https://doi.org/10.26160/2474-5901-2025-51-80-89

MONITORING IN MODERN IT INFRASTRUCTURE: PRACTICAL APPROACHES TO ENHANCING SYSTEM OBSERVABILITY

Tiumentsev D.V.
CJSC Integro Technologies, Moscow, Russia

Keywords: monitoring, observability, telemetry, distributed systems, OpenTelemetry, operational risks, streaming processing.
Abstract. The article examines modern approaches to monitoring and enhancing observability in distributed IT infrastructure under conditions of high dynamics, microservice architectures, and cloud deployment models. The limitations of classical resource-oriented monitoring are analysed, and the transition to the observability concept based on the correlation of metrics, logs, and distributed traces is substantiated. The practical aspects of component instrumentation, streaming telemetry processing, its aggregation, and analytical interpretation in a near real-time mode are emphasized. Special attention is paid to the impact of observability on operational risk management, reduction of service recovery time, and improvement of the predictability of distributed system behaviour. It is concluded that observability represents not only an extension of monitoring, but also a systemic architectural characteristic of modern high-load digital platforms.

МОНИТОРИНГ В СОВРЕМЕННОЙ IT-ИНФРАСТРУКТУРЕ: ПРАКТИЧЕСКИЕ ПОДХОДЫ К ПОВЫШЕНИЮ НАБЛЮДАЕМОСТИ СИСТЕМ

Тюменцев Д.В.
ЗАО Integro Technologies, Москва, Россия

Ключевые слова: мониторинг, observability, телеметрия, распределенные системы, OpenTelemetry, эксплуатационные риски, потоковая обработка.
Аннотация. В статье рассматриваются современные подходы к мониторингу и повышению наблюдаемости в распределенной IT-инфраструктуре в условиях высокой динамики, микросервисных архитектур и облачных моделей развертывания. Анализируются ограничения классического ресурсно-ориентированного мониторинга и обосновывается переход к концепции observability, основанной на корреляции метрик, логов и распределенных трассировок. Подчеркиваются практические аспекты инструментирования компонентов, потоковой обработки телеметрии, ее агрегации и аналитической интерпретации в режиме, близком к реальному времени. Отдельное внимание уделяется влиянию observability на управление эксплуатационными рисками, сокращение времени восстановления сервисов и повышение предсказуемости поведения распределенных систем. Делается вывод о том, что observability выступает не только расширением мониторинга, но и системной архитектурной характеристикой современных высоконагруженных цифровых платформ.

Introduction

Modern IT infrastructures are characterized by a high degree of distribution, dynamic variability, and heterogeneity of technological components, driven by the widespread adoption of microservice architectures, containerization, resource orchestration, and cloud deployment models. The operation of such systems is accompanied by continuous configuration changes, frequent release cycles, and

non-stationary load profiles, which fundamentally complicate the tasks of controlling their state and behaviour. Traditional monitoring approaches focused on the collection and analysis of aggregated operational metrics of individual nodes and services provide only a limited view of cause-and-effect relationships in distributed environments and often prove insufficient for timely diagnosis of complex incidents and latent performance degradation.

Under increasing requirements for reliability, availability, and engineering observability of digital services, the observability concept, based on comprehensive correlation of metrics, structured logs, and distributed traces, is gaining substantial practical importance. The aim of this article is to analyse practical approaches to enhancing the observability of modern IT infrastructures by comparing classical monitoring and the observability concept, as well as to assess their role in improving the controllability of distributed systems and reducing operational risks.

Main part. Monitoring as an element of modern IT infrastructure management

Monitoring in modern IT infrastructure represents a core managerial mechanism that ensures controllability of digital systems across all stages of their lifecycle, from design and testing to production operation and scaling [1]. At early stages, monitoring data are used for empirical validation of architectural decisions, assessment of load resilience, and identification of bottlenecks in storage, compute, and network subsystems. In production environments, monitoring forms the basis for capacity management, compliance with availability and performance targets, fault resilience analysis, and incident response. Within DevOps and SRE practices, monitoring is embedded into continuous operations as a closed feedback loop linking real system behaviour with architectural and operational decisions.

As digital platforms evolve toward high-load distributed systems, monitoring becomes an integral element of architectural governance, directly associated with scalability, fault tolerance, and predictability of operational characteristics. Control of compute nodes, network subsystems, and application components enables evaluation of architectural patterns, identification of structural constraints, and development of justified horizontal and vertical scaling strategies [2].

Classical monitoring, originally designed for static and monolithic systems, is primarily focused on aggregated resource metrics such as CPU, memory, disk, and network usage. While adequate for baseline infrastructure control, this approach becomes limited in microservice environments characterized by dynamic orchestration, autoscaling, and service meshes. These conditions reduce the diagnostic value of isolated metrics and drive the adoption of observability-driven monitoring focused on end-to-end execution analysis and cross-service correlation. A generalized comparison of these approaches is presented in table 1.

The differences summarized in table 1 show that the architectural and operational characteristics of cloud-native and microservice-based systems cannot be adequately addressed by resource-centric, threshold-based monitoring alone, as dynamic scaling, asynchronous interactions, and growing inter-service complexity increase the likelihood of non-obvious failure scenarios. In such environments,

operational risks are primarily driven by latent degradations, internal latency accumulation, consistency violations, and cascading partial failures, which are not reliably detected through static metrics and require temporal analysis of system dynamics, including execution chains and inter-service dependencies. Under these conditions, a conceptual shift occurs from observing infrastructure "state" to analysing system "behaviour" as an integrated distributed control loop, forming the methodological basis for the transition from classical monitoring to the observability paradigm focused on behavioural interpretation of modern IT infrastructure.

Tab. 1. Classical vs observability-driven monitoring in IT infrastructures

Dimension	Classical monitoring	Observability-driven monitoring
Object of observation	Isolated physical or virtual nodes and standalone service instances.	End-to-end distributed execution across multiple interconnected services.
Primary data	Low-level infrastructure metrics collected at host or VM level.	Correlated metrics, structured logs, and distributed request traces.
System topology	Predominantly static or slowly changing infrastructure topology.	Highly dynamic topology controlled by orchestration and autoscaling.
Component lifetime	Long-lived servers and persistent service instances.	Short-lived containers, pods, and serverless runtime instances.
Failure detection	Threshold-based triggering on individual metric deviations.	Pattern-based detection using temporal correlations and anomalies.
Root cause analysis	Manual analysis of local metric deviations.	Automated or semi-automated reconstruction of request paths and dependency graphs.
Suitability for microservices	Limited support for highly distributed service meshes.	Native support for microservice, container, and cloud-native architectures.

The concept of observability and its technological implementation

The concept of observability in the context of modern IT infrastructure originates from control theory, where observability describes the ability to reconstruct the internal state of a system from its external outputs [3]. When this approach is transferred to distributed computing systems, observability is understood as the ability of an engineering team to draw well-founded conclusions about the state and behaviour of services based on available telemetry, without having direct access to all internal components [4].

According to an industry survey by Logz.io, most organizations recognise the importance of observability for their applications and infrastructure (fig. 1).

The results of the 2024 Observability Pulse survey are based on 501 responses from specialists in various roles, ranging from software developers and DevOps engineers to IT executives, representing companies of different sizes (from 1-20 to more than 5,000 employees) across all major global regions, including the

Americas, Europe, the Middle East, and the Asia-Pacific region. According to the findings, the most widely used observability practices at present remain the use of metrics to gain insights into systems and applications (64%) and the consolidated analysis of logs and metrics (61%), indicating the predominance of relatively basic observability approaches. At the same time, more than half of the respondents reported tangible benefits from observability adoption: 60% indicated improved and faster troubleshooting processes, while 56% reported achieving centralized access to application and infrastructure data.

Fig. 1. Results of an industry survey on the current level of observability adoption across organizations in 2024 [5]

In contrast to classical monitoring, which focuses on tracking a predefined set of metrics and threshold states, observability emphasizes the ability to infer and explain the internal state and behavior of a system during incidents, performance degradations, and anomalies based on its telemetry data. In essence, this represents a transition from static control of individual indicators to the construction of an integrated behavioural model of a distributed application. The technological implementation of observability is based on the consolidation of three core classes of telemetry data: metrics, logs, and distributed traces (table 2).

In practical implementations of observability, standardized telemetry protocols and formats are becoming increasingly widespread, as they help minimize tight coupling between applications and tooling. The most illustrative example is the use of **OpenTelemetry** as a unified instrumentation layer for collecting metrics, logs, and traces, followed by their routing to analytical storage systems and visualization platforms. According to recent studies by Splunk and Enterprise Strategy Group, OpenTelemetry is used in 78% of organizations with a mature monitoring and observability practice, and incidents in such environments are detected on average 2.8 times faster than in organizations with a low level of observability maturity [7]. In addition, data from the OpenTelemetry end-user community confirm the broad industrial adoption of this technology as a baseline telemetry collection mechanism in modern distributed applications (fig. 2).

Thus, the data presented in the figure indicate not only the widespread industrial adoption of OpenTelemetry, but also the growing scale of observability system operations in distributed environments. In typical engineering stacks, metrics are aggregated and processed by time-series databases, logs are indexed and analysed using text-based or document-oriented platforms, and traces are visualised

and examined through specialised tracing engines. The consolidation of these components into a unified observability architecture enables the transition from fragmented monitoring dashboards to coherent, end-to-end diagnostics of application and infrastructure behaviour [9].

Tab. 2. Core telemetry data classes in observability architecture [6]

Telemetry class	Collection level	Typical parameters	Functional purpose	Engineering examples
Metrics	Infrastructure, platform, application	Latency, throughput, error rate, CPU, memory, disk I/O, network I/O, saturation	Quantitative assessment of performance, availability, and reliability; SLI/SLO definition and alerting	Prometheus metrics, Kubernetes metrics, RED/USE
Logs	Application, middleware, infrastructure	Errors, exceptions, business events, system messages, execution context	Detailed recording of events and execution context for diagnostics and incident investigation	JSON application logs, audit logs, system logs
Distributed traces	Inter-service communication	Trace ID, Span ID, operation duration, service dependencies, network latency	End-to-end request path reconstruction, latency analysis, and causal failure analysis	OpenTelemetry, Jaeger, Zipkin

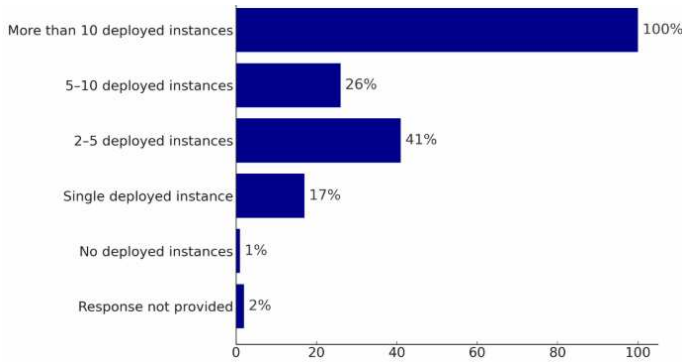

Fig. 2. Distribution of organizations by the number of deployed OpenTelemetry Collector instances based on the 2024 survey, % [8]

Streaming telemetry processing enables near real-time observability and becomes necessary as periodic data collection is insufficient for timely anomaly detection in high-volume distributed systems. Real-time pipelines based on message brokers and stream processing platforms support continuous filtering, aggregation,

enrichment, and ML-based correlation of telemetry, transforming observability into a continuously operating analytical layer. The key differences between batch and streaming telemetry processing are summarized in table 3.

Tab. 3. Comparison of batch and streaming telemetry processing architectures in observability

Parameter	Batch telemetry processing	Streaming telemetry processing
Data collection mode	Periodic data export at fixed intervals.	Continuous real-time data ingestion.
Anomaly detection latency	Minutes to hours.	Seconds to tens of seconds.
Incident response model	Post-factum, after data accumulation.	Near real-time detection and response.
Scalability	Limited by batch size growth.	Horizontally scalable.
Correlation analysis	Restricted to fixed time windows.	End-to-end, event-stream correlation.
Context enrichment	Typically performed offline.	Performed in-stream (topology, config, releases).
Support for ML-based detection	Limited, offline models.	Online and quasi-real-time models.
Typical platforms	Cron + ETL, Data Warehouse.	Kafka, Pulsar, Flink, Spark Streaming.
Role in observability	Retrospective analytics.	Continuous operational analytics.

From an engineering perspective, a mature observability implementation requires not only appropriate tools but also consistent practices for instrumentation and telemetry management, including standardized tracing and metrics, unified labeling, integration with orchestration platforms, defined retention policies, and SLO-based incident analysis. This enables a shift from a dashboard-driven approach to hypothesis-driven investigations, where the observability stack supports rapid identification of failure and degradation causes. In this sense, observability becomes not merely an extension of monitoring, but an architectural property that defines the transparency and controllability of distributed systems in real operating conditions.

Practical approaches to telemetry collection, aggregation, and analysis

Practical approaches to the collection, aggregation, and analysis of telemetry data in modern observability systems are implemented as an end-to-end engineering pipeline that encompasses instrumentation, streaming processing, storage, and analytical interpretation of telemetry. In highly dynamic distributed environments, the effectiveness of observability is determined not by individual tools, but by the coherence of all processing stages operating in a near real-time mode. The structure of this pipeline is presented in table 4.

Tab. 4. Stages of the telemetry collection and analysis pipeline in observability architecture [10]

Stage	Purpose	Key operations	Typical tools
Instrumentation	Embedding telemetry into software components.	Metrics, logs, and traces injection into code and middleware.	OpenTelemetry SDK, sidecar agents.
Collection	Receiving telemetry from sources.	Ingestion, buffering, basic filtering.	OpenTelemetry Collector, agents.
Streaming aggregation	Real-time transformation of telemetry flows.	Filtering, normalization, enrichment.	Kafka, Pulsar, Flink, Spark Streaming.
Storage	Long-term data persistence.	Indexing, compression, retention.	TSDB, log storage, trace storage.
Analytical processing	Detection of anomalies and patterns.	Correlation analysis, ML-based detection, RCA.	Observability platforms, ML modules.
Visualization and control	Operational monitoring and control.	Dashboards, alerting, SLO monitoring.	Grafana, Kibana, APM systems.
Operational decision-making	Engineering and management actions.	Incident response, optimization, capacity planning.	ITSM, SRE toolchains.

The presented pipeline reflects the transition from fragmented telemetry collection to a coherent architecture of operational analytics. At the stages of collection and streaming aggregation, the integration of OpenTelemetry standards with centralized logging and analytics systems plays a key role, enabling the construction of a unified telemetry contour for large-scale distributed systems [11]. The streaming processing model significantly reduces anomaly detection latency and makes it possible to correlate events at the level of inter-service interactions before local degradations are transformed into user-visible incidents.

At the stage of analytical telemetry processing, diagnostic methods are extended through the use of behavioural analytics and the correlation of technical metrics with indicators of user activity and engagement [12]. These approaches are of particular practical importance in serverless and FaaS architectures, where latency and runtime initialization introduce additional operational risks and require detailed trace-based analysis of call chains [13]. Taken together, this enables a shift from reactive monitoring to proactive management of distributed system state based on continuous analysis of telemetry streams.

Impact of monitoring and observability on operational risks

Operational risks in modern distributed systems are shaped by high infrastructure dynamics, complex inter-service dependencies, asynchronous interactions, and variable load profiles. Under such conditions, classical monitoring, primarily focused on resource metrics and threshold states, is able to effectively detect only a limited class of incidents related to explicit node failures and resource

exhaustion. A **significant portion of critical operational risks is associated with latent performance degradation,** accumulation of internal latency, component desynchronization, cascading partial failures, and violations of service interaction contracts, which are often not accompanied by immediate threshold breaches and therefore remain poorly detectable by traditional monitoring approaches. Industry research confirms this gap: according to the Splunk State of Observability 2024 survey, organizations with low observability maturity are significantly less likely to achieve mean time to recovery (MTTR) in the range of minutes or hours, whereas high-maturity organizations are 2.3 times more likely to maintain MTTR within these time ranges despite high incident volumes and the substantial cost of downtime for large enterprises.

The introduction of observability fundamentally expands the ability to manage operational risks by shifting from static observation of individual component states to **analysing the behaviour of the system as an integrated distributed whole**. Correlating metrics, logs, and traces enables causal analysis of failures at the level of end-to-end user scenarios, providing more accurate localisation of degradation sources, reducing incident diagnosis time, and lowering the MTTR. Practical cases reported by major observability platform vendors show that centralizing telemetry and abandoning a fragmented “tool zoo” lead to a marked reduction in MTTR: in the Grafana Observability Survey 2024, several companies report a 30–35 % decrease in MTTR and a parallel reduction in operational overheads by almost one third after transitioning to a unified observability platform model [14].

Quantitative effects of observability adoption are also confirmed by industry reports at the level of specific sectors. In the New Relic State of Observability study for the industrial and manufacturing sector, 65% of enterprises report **accelerated incident resolution** after implementing a full-scale observability architecture, while in the energy and utilities sector observability is emphasized as a foundational element for the reliability of digital services in the context of critical infrastructure. In selected cases described in analytical reviews and vendor validation reports, MTTR has been reduced from weeks to hours as a result of telemetry consolidation, the elimination of fragmented toolsets, and the introduction of a unified data access layer for engineering teams. In addition, it is noted that the application of automated root cause analysis and AI-supported signal correlation **enables MTTR reduction for critical incidents** [15].

Beyond direct technical effects, monitoring and observability have a significant **impact on the manageability and economic performance of IT** infrastructure operations by increasing system transparency, improving predictability under load, and supporting justified scaling decisions. Within the SRE framework, observability serves not only reactive response but also preventive reliability management based on SLOs and error budgets, enabling formalized risk control and a balance between change velocity and system resilience. As a result, the probability of severe incidents and losses from downtime and performance degradation is reduced.

Conclusion

Monitoring in modern IT infrastructure is evolving from an auxiliary control tool into a system-forming element of operational reliability management for distributed digital platforms. The transition from resource-centric monitoring to a behavioural observability model based on the correlation of metrics, logs, and distributed traces provides a qualitatively new level of diagnostics, enabling the detection of latent degradations, cascading failures, and hidden dependencies between system components. Practical approaches to telemetry collection, aggregation, and analysis, implemented through streaming architectures and standardized instrumentation, form the foundation for continuous operational analytics in a near real-time mode. The impact of observability on operational risk management is reflected in reduced service recovery times, improved predictability of system behaviour, and decreased economic losses from incidents. Thus, observability should be regarded not as an extension of classical monitoring, but as an inherent architectural property of modern high-load distributed systems that determines their resilience and controllability under real-world operating conditions.

References / Список литературы

1. Allam H. Metrics that Matter: Evolving Observability Practices for Scalable Infrastructure // International Journal of AI, BigData, Computational and Management Studies. 2022, vol. 3, no. 3, pp. 52-61.
2. Berezhnoy A. Architectural design patterns for high-load systems: principles, tools, and scalability constraints // Professional Bulletin. Information Technology and Security. 2025, no. 3, pp. 33-39.
3. Kansara M. The role of observability in modern cloud database architectures // International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 2025, vol. 11, no. 2, pp. 2558-2576.
4. Nuzhdin D. Architecture, UX, and personalization algorithms of digital platforms for personal branding and business communications // International Journal of Engineering in Computer Science. 2025, vol. 7(2B), pp. 180-185.
5. 2024 Observability Pulse Report / Logz.io [Electronic resource]. – Access mode: https://logz.io/observability-pulse-2024/
6. Wang X.Z., Tseng C.C. Building an Observable System Based on OpenTelemetry // 2024 International Computer Symposium (ICS). IEEE, 2024, pp. 76-81.
7. The State of Observability 2024: Key Findings & Leadership Insights / Splunk [Electronic resource]. – Access mode: https://discover.splunk.com/The-State-of-Observability-2024-Key-Findings-and-Leadership-Insights.html.
8. Insights from the OpenTelemetry Collector Survey / OpenTelemetry [Electronic resource]. – Access mode: https://opentelemetry.io/blog/2024/otel-collector-survey/
9. Bogutskii A. Enhancing the resilience of stream processing platforms: engineering approaches to scalability and fault tolerance // ISJ Theoretical & Applied Science. 2025, no. 08(148), pp. 170-175.
10. Patel H.R. A Maturity Model for Observability in Big Data Pipelines: From Reactive Monitoring to Predictive Resilience // Journal of Computer Science and Technology Studies. 2025, vol. 7, no. 11, pp. 195-202.

11. Smirnov A. Monitoring and logging in distributed systems: application of OpenTelemetry and the ELK stack // Universum: technical sciences: electronic scientific journal. 2025, no. 3(132), pp. 30-33.
12. Andreev G. Development of ranking systems based on engagement metrics: hotness algorithms and user interest models // International Journal of Latest Engineering and Management Research. 2025, vol. 10(9), pp. 1-6.
13. Maksimov V.Yu. Startup Latency Analysis in Java Frameworks for Serverless AWS Lambda Deployments // The American Journal of Engineering and Technology. 2025, vol. 7, no. 04, pp. 16-21.
14. Observability Survey 2024 / Grafana [Electronic resource]. – Access mode: https://grafana.com/observability-survey/2024/
15. Don’t just react: How executives can predict and prevent outages to maximize availability / Dynatrace [Electronic resource]. – Access mode: https://www.dynatrace.com/news/blog/dynatrace-for-executives-improved-availability.

Тюменцев Денис Викторович – ведущий DevOps-инженер	**Tiumentsev Denis Viktorovich** – lead DevOps engineer
tyumencev_dv@rambler.ru	

Received 08.12.2025