479.3127492.