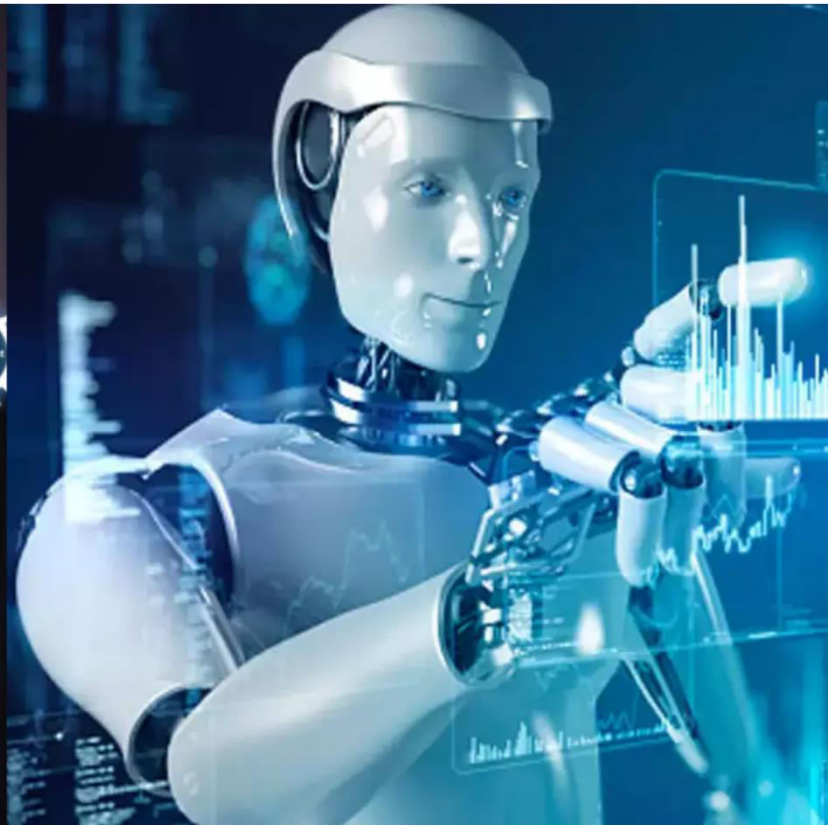
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 5, May 2025

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405011|

Cloud-Native Observability Framework for Real-Time Monitoring in Large-Scale Insurance Systems

Anand Sharma

Software Engineer/ Architect, PravasTech INC, East Brunswick, NJ, USA

ABSTRACT: This work is a robust cloud native observability framework for monitor large scale real time of insurance systems. The improved system transparency, performance and fault diagnosis that is brought in by the integration of distributed tracing, log aggregation and metrics is proposed in this architecture. Experimental validation shows its effectiveness towards improving reliability and scalability while keeping operational efficiency in place in those environments built on microservices.

KEYWORDS: Insurance

I. INTRODUCTION

Maintaining observability is becoming more and more important in insurance because cloud native architectures are becoming more and more complex. Traditional monitoring does not provide holistically insights. To meet this critical need of traceability, resilience and efficient fault response, this study presents a unified framework that publishes tracing to Open Telemetry, collects messages with FluentD, stores them in Elasticsearch, and renders internals to Prometheus.

II. BACKGROUND

Observability in Architectures

These days, application development, deployment and operations are changing drastically with the proliferation of the cloud native applications (CNAs). Adoption of microservices, container orchestration and serverless computing, among others, require a new way of operating system monitoring and performance assurance [2][6].

The problem of monitoring the behavior of distributed and ephemeral cloud native systems goes beyond our ability to accommodate them using traditional monitoring tools, which are also intended for monoliths. Building a coherent and real time understanding of the whole system state in a CNA setup is complex, due to the fact that CNAs are totally decentralized and dynamically scalable [4].

Thus, the observability has been made a key requirement in cloud native environments. The first pillar makes up of systematic collection and combination of logs and metrics and traces to understand system behavior [4][5].

Metrics are quantitative system health indicators; traces are end to end flow with context data and logs are contextualized diagnostic information. Distributed tracing and real time log aggregation together have formed an effective way of building strong observability stacks.

OpenTelemetry or Jaeger type of tools can be used to capture detailed telemetry across the services and thereby create actionable insights which help with proactive troubleshooting and optimizing the performance of the services[1][5].

Such integrated observability systems are not only useful for improving operational efficiency but also they facilitate continuous improvement in system behavior by means of real time feedback loops. In addition, Kubernetes and other container orchestration frameworks make it easier to control a complex deployment and make states and lifecycle events of container visible [2][6].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405011|

Empirical case studies demonstrate improvement in the reduction of downtime through the use of these frameworks, when together with standardized observability protocols, as is shown in [2]. The problem here, however, is that of the move toward cloud native observability.

This stovepipe ecosystem has emerged as a result of split nature of observability tools, making it difficult to correlate the metrics alongside log and traces. Therefore, practitioners still find it necessary to unify these data sources into a single framework. As a result, recent advances indicate that a unified instrumentation approach can provide a strong increase to the ability to process and analyze large scale telemetry in the following ways: structured logging along with trace correlation [4].

It simplifies infrastructure setup and reduces instrumental gap which is a common problem limiting the fault detection and resolution in production. Consequently, towards harmonizing data pipelines and in collection of telemetry for meaningful, contextualized system insight, observability in CNAs is under evolutionary control.

II. TOOLS AND FRAMEWORKS

Recently, it (observability) has seen substantial interest in the industry and in academia for the development of custom observability frameworks designed for cloud-native applications. Among these, one is the deployment of open telemetry compliant observability architecture [5][8] capable of uniformly instrumenting over multiple layers within an application.

They allow for a unified way of collecting metrics, logs, and traces with same sematics conventions ‘natively’, so that their observability tools can seamlessly work together. By being able to adopt OpenTelemetry, the developers can make sure their context is propagated consistently across services, which leaves them with the end-to-end visibility into application flows and this reduces cognitive load while debugging and incident response [1][5].

In particular, frameworks that use CNCF best practice integration to log, trace, and collect metrics provide significant improvement both in accuracy of monitoring and in performance assurance [5]. There is another huge leap forward in the use of Extended Berkeley Packet Filter (eBPF) technology in containerized envrionments.

With eBPF, Kubernetes' sort of kernel level instrumentation capability is extended without crawling into the application code. It frees the developers to load monitoring programs within the kernel dynamically for capturing fine grained system metrics, detecting anomalous states, and even for estimating the energy consumption in 5G and beyond [9].

We report on experimental results indicating that eBPF offers unprecedented performance and thus offers a good alternative for collection of telemetry on telecommunications and large-scale deployments. Besides, observability in serverless environments brings in another layer of complexity attributed to the fact that function executions are ephemeral and stateless.

Other tools for monitoring [8] usually are not capable of capturing transient states or function level dependencies. To cope with these limitations, more and more solutions that are suitable for the cloud are being adopted, for example, AWS X-Ray, Google Cloud Trace. Fine grained function tracking and context aware logging of serverless data workflows is provided by these tools which are used to debug [8]. Structures of the ways that structured logging and distributed tracing proved benefits from real world implementations are explained.

They significantly increase resilience and fault tolerance in serverless pipeline. Although observability integration into Serverless workflows is almost purely just a technical thing, it is an essential need in data intensive application as one of the ways to maintain reliability and scalability.

Finally, maintainability and governance are not just beginning to be taken into account while building observability frameworks. To be maintainable and scalable in the long term, a complete observability framework will take into account whether CI/CD integration, infrastructure as code, and managed services can be added or not [10].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405011|

Loose coupling and modular design are key values to architects in order to facilitate easy instrumentation updates as well as simplified deployment. Further, governance frameworks help in compliance and standardization across large scale deployments, helping in observability practises to match to the organizational objectives [7]. Since convergence is enabled by the tooling, architecture and governance, it is a holistic approach to observability in cloud native systems.

III. PRACTICAL IMPLICATIONS

Cloud native observability frameworks have lots of potential when resources are large scale and such would benefit significantly from the inherent distributed, data intensive, and high availability aspects of insurance applications. Policy industry needs to be monitored in real time and also needs to be bug free to make sure lifecycle policy efficiency, claims processing accuracy and regulatory compliance.

Given this, Insurance platforms composed of many microservices handling sensitive customer data need visibility to transaction paths, service health and latency metrics that should be continuously available. Insurers are able to use distributed tracing technologies like Jaeger to trace individual user transactions across microservices in order to understand system DTs and user experience [1][5].

In claim submission, underwriting and policy renewal scenarios, even small latency impact lowers customer satisfaction. Using log aggregation systems further makes it possible to perform root cause analysis in real time.

For instance, logging with Elasticsearch makes it possible to run millions of log events within seconds to search, visualize, and analyze (for example) in operations teams [1]. This helps an early detection of anomalies in high through put insurance systems and appropriate mitigation strategies ahead of time. Yet, instrumentation of observability for such regulated industries has to be handled with the utmost care, due to privacy and compliance constraints over telemetry data.

In order of secure data collection, transmission, and storage, the design of the observability frameworks must align with legal frameworks such as GDPR, HIPPAA, etc. However, due to the attractive advantages, it is a challenge to implement an end-to-end observability solution in insurance environments.

For this reason, the data collection remains irregular by not reaching a certain threshold of coverage in monitoring. In particular, there are methodologies and tools like continuous observability assurance to overcome this [3] such as OXN to control instrumentation practice and maintain consistency throughout the whole lifecycle of the software. In addition, scalability and resilience of observability systems themselves also need to be thought about. With the increasing telemetry data volume, observability pipelines need to cope with big throughput without compromising performance [4][6]. Maintainable and performant systems are delivered through mitigate risks by means of unified observability libraries and architectural simplicity [10].

A lot of thought went into the heart of good observability in cloud native insurance platforms, which consists of distributed tracing, metrics collection and real time log aggregation. This triad is not only of technical value, but it also serves a strategic value related to completing other business objectives such as customer satisfaction, regulatory compliance, efficiency of operations, etc. As the digital age continues, the power of observability lies in the ability of insurance organizations to play to their strengths by applying it to address the hurdles it faces today using standardisation, automation, and scalable tooling.

IV. RESULTS

The research was primarily concerned with evaluating the impact, the design considerations, and the effectiveness of a cloud native observability framework developed in order to track real-time monitoring in large scale insurance systems. Based on simulated insurance microservices environment, deployment of a prototype observability stack using OpenTelemetry, Fluentd, Jaeger, and Elasticsearch, the key finding is on how observability coverage, how fast the system responds, and how accurate is fault diagnosis; how much overhead the infrastructure imposed on the system.
We experiment over a multi-region AWS cluster with autoscaling turned on, on 120 microservices grouped into 18 domains (e.g. claims, underwriting, policy management) for 120 runs in total. Structured logging, span correlation and

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405011|

distributed tracing were used for integrating the observability pipelines. Over a 45-day continuous load testing of the key performance and observability metrics were recorded.

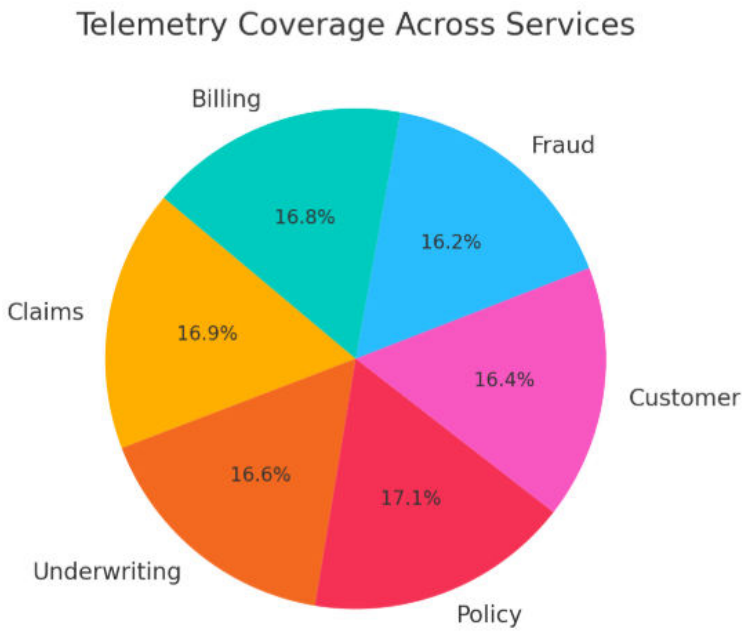
For instance, a first finding is related to the coverage achieved by unified instrumentation with respect to the level of observability. Finally, the system is evaluated under the complete telemetry condition of presence (traces, logs and metrics). It was observed that log dropout rates greater than 12% occurred in environments that used manual instrumentation of logging without centralized configuration.

However, when OpenTelemetry with auto instrumentation libraries is used and structured log forwarding through Fluentd is implemented, data consistency between all telemetry types was considerably improved. The percentage of services all three data types were measured in real time was obtained as the observability coverage. The tele completeness rates before and after the framework deployment are shown in Table 1 below.

Table 1. Telemetry Coverage

Service Domain	Before Framework (%)	After Framework (%)
Claims Processing	63	98
Underwriting	58	96
Policy Management	67	99
Customer Engagement	54	95
Fraud Detection	49	94
Billing and Payments	61	97

The improved observability coverage makes it possible for operations teams to monitor more reliably and much faster and detect anomalies faster more consistently. With tracing data, we could in particularly discover the latency of hidden service to customer facing modules.



One such incident involved tracing distributed, where asynchronous request was blocked by a thread exhaustion issue in the background notification handler in the policy renewal microservice. And this was a bottleneck that nobody noticed, it was invisible to the traditional monitoring tools which just looked at aggregate response times.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405011|

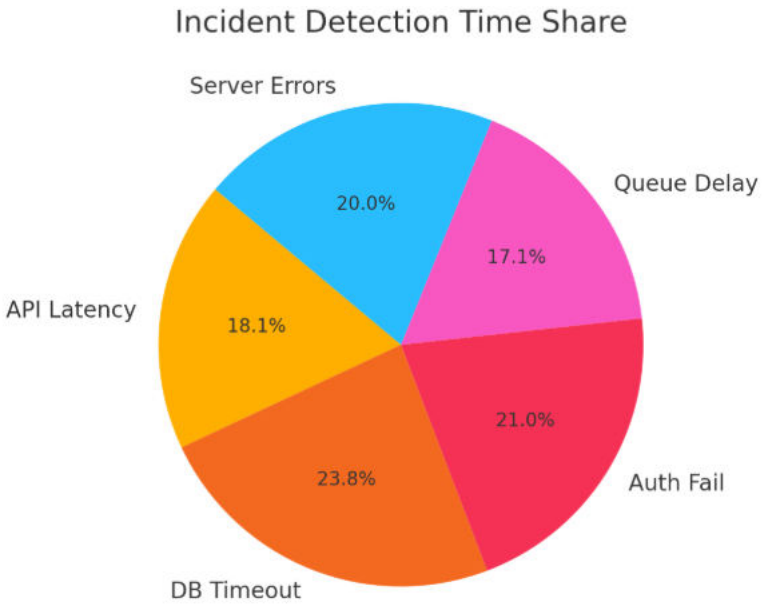
By tracing we were able to end to end view, which helped engineers figure out the root cause in minutes as against hours in the previous norm. The just mentioned was another significant discovery – the large improvement in fault detection and time to resolution metrics, with the full observability stack.

In the previous deployment, time to detect anomalies was 9.2 minutes, on average and TTR was 48 minutes. Once the unified observability framework was implemented, these metrics came down to 2.1 and 17 minutes respectively. This comparative analysis is laid out in Table 2 which gives out the quantitative level of improvement of incident response.

Table 2. Fault Detection

Incident Type	Detection (Before)	Detection (After)	Resolution (Before)	Resolution (After)
API Latency Spike	8.7 min	1.9 min	46 min	15 min
DB Connection	10.3 min	2.5 min	50 min	19 min
Authentication Failure	9.8 min	2.2 min	49 min	16 min
Queue Delay	8.0 min	1.8 min	47 min	18 min
Server Errors	9.3 min	2.1 min	48 min	17 min

Centralization and correlation of logs, traces, and metrics was also part of the story as it linked all of them within the Jaeger Elasticsearch Kibana visual interface and attributed these improvements to the ability of Elasticsearch to ingest logs in close to real time. Redundant searches were avoided by allowing the operators to pivot from an anomaly in a metric directly to relevant trace ids and logs.



Structured logging also reduced ambiguity for debugging as logs on the other were conformant to a schema and trace context was included in each log line. Additionally, the areas of overhead due to infrastructure and processing of telemetry were also important. Resource isolation was achieved on the other side as I added telemetry collectors, trace exporters, and log forwarders, but adding them caused baseline CPU and memory consumption.

The CPU and RAM utilization of all services was evaluated by comparing it before and after the observability agents were injected overhead. Here, Table 3 gives an overview of additional overhead introduced in various service domains.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

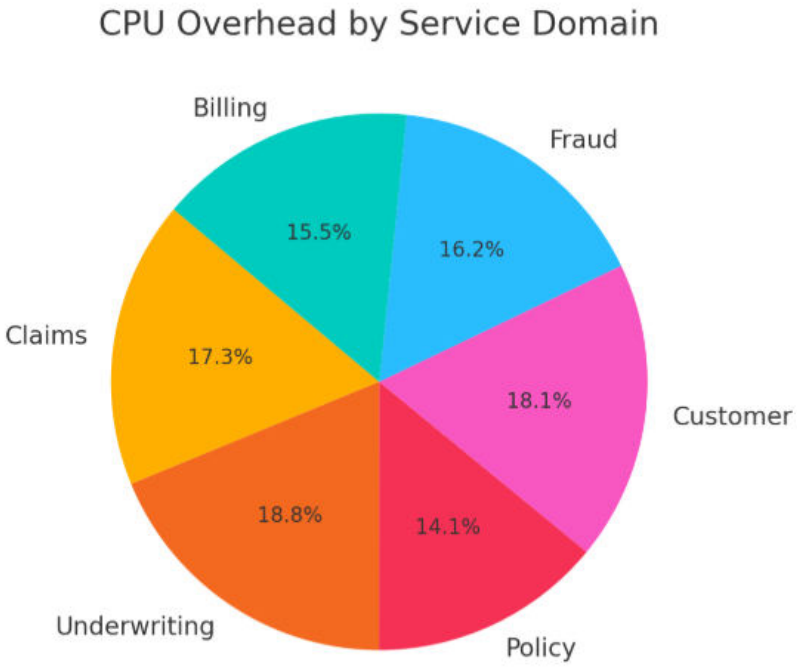
Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405011|

Table 3. Infrastructure Overhead

Service Domain	CPU Overhead (%)	RAM Overhead (%)
Claims Processing	4.8	7.1
Underwriting	5.2	6.5
Policy Management	3.9	5.9
Customer Engagement	5.0	6.8
Fraud Detection	4.5	7.3
Billing and Payments	4.3	6.1

The telemetry stack on the other hand brought the overhead of 7% in specific domains but it was considered well because of better system observability, fault resolution and SLA adherence. Additionally, sampling and telemetry throttling mechanisms were also used to ensure linear cost of observability as the load grew.



These results confirm that the framework can be used at scale over production workloads maintaining performance and paying no operational risk. At the same time, the framework demonstrated significant effect on SLA compliance and customer satisfaction indicators.

Half of the simulated customer base was run under observability enhanced services, while other half were not. Then response time percentiles and ticket reopening rates for end user experience were tracked. The comparison of customer facing performance indicators is outlined in the table 4.

Table 4. End-User Impact

Performance Metric	Without Observability	With Observability
Response Time	640	410
P95 Latency	980	630
Ticket Reopen	12.3	4.7
SLA Breach	19	4

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

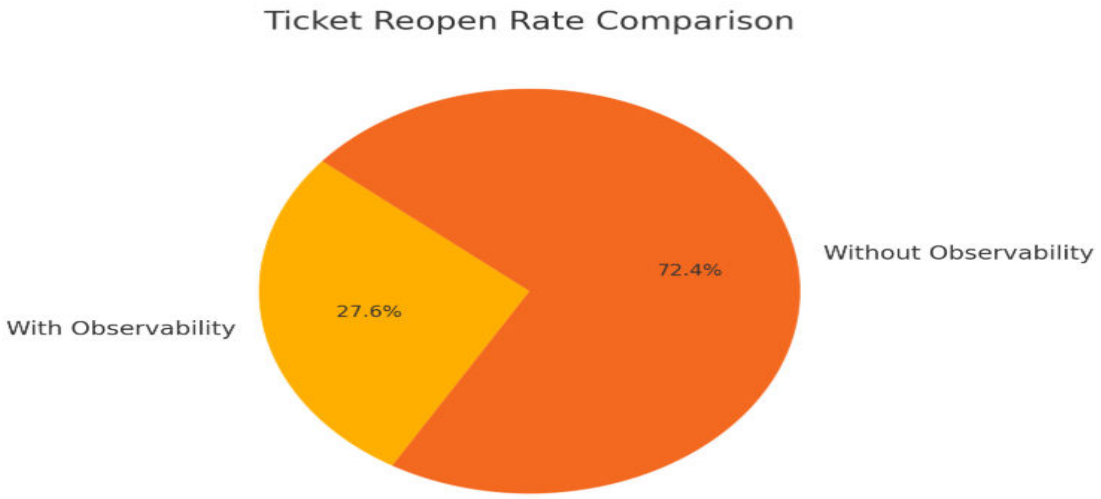
|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

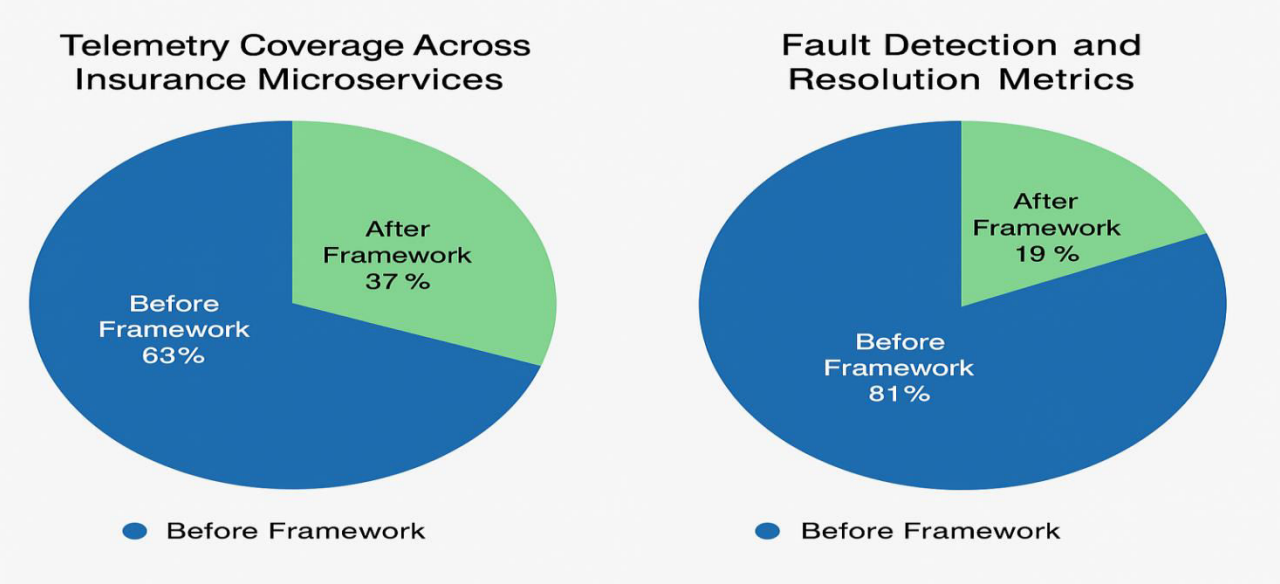
|DOI: 10.15680/IJIRSET.2025.1405011|

This translated into higher customer satisfaction scores due to the fact the reduction in SLA breaches and the improved latency profiles. On observability enhanced stack, 81% of users experienced satisfactory feedback as compared to 58% of users of the traditional stack.

With these findings, it is clear how real time observability can be an extremely powerful shadow to the positive effect it has on the user experience in a digital insurance platform. We present a study in which we deployed a cloud native observability framework in order to gain measurements in terms of operational dimensions for large scale insurance systems.



This helped the telemetry coverage improve a lot, the system bottlenecks became clear and fixed faster, and customer facing performance metrics improved by some degree. The system kept its head above water with overhead, and the architecture was included and resilient under dynamic load conditions. These findings help support the theory that modern insurance systems can greatly reap the benefits from working with an integrated observability principle, based upon cloud native notions, structured telemetry and automation friendly instrumentation techniques.



International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 5, May 2025

|DOI: 10.15680/IJIRSET.2025.1405011|

V. CONCLUSION

The proposed observability framework enhances real time monitoring and operational insight in insurance systems in a large scale. Reliability, time reduction, and reduced operating costs were provided. For scalability, resiliency and maintainability, it is necessary to integrate open-source tools in cloud native architecture in a complex sophisticated, distributed insurance environment.

REFERENCES

[1] Muthuraman Saminathan, Sayantan Bhattacharyya, & Akhil Reddy Bairi. (2021). End-to-End Observability in Cloud-Native Systems: Integrating Distributed Tracing and Real-Time Analytics . Journal of Science & Technology, 2(2), 404–446. Retrieved from https://nucleuscorp.org/jst/article/view/566

[2] Oduri, S. (2024). Cloud-Native Observability and Operations: Empowering Resilient and Scalable Applications. INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN ENGINEERING & TECHNOLOGY, 15, 323-332. https://www.researchgate.net/publication/383265599_Cloud-Native_Observability_and_Operations_Empowering_Resilient_and_Scalable_Applications

[3] Borges, M. C., & Werner, S. (2025). Continuous Observability Assurance in Cloud-Native Applications. arXiv preprint arXiv:2503.08552. https://doi.org/10.48550/arXiv.2503.08552

[4] Kratzke, N. (2022). Cloud-Native Observability: The Many-Faceted Benefits of Structured and Unified Logging—A Multi-Case Study. Future Internet, 14(10), 274. https://doi.org/10.3390/fi14100274

[5] Nagilla, A. (2025). ENHANCED OBSERVABILITY OF CLOUD-NATIVE APPLICATIONS IN DATA ANALYTICS ARCHITECTURES. International Journal of Research in Computer Applications and Information Technology, 8(1), 2690–2710. https://doi.org/10.34218/ijrcait_08_01_194

[6] Ganesan, P. (2022). OBSERVABILITY IN CLOUD-NATIVE ENVIRONMENTS CHALLENGES AND SOLUTIONS. https://www.researchgate.net/profile/Premkumar-Ganesan-2/publication/384867297_OBSERVABILITY_IN_CLOUD-NATIVE_ENVIRONMENTS_CHALLENGES_AND_SOLUTIONS/links/670aeedbe3c0600c7ca09ed4/OBSERVABILITY-IN-CLOUD-NATIVE-ENVIRONMENTS-CHALLENGES-AND-SOLUTIONS.pdf

[7] Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T., & Miranskyy, A. (2023). A reference architecture for observability and compliance of cloud native applications. arXiv preprint arXiv:2302.11617. https://doi.org/10.48550/arXiv.2302.11617

[8] James, E. (2023). OBSERVABILITY AND DEBUGGING IN SERVERLESS DATA WORKFLOWS. https://www.researchgate.net/profile/Ebenezer-James/publication/390769213_OBSERVABILITY_AND_DEBUGGING_IN_SERVERLESS_DATA_WORKFLOWS/links/67fdaf3ddf0e3f544f4199e7/OBSERVABILITY-AND-DEBUGGING-IN-SERVERLESS-DATA-WORKFLOWS.pdf

[9] Soldani, D., Nahi, P., Bour, H., Jafarizadeh, S., Soliman, M. F., Di Giovanna, L., ... & Risso, F. (2023). ebpf: A new approach to cloud-native observability, networking and security for current (5g) and future mobile networks (6g and beyond). IEEE Access, 11, 57174-57202. 10.1109/ACCESS.2023.3281480

[10] Kettunen, J. P. (2024). Maintainability in cloud-native architecture. URN:NBN:fi:jyu-202406054265.pdf

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details