Cost optimization and performance control in the hybrid multi-cloud environment

Boris Zibitsker, PhD,
BEZNext, Chicago, IL

Alex Lupersolsky, PhD,
BEZNext, Chicago, IL

ABSTRACT

Escalating cloud cost and unstable performance is a typical challenge for any organization. It is especially related to deployment of new Generative AI (GenAI) applications in a hybrid multi-cloud world, which projected to become a nearly twenty trillion dollars market by 2030.

This paper introduces a novel systems approach to optimizing cost, controlling performance, and improving FinOps decision-making through automated observability, advanced queueing network modeling, and gradient optimization.

Observability automation narrows the scope of the tuning efforts by focusing on the most resource-consuming and credit use applications, applications with the highest rate of performance and cost anomalies, applications with the highest frequency of failed queries, and applications with the highest volume of data spilled to local and remote storage.

Modeling and optimization determine the minimal configurations, resource allocation, workload management, and budgets needed to meet Service Level Goals (SLGs) for all business applications running on different cloud data platforms in the Hybrid Multi-Cloud environment. Modeling and optimization evaluate options and set cost and performance expectations for proposed changes.

Actual performance and cost compared with expected to organize a closed loop performance and cost control and mitigate risks of unexpected financial and performance outcomes.

Presented case studies illustrate the value of our technology in optimizing application costs and controlling performance for a wide range of projects, such as sizing new applications before deployment to the cloud, appropriate cloud platform selection, optimizing cloud migration decisions, and organizing dynamic capacity management applications in the Hybrid Multi-Cloud environment.

We demonstrated a high accuracy of our predictions. The difference between measured and predicted cost is within 10%.

CCS CONCEPTS

Hybrid Multi-Cloud; Cost optimization; *Performance control*; *Observability*; *Modeling*; *Gradient optimization*.

KEYWORDS

Hybrid multi-cloud, cost optimization, performance control, Generative AI, Hybrid Multi-Cloud, FinOps, observability, modeling, closed-loop feedback control, gradient optimization.

ACM Reference Format:

Boris Zibitsker and Alex Lupersolsky. 2025. Cost optimization and performance control in the hybrid multi-cloud environment. In *Proceedings of the 2025 ACM/SPEC International Conference on Performance Engineering (ICPE '25)*, May 7–11, 2025, Toronto, Canada.

Introduction

Motivation for this work

Organizations running business applications in the cloud face dual challenges: how to reduce cost and maintain consistent performance. Rising volume of data, increasing number of users, migrations applications to the cloud and deployment new GenAI applications often dramatically increase the costs and affect performance. Critical decisions - such as selecting cloud platforms, cloud migration, proactively managing Hybrid Multi-Cloud environments and sizing new applications before deployment - must be optimized to minimize budget overruns and avoid performance inconsistencies.

A related work

FinOps methodology and technology is widely used to optimize cost by organizing observability, modeling and control [1]. Storment and Fuller provide a detailed overview of the FinOps principles. They emphasize the importance of continuous observability and feedback loops for cost and performance [15].

FinOps implementation has limitations and challenges. It includes complexity in implementing and managing FinOps in the hybrid multi-cloud environments, and complexity in organizing monitoring and optimization. Inconsistent metrics generated by cloud providers make unified observability across multi-cloud environments complex.

Current FinOps observability does not use the system approach and lack automation. Proposed recommendations do not include cost-performance expectations and do not allow customers to check if implemented recommendations were successful. [2].

Articles [3,4,5,6] offer a detailed framework for managing cloud costs in hybrid environments, focusing on three stages: Inform (visibility), Optimize (cost-saving opportunities), and Operate (continuous adjustment and automation). They

also emphasize the role of accountability and real-time insights for optimizing cloud spend. FinOps limitations reported there pose risks by failing to determine the minimum resource requirements, workload management, and budget needed to meet the SLGs of diverse applications across the Hybrid Multi-Cloud environment. The technologies described in these papers determine cost and performance anomalies and root causes, but do not show the related queries, the most critical and costly queries consuming most of the resources, queries having the largest amount of failures, and queries spilling the large volume of data to remote and local storage. Several tools provide tuning recommendations, but do not estimate the expected impact of proposed changes on cost and performance.

Modeling and simulation approaches presented in [16] provide definitions and metrics for elasticity in cloud environments and present a conceptual model for evaluating elastic resource provisioning.

Applying modeling and simulation to optimize **r**esource allocation and scheduling in the Hybrid Multi-Cloud environment described in [17]. They propose a predictive model using time-series forecasting and reinforcement learning to allocate resources across multiple clouds**.** It demonstrates how ML can optimize real-time analytics pipelines, improving both latency and cost efficiency. Survey [18] presents techniques for workload orchestration in hybrid clouds, covering real-time vs. batch workloads, cost vs. performance objectives, and vendor lock-in considerations.

Models described in these papers help to evaluate different options, but they are limited in evaluation of the combined impact of the configuration and workload management changes on cost in the Hybrid Multi-Cloud environment with mixed workloads having different profiles and SLGs.

Several papers like [19] discuss the application of performance modeling for query and database tuning. They leverage deep reinforcement learning to automatically adjust caching, partitioning, and indexing parameters for distributed SQL engines. [20] proposes a learned cost model integrated into a cloud-based query optimizer, adapting to resource heterogeneity and changing data characteristics. It Illustrates that data-driven models can significantly improve query planning in dynamic cloud environments, reducing query runtimes and cost. The approach described focuses on enhancement of DBMS optimizers but has a limited application for tuning queries and databases in customers' cloud environments.

Increasing number of papers illustrate a growing emphasis on machine learning - particularly reinforcement learning - and multi-objective optimization for cloud DB tuning, resource allocation, and workload management. As hybrid and multi-cloud adoption increases, novel cost and performance models are emerging to handle the complexity of distributed resources, diverse pricing schemes, and real-time workload fluctuations.

Multi-Objective Optimization recognizes that trade-offs among cost, response time, energy efficiency, carbon footprint, and fault tolerance as additional objectives are essential [21]. Evolutionary algorithms or Pareto-based methods are frequently applied. Beyond cost and performance, they include energy efficiency, carbon footprint, or fault tolerance as additional objectives.

We currently focus on cost optimization and performance control, but we provide an option to take into consideration the estimated electrical power consumption and carbon footprint of individual applications on different cloud data platforms [22].

Paper [23] proposes applying Multi-Cloud Resource Management and Orchestration. It *i*ntroduces a dynamic data-movement algorithm using a cost-performance model that forecasts future query patterns and determines whether on-prem or cloud storage is more economical given expected performance needs. This paper focuses on data movement. With the introduction of the Open Table Format customers are concerned with where to run applications rather where to place data.

Many other research and industry papers underscore the current limitations in applying observability automation and incorporating modeling to optimize cost and control performance in the complex Hybrid Multi-Cloud AI world.

1. The novelty and superiority of our approach

In this paper, we present our novel holistic approach based on applying SLGs for optimizing application costs and controlling performance in the complex Hybrid Multi-Cloud AI world.

We automate the observability. Our agents continuously collect performance, resource usage, cost and data usage data from cloud data platforms logs. Measurement data aggregated and used for workload characterization and performance and cost analysis.

We leverage AI and machine learning to identify performance, resource utilization and cost anomalies and seasonal peaks, and recommend cost-saving measures. We detect critical queries using most of resources, having the largest number of failures and spilling the largest volume of data to local and remote storage. It narrows the scope of tuning efforts by focusing on the most critical applications, queries, tables, and databases.

Then we use modeling and gradient optimization to find the minimum configuration, resource allocation and workload management changes needed to meet business performance SLG for all applications with the lowest cost and predict the expected cost and performance metrics' values.

It allows us to organize a closed-loop cost optimization and performance feedback control for all business applications in a Hybrid Multi-Cloud AI environment.

Our methodology and proven technology help to reduce uncertainty and the risk of financial and performance

surprises during all phases of the journey to the cloud, starting with selecting the appropriate cloud data platform, optimizing cloud migration decisions, optimizing dynamic capacity management in the Hybrid Multi-Cloud environment, and sizing new applications before deployment in the cloud.

2. Observability Automation and Dynamic Capacity Management

Our methodology is implemented for AWS, GCP, and Azure cloud service providers and Snowflake, Vantage Cloud, Databricks, BigQuery, and Synapse cloud data platforms.

In this paper, we use Snowflake on AWS as an example to demonstrate how automating observability and applying advanced modeling techniques optimize cost and organize performance control.

Through automated data collection and workload characterization, we generate hourly performance, resource utilization, data usage, and cost profiles for each application.

Observability captures changes in configuration and workload management parameters. By automating observability processes, we provide important insights into each application's resource demand and usage patterns, which are essential for optimizing resource allocation and workload management in a complex hybrid multi-cloud environment.

We will review several examples illustrating how observability results are used to optimize tuning and dynamic capacity management:

1. **Resource Demand Patterns:** Automated observability helps track hourly, daily, and other seasonal patterns (Figure 1) in resource demand, enabling proactive adjustments to server capacity. For example, high-traffic periods may require additional virtual warehouses or increased memory allocation to prevent latency issues and meet SLGs.
2. **Performance Bottleneck Identification:** Observability data allows for the early detection of performance bottlenecks, often caused by workload spikes, query concurrency, or suboptimal configurations. By identifying the applications with the highest consumption or failure rates, targeted adjustments can be made to configuration and workload distribution, aligning resources with actual usage demands.
3. **Spillage Monitoring and Management:** One of the key indicators captured is data spillage—when allocated memory is insufficient, causing overflow to local or remote storage. Observability data reveals the volume of spillage, which defines memory allocation adjustments or storage configuration enhancements to minimize performance impacts and control associated costs.
4. **Real-Time Tuning Opportunities:** Automated observability also provides real-time insights for tuning efforts. By continuously profiling high-cost queries and those causing the most frequent anomalies are identified as tuning candidates. Tuning efforts are then focused on these critical queries to improve efficiency and cost-effectiveness.

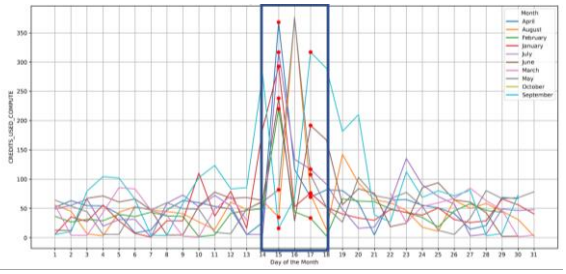

Figure 1. Example of seasonality in credit use by one of the business applications.

These observability-driven insights allow for the organization of dynamic capacity management, reducing the likelihood of unexpected costs or performance degradations. By continuously adjusting capacity based on real-time usage patterns, we establish a foundation for effective, adaptive resource management in complex hybrid multi-cloud environments.

2.1. Focus on Tuning Queries Using the Largest Number of Credits

Our approach begins by identifying the applications with the highest cost on each cloud data platform. Then, we narrow our focus to the critical queries within these applications, which have the longest execution times, as prime candidates for tuning.

Export Show 10 entries Search:

Application Name	Total Credits Used	# of Total warehouses
Business Application name	97,641	24
	67,875	15
	64,935	43
	47,414	21
	41,085	16
	28,393	4
	26,417	13
	16,551	12
	16,458	20
	14,104	7

Showing 1 to 10 of 242 entries 2 3 4 5 .. Next

Table 1: Credit usage by top 10 Snowflake applications during the last month

This trend is projected to continue, with substantial cost increases anticipated in the coming year. The top ten business applications in Table 1, account for over 70% of total credit usage, indicating a concentrated area for cost optimization.

The observability data further reveals the monthly distribution of credit usage across different virtual warehouses, providing insights into specific performance bottlenecks. Table 2 captures the hash values and

performance statistics for the top 30 queries, which are prioritized for analysis and tuning based on execution time and credit usage.

HASH_VALUE	EXECUTIONS COUNT	SUM ELAPSED TIME SEC	AVERAGE ELAPSED TIME SEC	SUM EXECUTION TIME SEC	AVERAGE EXECUTION TIME SEC	SUM MB PROCESSED	AVERAGE MB PROCESSED
a10b9ba530daf2ae3b28846799328 83b	15	1,457,462	97,164	1,457,400	97,160	21,913,782	1,460,919
fa44f1ce4d1c24327b5106346d96c42a	15	1,413,483	94,232	1,413,406	94,227	21,902,721	1,460,181
0c8b97d6ac38433810c22fb328f5227b	3	518,403	172,801	518,397	172,799	8,946	2,982
6abe18fb17cf3ed65c42b234499cf57b	21	402,538	19,168	402,502	19,167	95,550,978	4,550,047
52470da7dcb0637061b8cc6e54e8b078	9	228,436	25,382	228,434	25,382	15,413	1,713
13c3476270759437ab40048bac2094c3	4	173,316	43,329	173,314	43,329	6,264	1,566
c4fd2f41d5a508c785b69981d6305efc	1	172,802	172,802	172,796	172,796	41,567	41,567
db4508ea5a03b2625b6e40594f5fa4aa	4	85,936	21,484	85,913	21,478	3,655,056	913,764
e599e9cafe23ec72e74588d5544c41b2	2	77,890	38,945	77,889	38,945	98,632	49,316
f86096fe3d5583993671d146a11c5ab9	1	76,873	76,873	76,873	76,873	0	0
e596bae649b6649b6c3602d939a7d09a	1	72,863	72,863	72,863	72,863	126,563	126,563
f749fdbbfb15c9d5a213a0b3875bc6db	1	65,599	65,599	65,599	65,599	0	0
343597c7eddd80c26b49ea6806f63b0947	3	65,190	21,730	65,188	21,729	0	0
06935534d8deaf57486b5cdff50815dbf0	2	64,166	32,083	64,162	32,081	2,111,648	1,055,824
98005568b9e2d6659df3699ea1fbaed5f	1	63,456	63,456	63,456	63,456	0	0
c2ffb5aea33e24e80822116b6421cc0d	1	62,482	62,482	62,481	62,481	183,991,416	183,991,416
e84c3b115337c175029ca791bb787b4f	1	60,560	60,560	60,560	60,560	365,734	365,734
98224147b51c2a97f0054410453665d	2	59,189	29,595	59,181	29,590	3,223,870	1,611,935
7887d4660e7afd1d5d0465f9be6216d7	1	58,516	58,516	58,514	58,514	1,764,412	1,764,412
c0c230aa0f86d9282a454039b1af6da1	1	58,495	58,495	58,494	58,494	81,051	81,051
89f52c15f380799ed6d2849a6bdaf0a5	2	57,441	28,720	57,440	28,720	0	0
266884d5f8edb31b4aea99ffd3df377e	1	56,586	56,586	56,585	56,585	0	0
abc78a0552b1f7131c43805e55b9bd34	1	55,920	55,920	55,920	55,920	61,112	61,112
83405fb1cb03b5120e890350283ae045	1	54,789	54,789	54,789	54,789	59,858	59,858
71e3b92972cee4f60c1d61834b98a6c3	1	53,794	53,794	53,794	53,794	269,340	269,340
eb1107637c5615c444069cd106bb2f2e	1	53,740	53,740	53,740	53,740	0	0
16058501fb6dfebe9b5a2d9e76611b7b	1	53,398	53,398	53,398	53,398	0	0
d119fec6692aff0551128a5f197773b0	1	51,995	51,995	51,994	51,994	8,734,360	8,734,360
028477f962f94b588b4ff1d5583a2108	1	51,742	51,742	51,741	51,741	185,894	185,894

Table 2: Top 30 queries by hash values with highest execution times (candidates for tuning)

Database administrators (DBAs) can utilize various tools to analyze access paths and identify specific tuning options, allowing for a more targeted and efficient tuning process. By optimizing these high-cost queries, organizations reduce overall credit usage and improve performance for key applications, directly impacting the scalability and efficiency of their cloud data platforms.

We also automatically segregate queries into simple, medium, and complex categories - each with distinct cost and performance profiles.

For each group we determine the average response time. Response time for each group is used as SLG like:

- Small Group: 8.23 seconds
- Medium Group: 20.84 seconds
- Complex Group: 74.97 seconds

Observability results indicate that simple queries tend to cause the most spillage to local storage, while complex queries predominantly spill data to remote storage. These spillage patterns are factored into our modeling to determine each query group's optimal configuration and budget allocation. By doing so, we ensure that SLGs are met continuously across varying times and days of the month.

Using these data-driven insights, we can proactively allocate resources based on the specific service demands of each query group. This approach helps optimize capacity while managing costs, as tuning efforts are concentrated on the queries that most heavily influence performance and expenditure.

Focusing on the most resource and credit use applications and determining tuning recommendations for them increases the effectiveness of the cost optimization and performance control and complex hybrid multi-cloud environment supporting hundreds of applications on different cloud data platforms.

2.2. Focus on tuning queries causing frequent and severe cost and performance anomalies

Machine learning (ML) models detect application performance and cost anomalies, flagging applications where actual metrics deviate significantly from predicted values (Figure 2).

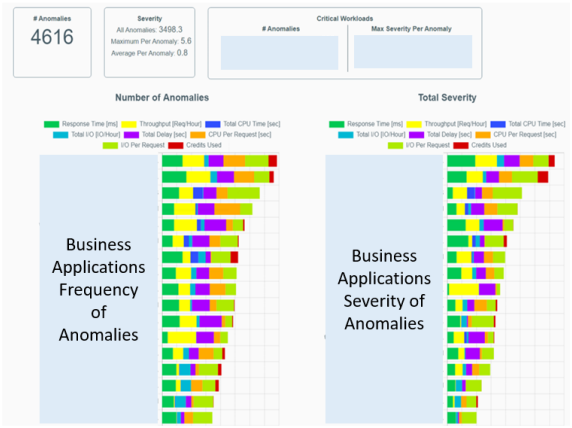

Figure 2: Frequency and severity of applications performance and cost anomalies

We analyze each detected anomaly's root causes, identifying critical SQL statements and the specific database tables involved. This analysis allows us to understand the underlying factors driving the anomaly, such as inefficient query patterns or suboptimal resource allocation, which can then be prioritized for tuning.

By continuously monitoring application performance and costs, we can pinpoint the queries responsible for performance degradation and cost increases.

Another contributor to performance issues and cost escalation in cloud environments is data spilling. Table 3 shows an example of 10 top queries for one of the applications with the largest number of MB spilled to the local storage.

2.3 Focus on Tuning Queries Generating the Largest Data Spill to Local and Remote Storage

Factors Affecting Data Spillage

Data spilling to local or remote storage typically results from memory limitations during query processing. When a query's allocated memory cannot accommodate all intermediate data, the system spills it to local storage. However, if local storage is insufficient or under high load, the system further spills to remote storage to maintain query completion. Remote storage is a flexible, cost-effective extension, enabling complex queries to proceed even when memory resources are stretched.

Joins and Aggregations: Complex queries with multiple joins or aggregations often generate large intermediate results that surpass available memory. In these cases, the

system offloads excess data to remote storage to continue processing without interruption.

HASH VALUE	TOTAL EXECUTION COUNT	FAILED EXECUTION COUNT	FAILED EXECUTION PERCENTAGE	TOTAL EXECUTION TIME SEC	FAILED EXECUTION TIME SEC	FAILED EXECUTION TIME	TOTAL MB PROCESSED	FAILED MB PROCESSED	FAILED MB PROCESSED PERCENTAGE	MB SPILLED LOCAL	MB SPILLED REMOTE
c1f8ecb56428a5e71d46bc35395b1a7e	7	0	0	73,452	0	0	8,722,099	0	0	54,170,485	5,562,299
73a0f4b9b19beeea68086907c15ad1d5	6	0	0	69,675	0	0	3,652,305	0	0	29,991,497	4,008,485
5e4876ce59128c22d21e3c7181fb880e	1	0	0	2,174	0	0	85,430,321	0	0	16,560,717	0
3161212a0c3617d9aedcf34ab3dad1ad	16	0	0	103,795	0	0	40,114,757	0	0	12,228,839	0
0dbd5fdbf4e8c6fb989c98e879b77109	1	0	0	6,564	0	0	1,942,581	0	0	12,228,324	0
7338d684cd5b67227f12198cf330922a	1	0	0	28,590	0	0	1,866,494	0	0	10,782,943	6,528,053
d67e0439636e28732e2367d24c45cfdb	4	1	25	7,466	3,597	48	2,186,553	566,485	26	10,718,302	0
4443522a4edeeca9daa74a066b94371c	1	0	0	24,737	0	0	858,763	0	0	9,860,362	2,207,928
596a5d61530feca7031860ab3e0400bc	1	0	0	32,293	0	0	671,607	0	0	9,856,752	0
f281540af78e5fa856c35e7362fe0185	1	0	0	8,250	0	0	1,135,930	0	0	8,719,966	2,976
e9c9ab259686be08f1d8fd1bbb6092ca	16	0	0	5,000	0	0	2,596,923	0	0	7,342,377	0

Table 3. ***An example of 10 queries for one of the applications with the largest number of MB spilled to the local storage***

High Query Concurrency: When multiple queries run concurrently during peak usage, memory and local storage demands increase sharply. Spilling to remote storage helps.

Skewed Data Distribution: In distributed systems, data partitions across multiple nodes can sometimes become unevenly distributed or "skewed." This condition can result in some nodes handling disproportionately larger data volumes, exhausting local storage and leading to higher spill rates on these nodes.

Insights from Observability Data

Analyzing observability data provides valuable insights into spillage patterns, which often vary by time of day and workload. In this example the highest data scan volumes tend to occur during nighttime operations (*Figure 3*). While compute clusters allocations are minimal at night (*Figure 4*) and credit usage is at its lowest (*Figure 5*), this period also shows the highest data spillage to local storage (*Figure 6*), with average elapsed times exceeding the SLG threshold of 75 seconds (*Figure 7*).

Resource Requirements to Meet SLGs

Significant resources may be required to meet SLGs, especially for complex query groups with an SLG of 75 seconds. For example, meeting this SLG often demands up to 3,500 GB of RAM and 218 nodes (typically organized as seven 2XL clusters), as depicted in *Figure 8*.

Proactive Spillage Management

Excessive spillage is a critical indicator of resource constraints or suboptimal query design, leading to degraded performance and increased costs. By monitoring spillage data, especially for large or complex queries, our observability process enables targeted interventions, such as:

- **Memory Optimization:** Adjusting memory allocations to better match query requirements.
- **Resource Allocation Adjustments:** Fine-tuning resource distributions to accommodate high-demand periods and complex workloads.
- **Query Simplification:** Breaking down complex queries into smaller tasks, reducing the likelihood of spillage.

Proactive tuning of queries with high spillage rates thus enhances system performance, reduces query failure rates, and achieves cost savings by optimizing resource utilization.

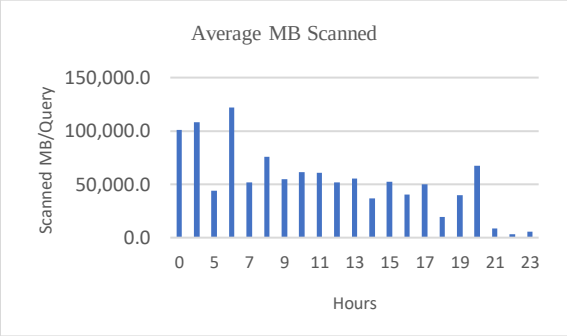

Figure 3: High data scan volume per query during nighttime

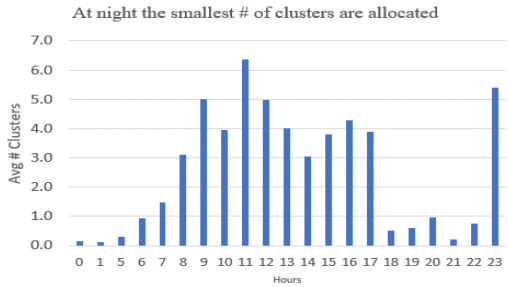

Figure 4: Minimal cluster allocations at night

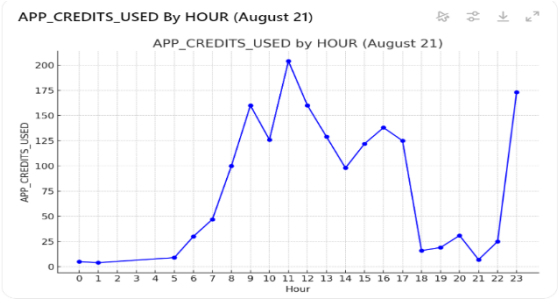

Figure 5. Low credit usage during nighttime, with peaks during daytime and late evening

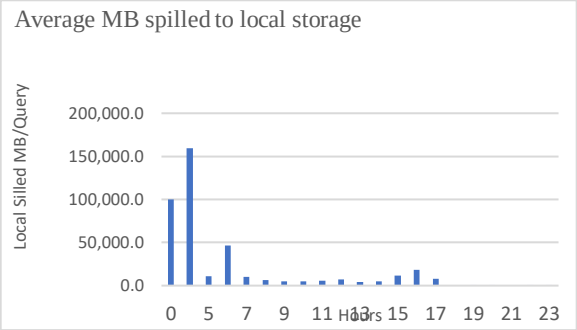

Figure 6: Maximum data spillage to local storage at night

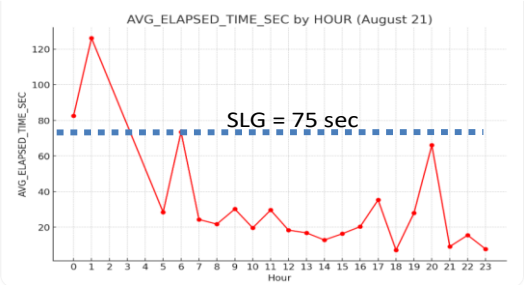

Figure 7: SLG breaches due to excessive nighttime spillage

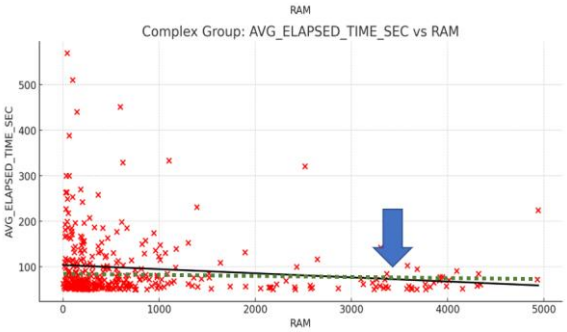

Figure 8. To meet SLG for the complex group (75 sec), 3,500 GB of RAM and 218 nodes (seven 2XL clusters) are needed

2.4 Tuning Queries with the Largest Number of Failures

We analyze SQL query failures for each application by examining the frequency of errors, error codes, error messages SQL hash values, and the associated lost execution time and credits. Additional parameters such as data processed and data spilled into local and remote storage, elapsed time, execution time, wait time, and throughput per application are also evaluated. By aggregating these metrics, we gain insights into how the failures contribute to resource waste and credit loss across applications.

Table 4 lists hash values and error codes for ten representative queries with the longest failed execution times, which guide DBAs in tuning these queries to reduce failures.

By targeting high-failure queries, we enhance overall system efficiency, reduce unnecessary cost expenditures, and improve application reliability. This tuning process mitigates financial waste and optimizes the allocation of compute and storage resources, further aligning performance with SLGs.

HASH VALUE	ERROR_CODE	ERROR_MESSAGE_SNIPPET	TOTAL EXECUTION COUNT	FAILED EXECUTION COUNT	FAILED EXECUTION PERCENTAGE	TOTAL EXECUTION TIME SEC	FAILED EXECUTION TIME SEC	FAILED EXECUTION TIME PERCENTAGE	TOTAL MB PROCESSED	FAILED MB PROCESSED	FAILED MB PROCESSED PERCENTAGE
eb1107637c5615c444069cd106bb2f2e	604	Uncaught exception of type 'STATEMENT_ERROR' on line 132 at position 0 : SQL execution canceled	1	1	100	53,740	53,740	100	0	0	0
d119fec6692aff0551128a5f197773b0	604	SQL execution canceled	1	1	100	51,994	51,994	100	8,734,360	8,734,360	100
aa58ab8fe9aa7b9cd77e9e0c3f53737a	630	Statement reached its statement or warehouse timeout of 7200 second(s) and was canceled.	2	2	100	14,399	14,399	100	287,178	287,178	100
ce452e9a02158241b84a2202c9c5ee36	100114	Uncaught exception of type 'STATEMENT_ERROR' on line 565 at position 0 : The following string is not	4	4	100	12,839	12,839	100	0	0	0
73ab0c7cd3efbf4f98e087ba8f9d0323	630	Statement reached its statement or warehouse timeout of 7200 second(s) and was canceled.	1	1	100	7,199	7,199	100	226,614	226,614	100
d8f65e07d226af11fc68e6875bd819d9	630	Statement reached its statement or warehouse timeout of 7200 second(s) and was canceled.	1	1	100	7,198	7,198	100	1,028,552	1,028,552	100
18eec32418cb6d7aaace0c64c7771f49	100114	Uncaught exception of type 'STATEMENT_ERROR' on line 565 at position 0 : The following string is not	4	4	100	5,643	5,643	100	0	0	0
5d7e5bc8a26ab44ec0a29bcf53ccf0a4	604	SQL execution canceled	1	1	100	5,606	5,606	100	226,522	226,522	100
fdf6fdfcfc48fec093ab2cbca5a25320	604	SQL execution canceled	2	2	100	4,854	4,854	100	384,486	384,486	100
22f3ab508a1a097c79106733a6d70fe9	100315	Executing Stored Procedure progress timed out after 3600 seconds in function PYTHON_WORKSHEET with h	1	1	100	3,713	3,713	100	0	0	0

Table 4. Hash values of the critical queries with the largest failed execution time provide value for application developers and DBAs

2.5 Value of Observability Automation

Automating observability in cloud environments provides substantial value for optimizing cost and performance. We automate data collection across business applications on multiple cloud data platforms, perform workload characterization and build detailed performance, resource utilization, data usage, and cost profiles on an hourly basis.

We also automatically detect application performance and cost anomalies and generate periodic reports.

Monthly analyses identify seasonal trends, patterns in resource usage, and critical SQL queries. Applying machine learning techniques, including regression analysis, clustering, and classification, allows us to uncover correlations among various factors, highlighting the

applications, queries, and databases that substantially impact performance and costs.

In addition, observability automation enables us to accurately assess the minimum memory requirements necessary to prevent excessive data spillage into local and remote storage. By adjusting memory allocations based on observed usage patterns, we can reduce the negative impacts of spillage on both performance and cost.

Overall, automating observability significantly enhances the effectiveness of tuning efforts by focusing on critical performance and cost issues within complex hybrid multi-cloud environments that host numerous business workloads.

By understanding the variations in resource demands, organizations can optimize resource allocation throughout the day, achieving improved cost efficiency and alignment with SLGs. This proactive approach allows for more efficient performance management and cost control, minimizing unexpected financial and operational risks.

3. Modeling and Optimization

Our approach employs universal queueing network models (QNM) and gradient optimization techniques applicable to performance evaluation across different cloud data platforms and configurations in a hybrid multi-cloud environment. This methodology is designed to identify the minimal configurations, tuning strategies, workload management adjustments, and budgets necessary to meet business SLGs for applications across all platforms. Using observability data, we continuously create and calibrate hourly models used to evaluate various configuration and resource allocation scenarios, providing insights that balance cost and performance control.

The results of our modeling process establish clear performance and cost expectations, enabling the implementation of a continuous, closed-loop feedback control system. This system verifies the outcomes of recommended configurations and enables ongoing optimization by continuously comparing actual performance and costs against predicted benchmarks.

We evaluated multiple modeling approaches and algorithms, including simulation modeling, generative AI, and queueing network models. While simulation models offer high accuracy, they are time-expensive to build and apply. Generative AI models, although promising, require extensive training on diverse measurement data that may not encompass all scenarios. Therefore, we selected queueing network models combined with gradient optimization to achieve the desired accuracy and efficiency for continuous SLG compliance across applications.

3.1 Queueing Network Model

Our approach is distinct in that it integrates analytical QNMs [7] into each step of gradient optimization, allowing us to find the minimum configuration, workload management and cost needed to meet SLGs for each workload.

To model the hybrid multi-cloud architecture effectively, we developed a multi-tier, multi-server QNM, where each server is represented by a separate QNM. The output from one server serves as the input to another in a cascading manner, continuing iteratively until the predicted performance metrics stabilize.

Each server's QNM comprises multiple nodes representing hardware resources, such as compute nodes, storage arrays, and interconnecting channels. Workload requests enter these queues, waiting for service according to the configuration at each node.

When moving workloads between platforms, we convert CPU service time and I/O operation counts to equivalent values on the target platform, accounting for differences in database management system (DBMS) optimizer efficiency, server software, and CPU performance. These conversion coefficients are derived from TPC-DS and BEZNext's customized benchmark tests. [24]

We adapted the Mean Value Analysis (MVA) algorithm [8] to model workload management parameters such as priorities, concurrency levels, and resource utilization limits. The iterative, approximate MVA algorithm allowing queue sizes to converge quickly to steady-state values is a more efficient approach for complex multi-chain queuing networks, where workloads share resources and requests are processed concurrently. This approach accommodates various service time distributions for processor-sharing nodes and exponential distributions for FIFO queues, providing a practical approximation of real-world computing processes.

The QNM is calibrated to maintain accuracy by comparing predicted and actual measured response times and adjusting model parameters accordingly.

The optimization process aims to identify the server configuration that minimizes cost while satisfying SLGs. Our two-step optimization algorithm includes:

1. **Redistribution of Existing Resources:** In this first step, we redistribute resources among workloads by setting workload priorities based on each workload's average response time relative to SLGs. This approach maximizes resource use within the existing configuration.
2. **Resource Allocation Adjustment:** If resource redistribution alone cannot meet SLGs, the second step uses gradient optimization to estimate the additional resources required. Conversely, if workloads exceed SLGs, the algorithm calculates how much resources can be freed, reducing costs without compromising SLG adherence.

3.2 Gradient Optimization

Our optimization algorithm [9] begins by evaluating the current or proposed hardware configuration. The QNM assesses server performance metrics, focusing on the average request response time and its components across server resources for each workload. These response times are then compared with the SLGs to determine the necessary configuration adjustments. The ratio of predicted response time to the corresponding SLG indicates the configuration change's amplitude, while response time components weights provide the direction of optimization. The optimization process is illustrated on Figure 9.

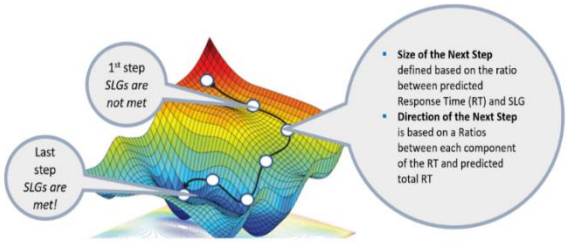

Figure 9: BEZNext Queueing Network Modeling and Gradient Optimization Process

The following steps outline the observability, modeling and optimization process:

1. **Build Workload Profiles:** Compile detailed profiles for each business workload, capturing key metrics, including cost, performance, resource utilization, and data usage for each interval.
2. **Select Representative Time Intervals:** Choose time intervals that reflect typical workload behaviors and patterns, ensuring that the model captures peak and off-peak conditions.
3. **Modeling and Optimization:** Use QNM and gradient optimization to determine the minimum configuration and workload management adjustments needed to meet SLGs for each workload at different times of day and throughout the year.
4. **Budget Calculation:** Estimate costs for the optimized configurations using specific cloud provider pricing models, ensuring budget alignment with performance requirements.
5. **Provide Performance and Cost Expectations:** Deliver realistic expectations for performance, resource utilization, and cost based on the optimized configurations, enabling proactive cost and performance management.
6. **Verify results:** Compare measurement cost and performance with expected

This process creates a robust framework for modeling and optimization, allowing organizations to achieve cost-effective performance control in complex hybrid multi-cloud environments. By continuously refining configurations and resource allocations, this approach supports ongoing FinOps decision optimization, aligning operational costs with performance goals while accommodating fluctuations in workload demands.

3.3 Optimization scenarios

3.3.1 Determine the minimum configuration and budget needed to meet SLGs by each business workload

It includes finding the node types, number of nodes, storage configuration, and memory size required during different hours of the day, days of the month, and months of the year. The results help recommend the most appropriate cloud data platform based on performance and cost considerations [10].

Figure 10 shows the predicted change of the application's response time components as a result of reducing spilling by 30%. Predicted response time components help to determine the current and potential performance bottlenecks for each application on each of the cloud data platforms. They also serve as a basis for evaluating options and determining the minimum configuration and budget needed to continuously meet SGs for each workload.

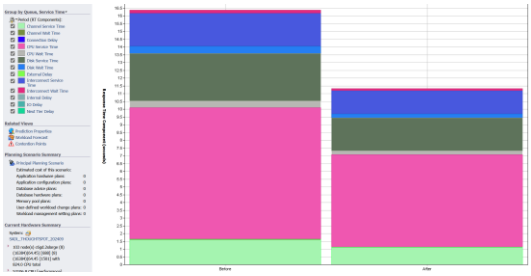

Figure 10. Predicted response time components illustrating the impact of reducing spilling by 30%

Modeling and optimization are used to address various tactical and strategic challenges. For example, during capacity management and budget planning for next year, modeling helps predict the minimum configuration needed to meet SLGs for the expected increase in the number of users by 12% per year and data growth of 10% per year.

As shown in Figures 11 and 12, modeling and optimization predict when and what configuration changes will be required to continuously meet SLG with the lowest cost. As the Snowflake scale-out limit (10 clusters) is reached for 2XL VW, we recommend scaling up to 3XL and then scaling out.

Figure 11. The predicted minimum configuration will be sufficient to provide the response time below SLG

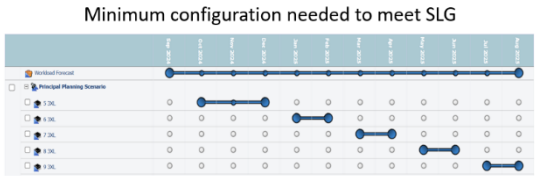

Figure 12. Predicted size of the configuration changes needed to meet SLGs

3.3.2 Cloud Migration Decision Optimization

Models can be used to evaluate and optimize cloud migration decisions [11,12]. For example, the data load on-premises takes nine days. The objective is to determine the minimum Snowflake configuration needed to finish the load within three days.

First, we analyze observability results on premises. Figure 15 shows the differences in CPU utilization during four phases of loading data during 9 days. The challenge is to determine the minimum configuration, including scaling rules needed to parallelize data load and finish it 3 days.

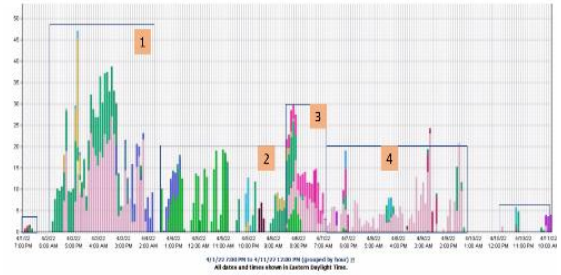

Figure 13. Change in CPU usage during four phases of data loading on-premises before migration to the cloud.

Interval On-Prem	On-prem Time (hours)	Snowflake Time (Hours)	Snowflake Clusters	Credits	Monthly Cost
4/02 8am - 4/04 4am	44	14.5	2*4XL	3,705	
4/04 8am - 4/06 5am	45	14.9	1*4XL	1,901	
4/06 8am - 4/07 8am	24	8.3	1*4XL	1,059	
4/07 8am - 4/09 9pm	61	15.8	1*4XL	2,028	
Total	174	53.5		8,693	$20,863

Table 5. Predicted minimum configuration and budget needed to reduce load time to 3 days after migration to the cloud.

For each phase of the data load, modeling results presented in Table 5 determine the minimum Snowflake configuration needed to reduce load time from nine to three days.

After migrating the data load workload to the cloud, we compared the actual, measured performance and cost with the expected.

Cost verification

The number of credits used during 8 months after migrating the load workload to the cloud is within 10% of expected (Figure 14). Exception is the third month when the load process was performed twice.

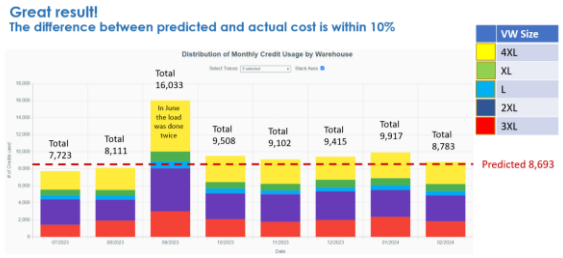

Figure 14. The actual cost is within 10% of expected

Performance verification

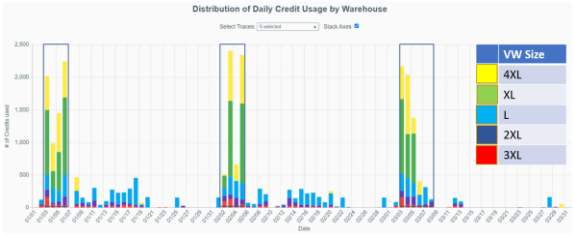

Figure 15. We expected that the data load will take 3 days, but in reality, it takes almost 4 days

The cost was predicted accurately in this example, but additional tuning is needed to meet performance SLGs continuously (Figure 15).

3.3.3 New application sizing

Measurement data collected during the testing of the new applications (Figure 16) are used by modeling and optimization to determine the minimum configuration and budget needed to meet the SLG of new applications before deployment in the cloud [13].

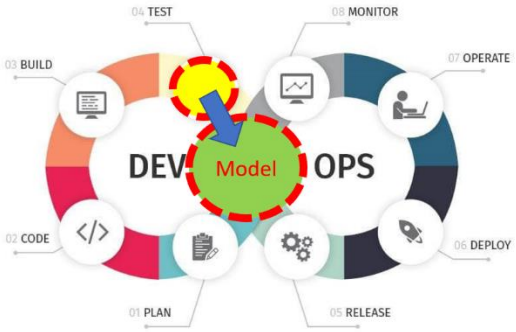

Figure 16. Use measurement data collected during the testing of a new application

The model predicts the impact of expected workloads and data growth, identifying the changes required over the next 12 months (Figure 17).

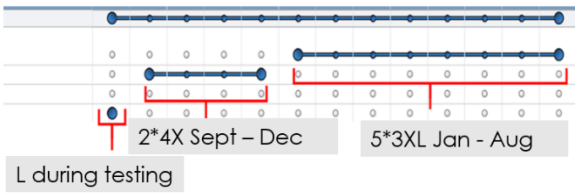

Figure 17. The predicted time of change and size of the Snowflake configuration are needed to meet SLG for the new application after deployment into the cloud.

3.3.4 Building realistic budget

Modeling and optimization determine the minimum configuration and budget needed to meet SLGs for all existing production applications, applications which are planned to migrate to the cloud and new applications planned for deployment. Figure 18 shows an example of the budget based on determining the minimum configuration and workload management and tuning actions needed to meet SLGs for all applications.

Modeling results set realistic cost and performance expectations and enable organizing closed loop control.

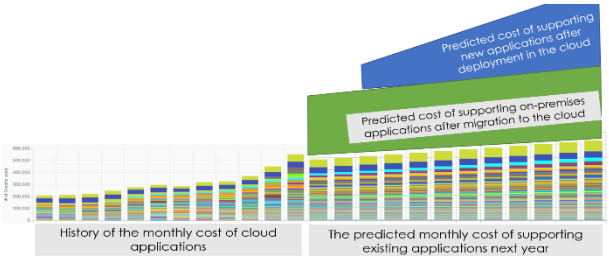

Figure 18 An example of a typical challenge while planning next year budget to meet SLGs for growing business applications and migrating applications from on-premises to the cloud, and for newly deployed cloud-based applications.

4. Performance and Cost Control: Closed-Loop Feedback System

Our methodology establishes a closed-loop performance and cost control system by comparing actual cost and performance metrics with predicted values. This feedback loop continuously monitors outcomes to ensure SLGs are met across applications (Figure 19). By refining configurations based on real-time performance data, this system reduces the risks of unexpected costs and performance issues, maintaining alignment with SLGs.

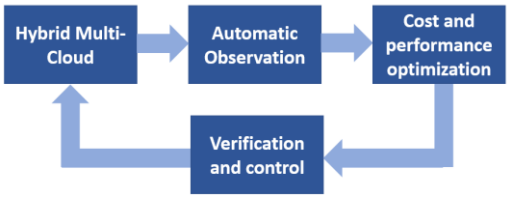

Figure 19: Organization of the closed-loop cost optimization and performance control.

The closed-loop process verifies the effectiveness of recommended changes and supports ongoing optimization. Organizations can improve resource allocation and reduce costs across hybrid multi-cloud environments by applying continuous modeling and optimization.

5. Summary

In this paper we introduced a comprehensive methodology and systems approach for optimizing cost and performance control of applications in the Hybrid Multi-Cloud environment.

Observability and some of the modeling and optimization functions are automated. BEZNext software can be used as SaaS or installed on customer private cloud.

The main challenge of applying our technology is an availability of metrics characterizing applications and queries resource usage. We overcome this challenge by automatic model calibration.

The presented case studies illustrate the value and benefits of our approach to optimizing application costs and controlling performance in complex hybrid multi-cloud environments. Cost savings and performance control are achieved by automating the observability process and focusing tuning efforts on the most resource-intensive and cost-consuming applications, queries, and databases.

This includes targeting queries and applications with the largest volume of data spilled to local and remote storage and those with the greatest time lost due to query failures.

We apply queueing network models and gradient optimization to evaluate options and optimize strategic cloud decisions by determining the minimum configuration, workload management, resource allocation, and budget needed to meet SLGs cost-effectively for all applications on all cloud data platforms.

Finally, we provide cost and performance expectations based on the modeling results. It facilitates closed-loop feedback cost optimization and performance control, reducing the risk of performance surprises in hybrid multi-cloud environments.

The presented case studies cover the benefits of our solutions for a wide range of projects, starting with the sizing of new

applications before deployment to the cloud, appropriate cloud platform selection, optimizing cloud migration decisions, and organizing dynamic capacity management applications in the Hybrid Multi-Cloud environment.

We demonstrated in our case study that the difference between the measured and predicted costs is within 10%.

6. Future Work

The AI Hybrid Multi-Cloud world is expanding rapidly, introducing new technologies, platforms, and countless options. Cloud decisions must consider not only cost and performance, but also storage options, the carbon footprint across different platforms, and other relevant factors. As a result, organizations face a complex multi- objective optimization.

We will continue to prioritize resource investments based on customer requirements and emerging technology trends.

In particular, we intend to enhance observability automation, evaluation of the cloud storage options, improve cost and performance recommendations and support multi- objective optimization.

In addition, we plan to reengineer our software by deploying specialized AI agents for data collection, workload characterization, and continuous comparison of observed and expected results.

We will also strengthen closed-loop feedback control.

Finally, we intend to expand and improve cost optimization and performance control across new cloud data platforms.

ACKNOWLEDGMENTS

The authors acknowledge the contributions of BEZNext colleagues in the development of cost optimization and performance control software.

REFERENCES

[1] FinOps Foundation. What is FinOps? https://www.finops.org

[2] Slingshot: https://www.capitalone.com/tech/cloud/introducing-slingshot/

[3] "Cost Estimation of AI Workloads" by FinOps.org

[4] Top 20 FinOps tools to consider in 2024: https://www.finout.io/blog/finops-tools-guide

[5] "Cloud Cost Optimization: A Comprehensive Review of Strategies and Case Studies" (2023). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4519171

[6] "FinOps for Cloud Cost Management": https://www.ibm.com/think/topics/finops-cloud-cost-management

[7] L. Kleinrock. 1976. Computer Applications (1st ed.). Queueing Systems, Vol 2. Wiley-Interscience, New York.

[8] M. Reiser, S. Lavenberg. 1980. Mean value analysis of closed multichain queueing networks. J.ACM 27, 2 (April 1980), 313-322. https://doi.org/10.1145/322186.322195

[9] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[10] B.Zibitsker and A. Lupersolsky. A new approach to cloud cost optimization and performance control: https://bit.ly/4hwKU0b

[11] B. Zibitsker. "Which Platform is Best for your Cloud Data Warehouse? https://www.beznext.com/wp-content/uploads/2022/02/BEZNext-White-Paper-Which-Platform-is-Best-for-your-Cloud-Data-Warehouse-2-17-2021.pdf

[12] B.Zibitsker and A. Lupersolsky. 2022. The Journey to the Cloud-Hybrid multi-cloud. https://www.beznext.com/wp-content/uploads/2022/02/220225-BEZNext-White-Paper.pdf

[13] B. Zibitsker. 2022. Cloud Performance and Financial Governance Optimization (FinOps). Conference session. In CMG Impact 2022.

[14] B. Zibitsker and Alex Podelko. ICPE 2020. Performance Testing and Modeling for New Analytic Applications. Video. (21 May 2021). Retrieved November 16, 2023, from https://youtu.be/dnTrMiIYR98?si=pzugRScwJEeODMpZ

[15] J.R. Storment & Mike Fuller, Cloud FinOps, O'Reilly Media, 2020

[16] Herbst, N. R., Kounev, S., Reussner, R. (2013). "Elasticity in Cloud Computing: What It Is, and What It Is Not.": 10th International Conference on Autonomic Computing (ICAC '13), pp. 23-27, https://www.usenix.org/system/files/conference/icac13/icac13_herbst.pdf

[17] Wu, S., Li, J., & Kumar, S. (2021). "Machine Learning-based Multi-cloud Resource Allocation for Real-time Big Data Analytics." IEEE Transactions on Parallel and Distributed Systems

[18] Singh, R. K., Gupta, P., & Alshamrani, S. (2023). "Workload-aware Hybrid Cloud Orchestration: A Survey" Journal of Cloud Computing.

[19] Fu, A., Li, R., & Chen, H. (2021). "Adaptive DB Tuning with Deep Reinforcement Learning in Cloud-based Data Lakes" Proceedings of the VLDB Endowment

[20] Sun, X., Jermaine, C., & Mozafari, B. (2022). "Autonomous Query Optimization in the Cloud via Learned Cost Models." ACM SIGMOD Conference

[21] Zhang, L., Xu, M., & Buyya, R. (2021). "Evolutionary Multi-Objective Optimization for Multi-Cloud Resource Scheduling." Journal of Parallel and Distributed Computing

[22] Predicting Cloud Data Platforms Carbon Footprint for Large Data Warehouses: bit.ly/49cqq8I; bit.ly/3ScrAu4

[23] Li, X., Xu, W., Wang, G. (2022) "Adaptive Data Placement in Hybrid Clouds: A Performance-Cost-Aware Approach" Future Generation Computer Systems.

[24] Cloud benchmarks aren't enough. The use of modeling to address the limitations of benchmark tests: bit.ly/49cqq8I