Agentic AI observability automation, performance and cost control

Boris Zibitsker, Alex Lupersolsky

BEZNext

Contents

1 Executive Summary

Observability automation is the foundation of cost optimization and performance control.

The rapid adoption of agentic AI is transforming enterprise computing from a world of relatively stable, deterministic applications to a dynamic, fast-growing landscape of multi-agent systems that reason, retrieve context, invoke tools, query enterprise platforms, and act across multiple layers of infrastructure.

That shift creates major opportunities for automation and productivity, but it also poses a challenge in controlling the cost and performance of these systems. [1]

Agentic AI workloads are growing rapidly, and infrastructure constraints such as electricity, cooling, water availability, and data center capacity are emerging concerns that may influence future scaling strategies in some regions and environments.

This trend underscores the importance of hybrid and multi-cloud strategies, as future agentic AI demand may exceed the practical capacity, cost, or availability constraints of a single provider in some situations.

Vendors are developing CPUs, GPUs, and specialized AI accelerators, including ASICs such as TPUs and Trainium, to improve performance per watt and reduce infrastructure cost.

Cost and performance are no longer easy to predict. The execution path of one of the agent requests may be very different from the next. The number of agent interactions can grow quickly, and infrastructure usage can shift sharply as the number of agents, users, and data sources increases.

As a result, observability plays an increasingly important role in optimizing and controlling cost and performance in multi-agent systems. [2]

Observability relies on telemetry collection, including metrics, logs, and distributed traces, to capture agent response time components, resource consumption, cost, and cross-component interactions.

In this white paper, we treat observability for agentic AI as the foundation for cost optimization and performance control. We review how observability enables workload characterization, modeling, optimization, and closed-loop cost-performance control of agentic workloads.

> **Takeaway:** Observability must evolve from reporting to enabling prediction, optimization, and continuous control.

2 Introduction – Challenges of Organizing Observability for Agentic AI

Agentic AI systems introduce a new level of complexity in managing performance and cost. As organizations adopt multi-agent architectures, traditional assumptions about predictable workloads and stable execution paths no longer hold.

To manage these systems effectively, organizations need a deeper understanding of how agentic workloads behave and how they consume resources. This requires a shift from traditional monitoring to a more comprehensive observability-driven approach.

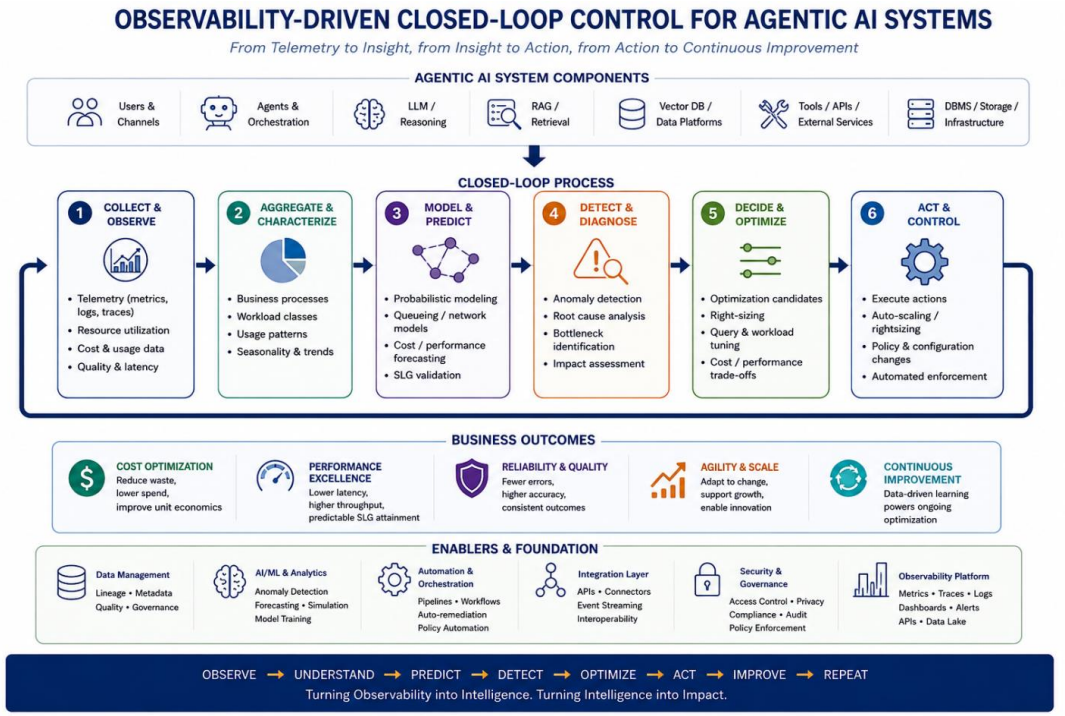
2.1 Observability Role

Adoption of agentic AI is accelerating at an unprecedented rate. Organizations are embedding autonomous intelligent agents across customer service, internal operations, analytics, software engineering, knowledge management, and decision support. These agents do not simply retrieve and present data. They interpret intent, build plans, call language models, retrieve context, query data platforms, invoke external tools, and often loop through these stages multiple times before producing a final response or acting.

That behavior makes agentic AI fundamentally different from traditional enterprise applications. In a conventional application, the execution path is typically known in advance. In an agentic system, the execution path depends on intermediate reasoning results and can vary significantly from one request to the next. As a result, organizations that want to manage cost and performance effectively must first build a much richer observability capability. [11, 12]

Telemetry collection mechanisms (metrics, logs, and traces) provide the data required for observability and performance analysis. The role of observability in this context is broader than in traditional monitoring approaches (limited to metrics and alerts). It includes collecting data on agents' response times, resource consumption, data usage, concurrency, and cost across all relevant layers of the environment. It also includes aggregating that data into business workloads, identifying the most important anomalies, estimating missing parameters, and providing the inputs required for modeling and optimization. [7, 8] In this sense, observability automation is a foundation for cost optimization and performance control for agentic systems, not just a source of dashboards (Figure 2.1).

Figure 2.1. Core functions of observability in agentic AI systems.



Each business process managed by agents has performance requirements or Service Level Goals. The goal of cost-performance optimization and control is to dynamically adjust configurations, workload management rules, data design, and queries to meet those goals at the lowest possible cost.

In this white paper, we discuss observability automation, workload characterization, modeling and optimization role, and the organization of a closed-loop control system.

> **Takeaway:** Agentic AI systems require observability as the foundation for managing dynamic, non-deterministic workloads and meeting SLGs at the lowest cost.

2.2 Architectural Layers of Agentic AI

Agentic AI systems operate across multiple layers and involve dynamic interactions between models, data platforms, and external tools. Understanding this structure is essential for analyzing performance, identifying bottlenecks, and managing cost.

This section introduces the architectural layers, and execution model that defines how agentic AI systems behave, providing the foundation for observability, modeling, and optimization.

Agentic AI components are organized into three logical architectural layers. The cognitive layer comprises large and small language models, along with agent orchestration functions that coordinate reasoning, planning, and iteration.

The knowledge layer comprises retrieval-augmented generation, vector stores, and enterprise data platforms or DBMSs that provide the structured and unstructured context agents need. The action layer comprises orchestration interfaces, Model Context Protocol interfaces, external systems, APIs, SaaS applications, and enterprise tools that enable agents to perform tasks and interact with the outside world.

This layered view is useful because it allows organizations to separate where agents spend time, where costs are incurred, where resources are consumed, and where bottlenecks arise. It also provides a practical foundation for workload aggregation and for constructing multi-layer models of end-to-end performance. Workload aggregation and workload characterization transform request-level telemetry into stable analytical workload profiles.

2.3 Agentic AI Workflow as a Probabilistic Process

In an agentic AI environment, an agent's request is not handled by a single component in a fixed order. Instead, it may begin with orchestration, call a language model to generate a plan, access retrieval components to gather context from a vector store or an enterprise data platform, return to the model for synthesis, invoke external tools via an MCP interface, and then repeat parts of the cycle before producing a final output. Because the path depends on intermediate reasoning results, the workflow is non-deterministic.

Observability and modeling define a probabilistic routing process across components. Each move from one component to another is characterized by a conditional probability. The resulting transition matrix or execution graph shown in Figure 2.2 captures the stochastic nature of agentic workloads and links observability to queueing network modeling, bottleneck analysis, and cost prediction. This is a major difference between agentic AI systems and traditional deterministic workflows. The sum of outgoing probabilities should equal 1.

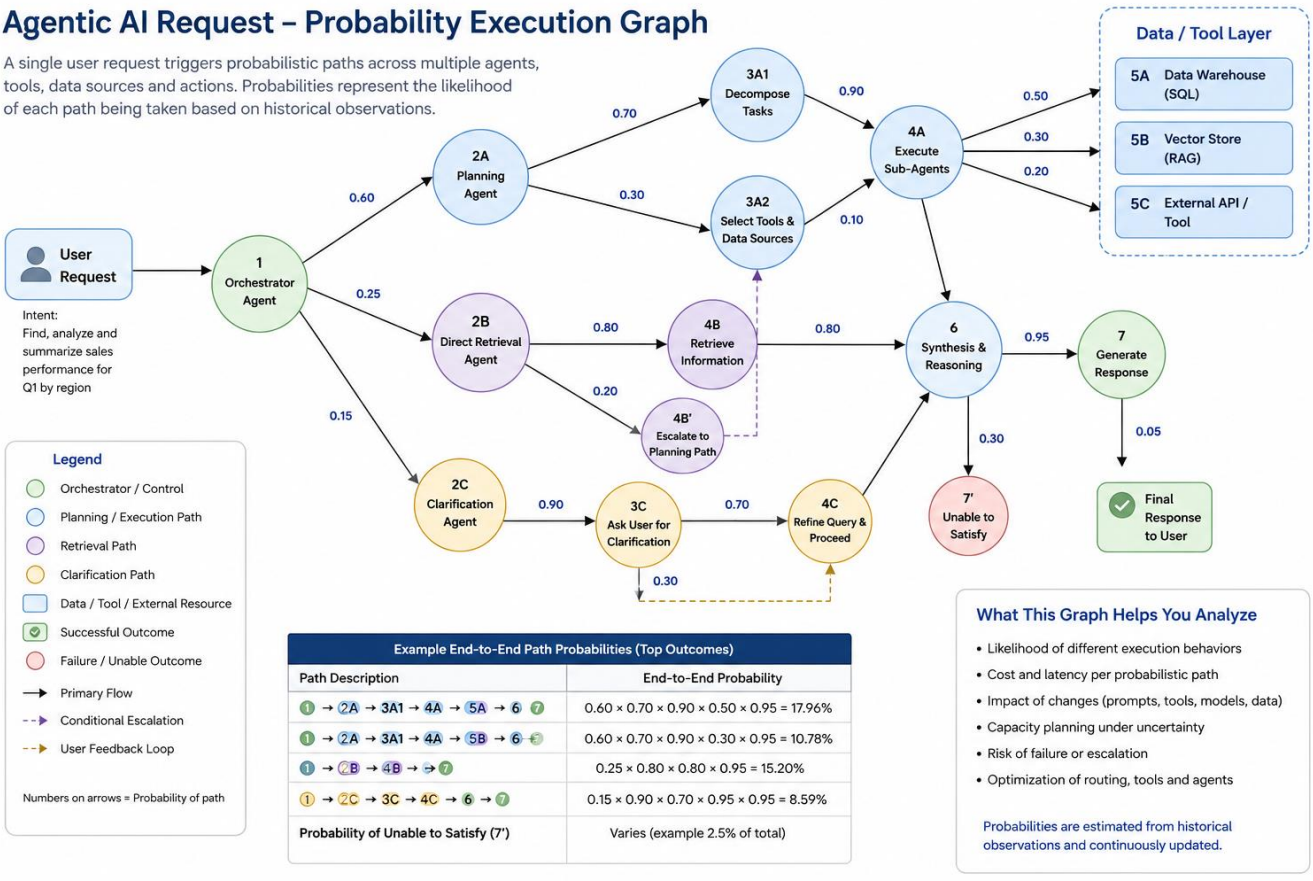


Figure 2.2. Probabilistic execution graph of an agentic AI system.

The objective of observability is to link agent activities related to LLMs, databases, vector stores, tools, and the corresponding costs. We address these challenges as the foundation for organizing closed-loop performance and cost control.

3 Observability Measurement Requirements

To enable cost-performance control in agentic AI systems, observability must provide a comprehensive and structured view of system behavior.

This requires collecting data at multiple levels—including configuration, system operation, and individual requests—and correlating this data across all layers of the environment.

This section defines the observability requirements needed to support workload characterization, modeling, and optimization.

3.1 Observability goals

Because the objective is not just retrospective reporting but predictive control, observability for agentic AI must collect a much broader body of information than conventional application monitoring typically provides. The required data spans server hardware, software configuration, hourly resource metrics, session activity, request-level measurements, and cross-component call relationships.

Observability must also include quality metrics such as response accuracy, consistency, and reliability.

Observability results focus performance management on the most critical cost and performance problems. In practice, that means identifying the agents, applications, and requests with the highest anomaly frequency, the highest cost, the most severe failures, or the largest contribution to poor end-user response time. Experience shows that major improvements often come from a relatively small set of high-impact queries or interactions. Observability narrows the field; modeling and optimization determine which change is most likely to produce the needed result.[4]

Each cloud data platform has different tools for analyzing performance measurement data and generating performance management recommendations.

Across platforms, modeling and optimization are used to evaluate options and improve scaling, workload management, performance tuning, and platform configuration decisions. The objective is not simply to detect drift after the fact. It is to continuously meet performance objectives at the lowest cost.

The value of observability is realized during workload characterization. Workload characterization begins with workload aggregation, which transforms agents' measurement data into business workloads' parameters. Typically, each business process is managed by a group of agents. For each business process, the workload characterization builds hourly profiles of performance, resource consumption, data usage, and cost. These workload profiles provide a more stable analytical object than raw traces and serve as practical input for modeling and optimization.

Once workload characterization is done, organizations can use queueing network models and other analytical methods to estimate expected response time, throughput, resource utilization, cost distribution, and capacity requirements across optional scenarios. Modeling also enables sizing new applications before deployment, evaluating migration options, comparing platforms, determining realistic budgets, and assessing the likely impact of workload growth. Optimization uses modeling on each step to identify the smallest, least expensive configuration that still meets the business process's Service Level Goals.

3.2 Performance, Resource Utilization, and Cost Metrics

At the configuration level, the required information includes node or instance types, node counts, CPU/GPU counts, processor speeds, memory size, network connectivity, local and remote storage characteristics, storage controller

connections, and software configuration parameters such as concurrency limits, workload priorities, performance goals, and intra-request parallelism limits. [5]

At the operational level, observability must include hourly measurements such as CPU and memory utilization, active disk counts, I/O operations, megabytes read and written, network traffic and messages counts, session start and end timestamps, and user or application identities. These measurements support workload characterization, performance management, anomaly detection, and calibration of analytical models.

At the request level, the information becomes even more crucial. For example, this includes cost in platform-specific units such as credits or DBUs, query identifiers, error codes, start and end timestamps, elapsed time, preparation time, execution time, total CPU time, total delay time, volume of data spilled to remote storage, I/O counts, data volumes processed, and the set of downstream components called by each request. This level of detail is exactly what is needed to identify expensive or slow requests, isolate the causes of failed or delayed requests, and connect end-user response times to the internal behavior of individual layers and components.

The message is clear: observability for agentic AI is not just about tracking prompts and token counts. It must connect business workloads to resource usage, data consumption, concurrency, and cost across the entire environment. Without that connection, organizations can explain symptoms but cannot confidently manage performance and cost, select the right platform, size a new deployment, or implement closed-loop control.

A key requirement for agentic AI observability is cross-layer correlation, linking agent requests to downstream model calls, data platform queries, and external tool interactions.

3.3 Why Missing Data Matters

One key point is that not all necessary information is available from every cloud data platform, which is a significant issue. Missing values, such as degree of parallelism, total CPU time per request, detailed I/O behavior, or exact cost attribution at the agent and business-process levels, directly limit the accuracy of workload characterization and modeling. When such values are missing, they must be estimated or inferred during model calibration. That is why observability alone is not enough. The management needs to combine observability with modeling, calibration, and optimization.

Another practical issue is overhead. Collecting detailed telemetry from thousands of agents and users can be costly and intrusive. Therefore, telemetry collection should be automated, selective, and adaptive. Similar agents can be grouped into representative classes; detailed tracing can be temporarily enabled during periods of instability or when anomalies occur; and the level of data collection can be adjusted based on the business value of the monitored workload.

4 Observability Across Cloud Data Platforms

Agentic AI systems rely on multiple data platforms, each with its own observability capabilities and limitations. These differences directly affect the ability to analyze workload behavior, attribute cost, and optimize performance.

This section examines how observability is implemented across major platforms and explains how these differences impact modeling, optimization, and control.

4.1 Platform-independent challenges

A single agentic request can increase latency and resource consumption across multiple layers. For example, an LLM call might take 200 milliseconds, RAG retrieval 400 milliseconds [13], LLM inference 2,500 milliseconds, tool or API calls 1,000 milliseconds, orchestration 200 milliseconds, memory access 100 milliseconds, verification 300 milliseconds, and telemetry logging 50 milliseconds. CPU, I/O, and memory usage vary by stage, with inference primarily consuming CPU and retrieval and tool interactions causing significant I/O activity.

This type of decomposition is crucial for performance management because it identifies where to focus optimization efforts. If most response time occurs in the inference layer, optimizing the vector store will have a limited impact. If I/O-heavy retrieval is the main issue, adjusting prompts will not solve it. If tool calls are slow, the bottleneck may be outside the language model entirely. The real value of observability lies in revealing the primary response-time components and their resource impacts, so optimization can target the most critical parts of the workflow.

Observability in agentic AI is challenging because these systems are complex and the decisions informed by observability are broad and strategically important. Organizations use observability results to select appropriate platforms, determine the minimum configurations needed to meet business-process performance goals, size new agents and agentic applications before deployment, manage dynamic capacity, analyze benchmark results to compare agentic platforms and configurations, evaluate infrastructure reliability, and even monitor energy consumption.

Several challenges are prominent. The **first** is architectural complexity, as agentic AI systems include orchestration, language models, retrieval layers, vector stores, MCP interfaces [14], enterprise databases, and external systems. The **second** challenge is workload variability, with the number of agents, users, and interactions often changing rapidly, and the depth of the execution graph varies significantly by request. The **third** challenge is the mismatch between required and available data, since FinOps, DevOps, and PlatformOps observability tools were not designed for highly variable, multi-layer agentic workloads, and many of their metrics fall short of what's needed for analytical modeling. The **fourth** challenge is collection overhead, because the more detailed the telemetry, the more carefully it must be managed.

These challenges explain why traditional observability tools can tell organizations what happened but still leave them unable to answer the most important questions: what minimum configuration is needed to meet a Service Level Goal, what budget is realistic for a growing workload, which change to prioritize first, and what will happen if the workload increases, shifts, or becomes more complex.

Observability features vary across cloud data platforms. Architectures, billing methods, data collected, and the maturity of AI-specific telemetry all differ. Monitoring goals remain the same: evaluate performance, allocate costs, trace agent activity, characterize workloads, and enable closed-loop cost and performance management.

The following sections review the specifics of observability for several cloud data platforms and then examine examples of analysis, modeling, and control for Snowflake and Teradata.

> **Takeaway:** Platform-native telemetry is necessary but not sufficient. Effective control requires cross-layer correlation and a vendor-independent analytical framework.

4.2 Snowflake

Snowflake provides strong workload isolation through independently scalable virtual warehouses and offers useful telemetry, including query history, warehouse and query cost metering history, and **Cortex**-related AI observability. It is well-suited to environments where predictable performance isolation and cost governance are important. However, Snowflake abstracts low-level CPU and execution details, requiring indirect inference for modeling purposes. In practice, Snowflake data helps build workload performance and cost profiles and conduct anomaly analysis, but model calibration is often required to estimate missing parameters and align predicted results with measured response times and throughput. [15]

4.3 Databricks

Databricks provides a rich set of system tables, MLflow tracing, model-serving telemetry, vector search information, deep request logging, vector-search status, and billing information. Databricks exposes metrics such as latency, request rates, error rates, provisioned concurrency, CPU utilization, memory utilization, GPU utilization, and GPU memory usage through model-serving endpoints and related APIs. That breadth is valuable for engineering-driven environments, but it also requires discipline for cost attribution and cross-layer workload mapping. Databricks provides more detailed AI-serving telemetry than many platforms, yet organizations still need a unifying framework to connect those measurements to end-to-end business workloads and Service Level Goals. [16]

4.4 Teradata VantageCloud

Teradata **VantageCloud** excels at mature workload management. MCP enables structured interaction between agents and tools. Measurement data include query response times, throughput, concurrency, queue wait times, CPU usage per node and per query, AMP worker task utilization, I/O throughput, memory and spool consumption, skew metrics, elasticity behavior, and cost and unit economic indicators. This makes the platform especially suitable for regulated, mission-critical, structured enterprise workloads where concurrency, predictability, and traceability are key. [17]

In Teradata VantageCloud pricing, consumption units are influenced by a combination of:

- **Compute usage** (CPU + memory + parallelism)
- **I/O activity** (data scanned, moved, joined)
- **Storage access and persistence**
- **System resources consumed by queries and workloads**
- **Concurrency/workload management overhead**

4.5 Google BigQuery, Microsoft Synapse, and Oracle

Google **BigQuery** provides large-scale elasticity and integrated logging and monitoring, and it is well-suited for bursty, retrieval-heavy workloads, though its serverless design limits direct control over execution details. Azure **Synapse** offers both dedicated and serverless options and integrates seamlessly with Azure's monitoring ecosystem, but performance can vary significantly depending on distribution design and governance enforcement. **Oracle** AI Database and HeatWave embed AI features directly into the data platform and support highly governed environments, but MCP integration is more limited and often requires custom setup. For those reading this white paper, the key takeaway isn't that one platform is always better, but that factors such as observability maturity, missing data, execution control, and billing transparency must be considered together when choosing a foundation for agentic AI workloads.

> **Takeaway:** the key takeaway isn't that one platform is always better, but that factors such as observability maturity, missing data, execution control, and billing transparency must be considered together when choosing a foundation for agentic AI workloads.

Let’s review two examples of how Snowflake and Teradata observability data are used for workload characterization, anomaly detection, identifying critical applications, assessing SQL impact on performance and cost, and generating performance optimization recommendations. We will also illustrate how observability data are used to determine the minimum configuration needed to meet the business's performance requirements, size new agents and applications before deployment, and support dynamic capacity management, build a realistic budget, and maintain continuous performance and cost control.

5 Example of organizing observability for agentic AI systems

Understanding observability concepts is important, but their value becomes clear only when applied to real workloads.

This section presents practical examples based on Snowflake and Teradata environments. These examples demonstrate how observability data are used to identify cost drivers, analyze performance behavior, and guide optimization decisions.

Let’s review two examples of how Snowflake and Teradata observability data are used for workload characterization, anomaly detection, identifying critical applications, assessing SQL impact on performance and cost, and generating performance optimization recommendations. We also illustrate how observability data are used to determine the minimum configuration needed to meet the business's performance requirements, size new agents and applications before deployment, support dynamic capacity management, build a realistic budget, and maintain continuous performance and cost control.

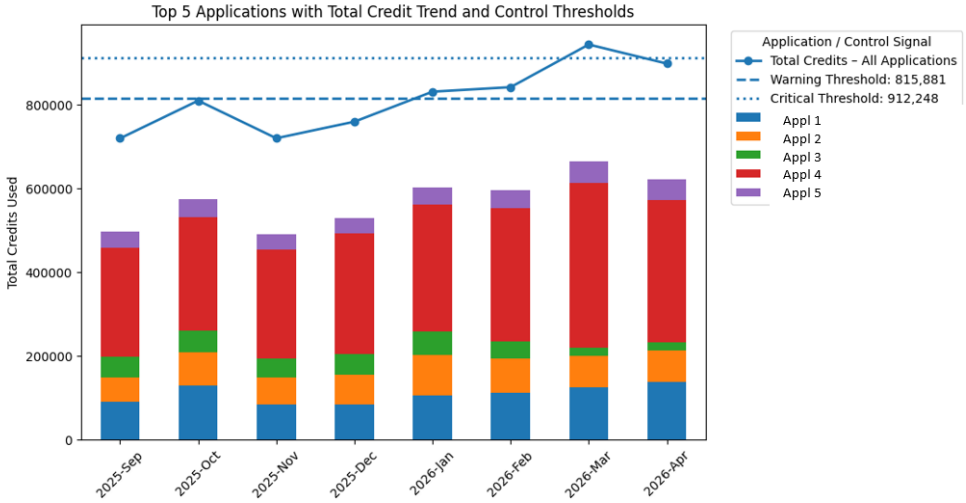
5.1 Snowflake example

The workload characterization automatically builds the hourly performance, resource utilization, and cost profiles for each application, which are used for trend analysis and development performance optimization recommendations.

Figure 5.1 below shows the dynamics of credit usage by Snowflake applications over the past five years. Colors represent different applications. The figure illustrates the uncontrolled growth of workloads over time. After an initial adoption phase, the system enters an acceleration phase, followed by a rapid expansion of application activity. At scale, the workload stabilizes at a high-cost plateau with significant volatility, indicating the absence of closed-loop cost and performance control.

During the first three years, credit use increased rapidly, and in 2024, performance optimization efforts halted that growth. An increase in the number of deployed applications and data volume at the end of 2024 led to a significant rise in credit use, but virtual warehouse consolidation, rightsizing, and performance optimization stabilized credit use growth in 2025.

Figure 5.1. Credit use growth by Snowflake applications over the last five years.



The deployment of agentic AI applications caused a continuous increase in credit use in 2026 as shown on Figure 5.2.

Figure 5.2. Credit use growth has been affected by the deployment of agentic AI applications during the last 8 months.

> **Takeaway:** Credit growth should be interpreted by application and workload profile, not only by total monthly consumption.

The figure below shows the credit use growth during the last 12 months

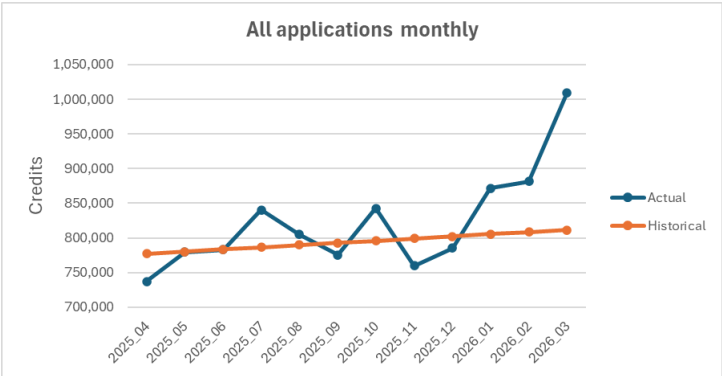


Figure 5.3. Total credit use by applications is growing faster than the historical trend.

In this case, a significant increase in the volume of data processed by Snowflake applications is a leading cause of the credit use increase.

Table 5.1 shows the top 5 applications that use over 80% of the credits. A significant percentage of credits is lost due to idle virtual warehouses.

APPLICATIONS	CREDITS PAID	CREDITS PAID COMPUTE	CREDITS USED BY QUERIES	CREDITS LOST
APPL4	3,579,367	3,303,227	2,978,102	325,125
APPL1	1,206,884	1,197,774	814,352	383,422
APPL2	875,988	855,143	632,127	223,016
APPL3	551,403	547,377	473,241	74,136
APPL5	497,657	496,322	363,861	132,461

Table 5.1. The top 5 applications use over 80% of the credits. Credit paid for Compute, credits used by queries and credit lost characterize provide different views on credit used

Each application shows a different trend in credit usage growth.

The objective of performance management, capacity planning, and cost optimization is to meet customer satisfaction requirements at the lowest cost.

Focusing on queries with the highest credit use, the highest frequency of cost and performance anomalies, the highest frequency of failures, and the highest volume of data spilled to local and remote storage narrows down the scope of performance optimization efforts.

> **Takeaway:** Observability narrows the problem space by identifying the queries, applications, and anomalies with the greatest cost and performance impact. Cost is usually concentrated in a small number of applications. Optimization should first focus on the workloads that account for the largest share of spending.

5.2 Teradata example

Teradata observability provides hourly profiles containing CPU utilization, parallelism sessions, response time, throughput, and cost for each business workload. For example, Figure 5.4 below shows the hour-by-hour CPU utilization by each workload.

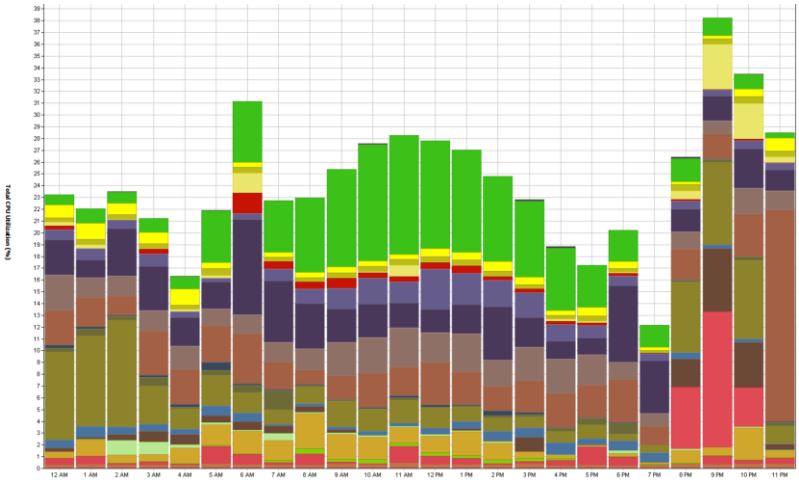


Figure 5.4. CPU utilization by business workloads for 24 hours of the average day

Teradata offers unique observability capabilities that differentiate it from Snowflake, Databricks, and other cloud data platforms.

Teradata's differentiation in agentic AI observability for resource consumption and cost control lies in the tight integration between workload management, telemetry, and cost semantics at the query, workload, and system levels.

Teradata natively correlates query, user, application, and workload information with resource usage (CPU, I/O, memory, spool), performance, queueing, skewness, parallelism, and cost, including CPU time, node usage, and consumption units. Teradata answers important questions, like “Which agent/workflow step caused the cost increase, and why?”

Teradata captures CPU time, I/O count, and I/O time per step. This provides deeper physical resource visibility than platforms that primarily expose query-level or Spark-stage-level telemetry.

Teradata can help isolate the impact of retrieval steps, feature joins, aggregation, and output generation when those activities are represented in database workloads.

Teradata’s Workload Management (TASM / TDWM) classifies queries into workloads, assigns priorities and resource limits, and tracks consumption per workload. It maps Agent → Workflow → Workload → Resource budget. This enables observation of cost by workload, SLA/SLG violations by workload, and resource contention between workloads. This capability is critical for multi-agent systems at scale.

Teradata exposes queue wait time, active vs delayed queries, concurrency thresholds, and resource contention signals It can identify cost spikes caused by concurrency rather than query complexity alone.

For Agentic AI, Teradata observes when queue delays cause retries, increasing response time and cost, and determines when to reduce skews by redistributing data. It also determines when skew affects joins, causing longer runtime, higher CPU utilization, and, as a result, higher cost. It can be especially important for RAG-related joins and other skew-prone workloads.

Teradata tracks the number of I/Os, I/O response time, data read/write volumes, and cache versus-disk data location. This helps monitor the current I/O profile and predict future behavior when data placement or object-storage access patterns change.

Teradata still lacks:

- Native agent-level tracing (like LangSmith)
- LLM/token-level cost tracking
- Cross-platform observability (multi-cloud agents)
- Native probabilistic workflow modeling

Teradata provides deep physical-resource observability, while platforms such as Databricks provide richer AI/ML-level telemetry. Each cloud data platform collects measurement data, but increasing the number of monitored agents can significantly increase monitoring overhead.

Takeaway: Teradata illustrates the value of deep resource-level observability, while also showing why agentic AI requires cross-layer tracing beyond any single platform.

6 Setting Service Level Goals for Agentic Workloads

Observability provides a foundation for organizing cost optimization and performance control. Optimization decisions use Service Level Goals as criteria for evaluating alternatives.

Meeting SLGs is necessary but not always sufficient for user satisfaction, particularly in AI systems where accuracy and consistency also play a major role.

For some business processes and applications, organizations have formal SLAs and SLGs that define acceptable response time, throughput, accuracy, consistency, and cost. In many cases, however, these goals are not formally defined.

As an example, statistical analysis of performance measurement data can be presented to business process owners, developers, and financial and IT leaders for approval of performance SLGs for different applications or lines of business.

An example of the proposed SLGs for five applications is shown in Table 6.1 below:

	Appl 1	Appl 2	Appl 3	Appl 4	Appl 5
Execution count	1,676,192	32,329,825	9,875,600	120,365,902	776,445
Avg response time (sec)	10.351	0.913	3.326	3.203	9.447
Avg MB processed	11,715	2,475	3,097	688	12,992
Avg query credits	0.036	0.001	0.005	0.002	0.040

Table 6.1.An example of the Service Level Goals determined for 5 applications

The typical month was selected in coordination with the customer based on application credit-use trends.

Takeaway: SLGs should be grounded in measured workload behavior and approved by business, IT, engineering, and finance stakeholders.

7 Measuring user satisfaction

Some business processes have formal performance requirements, but many do not. For this analysis, we assume that when application performance meets the SLG, and accuracy and consistency are sufficient, customers are satisfied. We focus first on customer satisfaction with performance.

The level of customer dissatisfaction can be approximated by the frequency of SLG violations.

For this analysis, we assume that when application performance meets the SLG, customers are satisfied with performance.

We measure the level of performance-related customer dissatisfaction by the frequency of SLG violations.

			Elapsed time (milliseconds)					Elapsed time (milliseconds)			
		Application 1					Application 5				
Year	Month	Execution count	Min Elapsed time	Avg Elapsed time (ms)	Elapsed time 90 percentile	Max Elapsed time	Execution count	Min Elapsed time	Avg Elapsed time (ms)	Elapsed time 90 percentile	Max Elapsed time
2025	4	1,186,563	43	20,212	15,225	14,401,190	669,418	50	10,275	21,791	1,204,958
2025	5	2,056,369	43	8,725	6,907	14,407,044	784,666	50	11,445	22,011	1,200,572
2025	6	1,984,658	41	9,082	8,426	14,401,309	776,716	51	11,854	21,426	1,200,498
2025	7	1,701,955	43	9,330	7,719	14,400,628	767,883	53	10,635	20,063	1,206,656
2025	8	1,541,517	46	8,933	7,842	12,397,328	753,673	54	9,315	17,967	1,200,994
2025	9	1,676,192	44	10,351	8,990	14,404,088	776,445	46	9,447	18,008	1,200,767
2025	10	1,894,492	46	15,118	8,202	14,400,349	855,177	53	10,148	18,928	1,200,698
2025	11	1,135,411	46	16,489	8,938	14,401,417	720,555	54	9,157	16,821	1,200,682
2025	12	1,228,216	40	13,747	8,530	14,401,275	619,530	62	11,028	21,374	1,049,978
2026	1	1,576,193	41	14,365	7,710	14,407,224	766,449	70	10,923	22,493	808,571
2026	2	1,852,780	49	16,447	9,399	28,804,846	656,829	58	12,497	25,153	365,921
2026	3	2,225,777	51	12,679	8,149	14,400,836	654,698	59	14,168	30,936	661,149

Table 7.1 shows an example of the SLG (yellow lane) for the two most credit-using applications, based on measurement results collected during the typical month. Highlighted in red are months with SLG violations

Application 1 Leading Indicators Dashboard Graph for 2026 is presented on Figure 7.1.

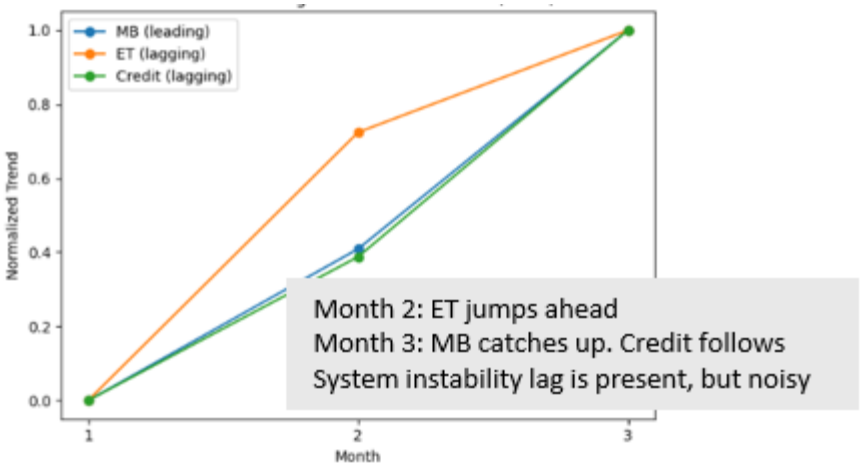


Figure 7.1. Leading indicator dashboard (MB → ET → Cost lag)

Figure 7.2 below shows the normalized trends of SLG violations for Application 1 monthly in 2026 (shapes matter, not absolute values)

- MB (leading) → data inefficiency (root cause)
- ET (lagging) → performance impact
- Credit (lagging) → cost impact

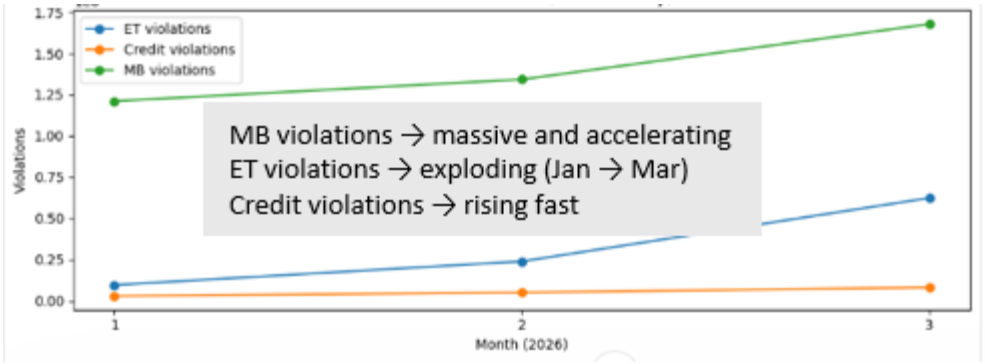


Figure 7.2. Elapsed Time, Credit Use, and MB processed SLG Violations

For data-intensive agentic workloads, MB processed is often a leading indicator, but for LLM-heavy workloads, token usage and model latency may dominate.

One opportunity for cost reduction is reducing the number of credits lost to idle virtual warehouses. According to observability results, application waste is between 15 – 30% credits due to idle virtual warehouses.

Takeaway: Leading indicators depend on workload type. Data volume may dominate data-intensive workloads, while token usage and model latency may dominate LLM-heavy workloads.

8 Performance optimization

8.1 Snowflake recommendations overview

Thousands of agents may generate millions of requests, but the number of recurring performance-optimization recommendation types is limited.

Snowflake analyzes millions of requests and generates performance-optimization recommendations. Table 4 shows 14 recommendation types generated during the analysis of 6.8 million Snowflake queries.

Row Labels	Sum of Execution count	Sum of GB processed	Sum of GB spilled remote	Sum of Query credits
QUERY_INSIGHT_EXPLODING_JOIN	115,766	20,129,135	156,885	70,802
QUERY_INSIGHT_FILTER_WITH_CLUSTERING_KEY	10,100	413,380	0	760
QUERY_INSIGHT_INAPPLICABLE_FILTER_ON_TABLE_SCAN	62,865	5,122,725	26,527	24,104
QUERY_INSIGHT_INEFFICIENT_AGGREGATE	1,072,120	91,523,987	1,226,717	161,981
QUERY_INSIGHT_INEFFICIENT_JOIN_CONDITION	144,713	23,150,311	17,139	64,764
QUERY_INSIGHT_JOIN_WITH_NO_JOIN_CONDITION	8,931	7,581,643	8,499	5,139
QUERY_INSIGHT_LIKE_WITH_LEADING_WILDCARD	4,944,657	871,487,244	10,099	94,035
QUERY_INSIGHT_NO_FILTER_ON_TOP_OF_TABLE_SCAN	63,240	7,389,381	570,123	23,106
QUERY_INSIGHT_QUEUED_OVERLOAD	3,586	77,952	6	381
QUERY_INSIGHT_REMOTE_SPILLAGE	1,431	2,165,066	1,692,364	12,236
QUERY_INSIGHT_SEARCH_OPTIMIZATION_USED	79,655	441,997	0	1,483
QUERY_INSIGHT_SNOWFLAKE_OPTIMA	44,045	733,750	0	893
QUERY_INSIGHT_UNNECESSARY_UNION_DISTINCT	32,185	4,268,676	8,042	10,459
QUERY_INSIGHT_UNSELECTIVE_FILTER	216,858	34,621,781	119,387	111,733
(blank)				
Grand Total	**6,800,152**	**1,069,107,029**	**3,835,789**	**581,877**

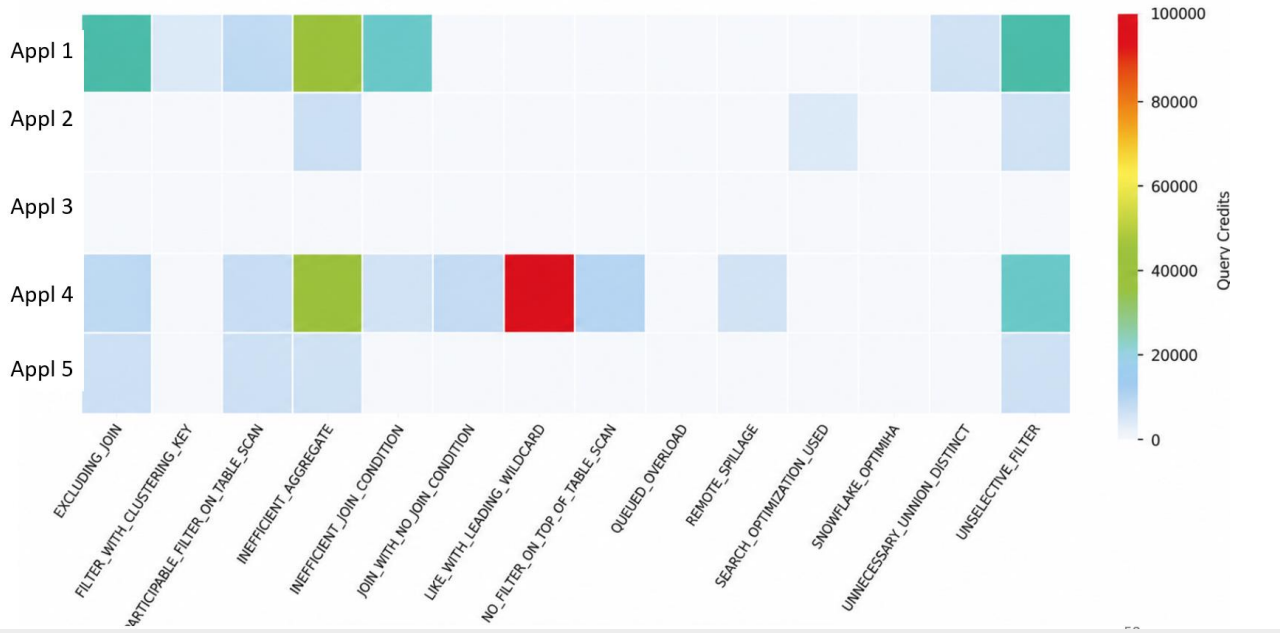
Table 8.1. Snowflake generated 14 types of recommendations for 6.8M queries

In this example, the biggest cost drivers are:

- Inefficient Aggregates
- Wildcard Searches Massive Volume - Strong candidate for indexing/search redesign
- Exploding Joins + Bad Join Conditions - Major SQL rewrite opportunity
- Remote Spillage - Extremely expensive per query

The heat map in Figure 8.1 shows that optimizing queries with “LIKE_WITH_LEADING_WILDCARD” can yield the greatest cost savings.

Figure 8.1. Heat map tuning recommendations vs. potential credit saving



Takeaway: Millions of queries, but a limited set of tuning recommendations

8.2 Role of modeling and optimization in controlling agentic AI cost and performance

Optimization actions may include virtual warehouse consolidation, improved scale-out and scale-up rules, rightsizing virtual warehouse sizes, and determining the minimum configuration and cost needed to meet business performance goals.

Each platform selection, performance optimization, configuration optimization, workload management, design, deployment, sizing, dynamic capacity management, and budgeting decision has many alternatives. The critical questions are:

- What is the minimum configuration required for the business workloads?
- Which performance-management and capacity-planning decisions are best?
- How can the cost and performance of alternatives be compared, and how can a continuous cost-performance control process be organized?
- How to optimize auto-scaling rules, changing prompts, routing requests to a lower-cost model, limiting concurrency, and caching results.

Different techniques can be applied at different time scales: real-time control for per-request routing, near-real-time control over minutes, and batch optimization performed daily or periodically. [3, 6, 9, 19]

The role of modeling and performance optimization is to answer questions like these. [20]

> **Takeaway:** Observability identifies what is happening; modeling and optimization determine which corrective action is most cost-effective.

9 Closed-Loop Cost and Performance Control

Observability provides visibility into system behavior, but visibility alone is insufficient to manage cost and performance in dynamic agentic AI environments.

To continuously meet performance goals at the lowest cost, organizations must move beyond analysis to active control. This section introduces a closed-loop framework that integrates observability, modeling, and optimization into a continuous control process.

In agentic AI systems, workloads evolve quickly. New agents are added, user demand rises, vector stores grow, prompt patterns change, model usage shifts, and retrieval behavior becomes more complex. A static sizing exercise is therefore not enough. Organizations need a control discipline that verifies whether assumptions remain true and adapts as the environment changes.

Observability-based intelligence is a foundation for organizing a closed-loop cost optimization and performance control system, as shown in **Error! Reference source not found.** [10, 19, 20]

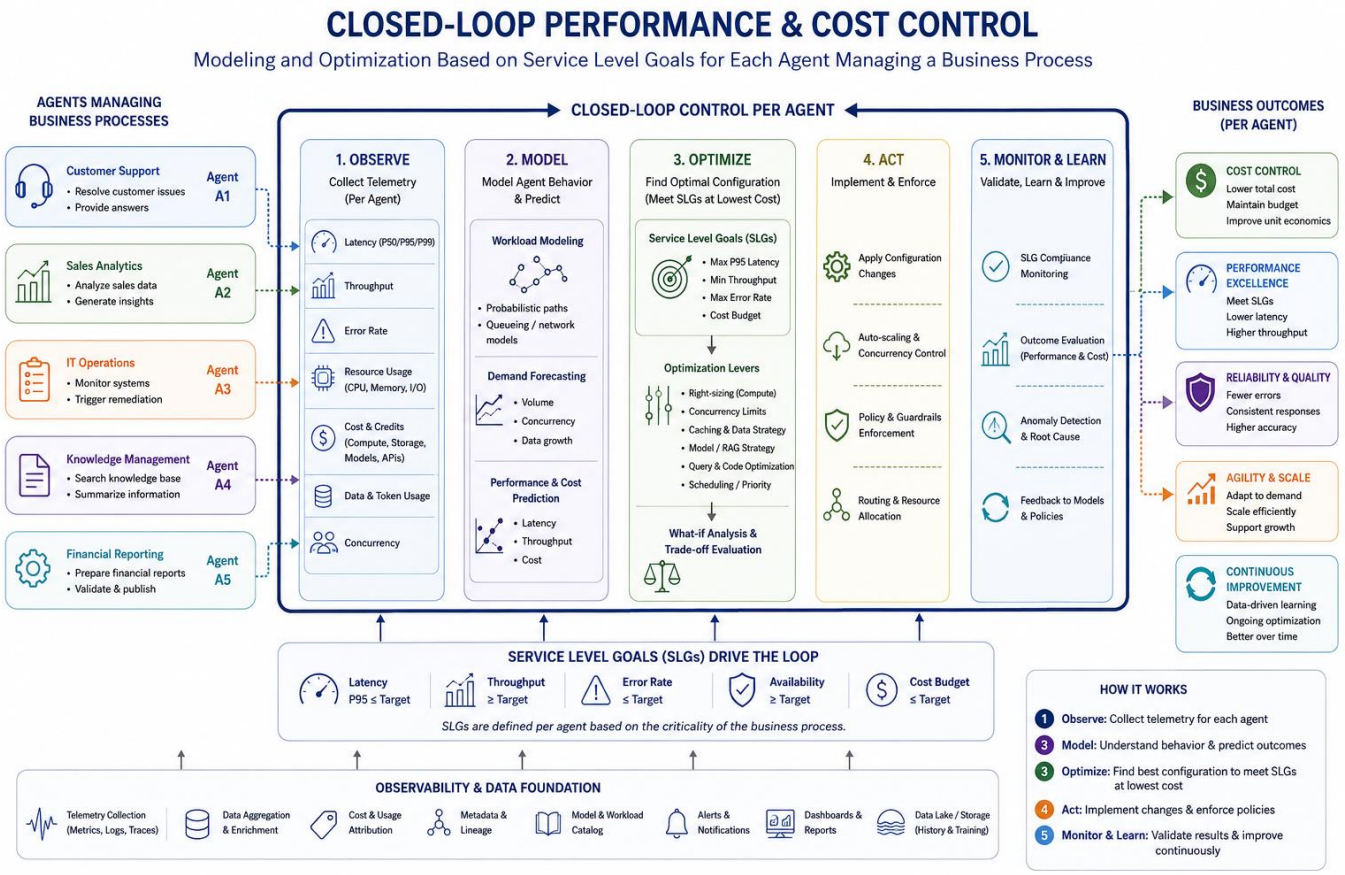


Figure 9.1. Observability information and Service Level Goals are used to optimize decision-making and to organize closed-loop cost and performance control for agent-based AI systems.

Takeaway: Closed-loop cost-performance control turns observability and service level goals into an operational discipline for continuously adapting to changing agentic AI workloads.

10 Business Value Across the Organization

The business value of observability automation and organizing performance and cost control spans financial, technical, and operational roles. Financial executives gain workload-driven budgeting and a clearer view of the cost drivers behind AI initiatives. IT executives gain a stronger basis for platform selection, capacity planning, and SLG-oriented governance. Business executives gain more predictable AI-enabled services and a lower risk of performance or budget surprises. Architects and engineering teams gain deeper insight into workload behavior and a stronger basis for comparing architectural options before implementation. Operations teams gain a disciplined way to prioritize root-cause analysis, anomaly response, and performance optimization.

Just as important, observability automation, performance, and cost control provide a common analytical language across these roles. By connecting business processes, agents, resources, costs, evaluation alternatives, and optimization aligned with Service Level Goals, the disconnect among financial reporting, engineering diagnostics, and business planning is reduced. That shared view is a precondition for governing agentic AI at the enterprise scale.

11 Conclusion

Agentic AI systems cannot be managed effectively with conventional monitoring alone. Their multi-layer architecture, probabilistic execution paths, variable workloads, and inconsistent telemetry require a more rigorous discipline.

Observability automation is the first step in that discipline, but not the last. Its real value lies in feeding workload characterization, analytical modeling, optimization, and closed-loop cost-performance control.

Organizations that adopt this approach are better positioned to select the right platforms, estimate the minimum configuration needed to meet Service Level Goals, size new applications before deployment, tune existing environments intelligently, and avoid cost and performance surprises as agentic workloads scale. In that sense, observability for agentic AI is not a tool category. It is a foundation for engineering predictability into systems whose behavior would otherwise be difficult to explain and even harder to control.

12 References

[1] J. R. Storment and M. Fuller, Cloud FinOps: Collaborative, Real-Time Cloud Value Decision Making, 2nd ed., O'Reilly Media, 2023.

[2] FinOps Foundation Working Group, "Cost Estimation of AI Workloads," 2024.

[3] FinOps Foundation Working Group, "How to Forecast AI Services Costs in Cloud," Jul. 2025.

[4] C. Majors, L. Fong-Jones, and G. Miranda, Observability Engineering, O'Reilly Media, 2022.

[5] OpenTelemetry, "Semantic Conventions for AI Systems," 2024–2025.

[6] J. Dean and L. A. Barroso, "The Tail at Scale," Communications of the ACM, 2013.

[7] D. A. Menascé, V. A. F. Almeida, and L. W. Dowdy, Performance by Design, Prentice Hall, 2004.

[8] D. G. Feitelson, Workload Modeling for Computer Systems Performance Evaluation, Cambridge University Press, 2015.

[9] L. Kleinrock, Queueing Systems, Volume 1: Theory, Wiley-Interscience, 1975.

[10] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," Computer, 2003.

[11] J. S. Park et al., "Generative Agents: Interactive Simulacra of Human Behavior," 2023.

[12] A. K. Pati et al., "Agentic AI: A Comprehensive Survey," 2025.

[13] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS, 2020.

[14] Anthropic, "Model Context Protocol (MCP) Specification," 2024.

[15] Snowflake Inc., "Snowflake Cortex AI Observability Documentation," 2025.

[16] Databricks, "Lakehouse Monitoring and ML Observability," 2025.

[17] Teradata Corporation, "Teradata VantageCloud Workload Management Documentation," 2025.

[18] Google SRE Team, Site Reliability Engineering, O'Reilly Media, 2016.

[19] B. Zibitsker and A. Lupersolsky, "Cost Optimization and Performance Control in the Hybrid Multi-Cloud Environment," ICPE 2025.

[20] B. Zibitsker, Agentic AI Cost Optimization and Performance Control, BEZNext Press, 2026.