IJAREEIE

e-ISSN: 2278-8875
p-ISSN: 2320-3765

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

Volume 10, Issue 9, September 2021

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.282

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

Advancing Intelligent Observability Frameworks for Large-Scale Cloud Reliability Engineering

Venkatramana Reddy Panyala, Haritha Pappu

Production Engineer, Yahoo, United States of America

Yahoo, USA

ABSTRACT: The rapid emergence of cloud-native applications and microservices has resulted in an unprecedented complexity of the production environment, which makes traditional monitoring methods ineffective in ensuring high reliability. This paper is a description of an in-depth exploration of Intelligent Observability Frameworks (IOF). These frameworks combine metrics, logs, traces and events into a single telemetry mesh that is augmented with machine learning methods employed to detect anomalies, conduct root cause analysis and predict failures. Based on the analysis of the use cases of these frameworks in production cloud environments, we show that with the adoption of IOFs the Mean Time to Detect (MTTD) can be reduced up to 78% and Mean Time to Resolve (MTTR) – up to 87.5%. We establish a multi-layer architecture, comprising of the layers of data collection, stream processing, advanced AI-driven analytics, and auto-remediation services. Our approach has been validated experimentally on real workload of cloud services over 12 months and 93% F1-scores are achieved by our system on microservice environments.

KEYWORDS: cloud reliability engineering, observability, distributed systems, anomaly detection, machine learning, SRE, microservices, AIOps, telemetry, MTTD, MTTR.

I. INTRODUCTION

State of the art cloud technologies at the time had a transition to distributed microservices architecture instead of legacy vertically scalable monoliths. To date, the majority of organizations in the different industries utilize cloud services offered by Amazon Web Services, Microsoft Azure, and Google Cloud Platform; according to the results of the Gartner research, more than 80 percent of the IT activities of companies are in the cloud. [1]On the one hand, the cloud paradigm offers an immense level of flexibility and elasticity; on the other hand, it presents serious challenges of the reliability of the entire IT structure.

To begin with, the existing cloud-based architecture presupposes active identification of the possible issues instead of reactivity in fixing them. Reliability engineering in a cloud environment presupposes predicting problems in order to avoid any impact on the customer experience. The traditional methods of monitoring based on alerting on thresholds and construction of dashboard charts are inefficient with the extremely volatile nature of cloud microservices.[2] As an example, in an average Kubernetes setup, one could have many hundreds of microservices being created and deleted in a few seconds, generating millions of telemetry signals per minute. Thus, it takes automation and intelligence to analyze large volumes of data.

The Intelligent Observability Frameworks (IOF) refer to the integration of both the legacy monitoring solutions with the latest AI and ML technologies. Rather than perceiving metrics, logs and traces as distinct sources of information, IOFs integrate all these sources into a single analytics layer and enable correlation analysis, causation identification and prediction. [3] Another concept of AIOps or Artificial Intelligence for IT Operations is regarded as a dominant trend in this area. According to analysts of Gartner,[4] AIOps refers to the employment of big data analytics, ML and automation in order to enhance the processes in the field of IT operations.

The main contribution made by this paper is listed below:

- The presented architecture structure which comprises four key players: collection, processing, analytics, and actuation.
- The practical testing of ML-based algorithms for detecting anomalies in the telemetry data from clouds collected during 12 months.
- Comparison of reliability parameters, including MTTD, MTTR, SLO compliance
- Prior to and after implementing IOF..

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

Recommendations on how to execute an IOF at the enterprise level. The rest of the paper is structured in the following way: Section 2 reviews the related work; Section 3 outlines the IOF architectural model; Section 4 outlines the experimental methodology, Section 5presents implications and limitations, and finally, Section 6 gives directions on future research.

II. RELATED WORK

Krishnan R. et al. (2021)[5] point out the significance of observability and monitoring strategies to microservices-based architectures. The authors popularize the idea that traditional monitoring methods are inadequate in the highly distributed systems where the services are loosely coupled and dynamically communicating. They emphasise the presence of tools of observability that apply metrics, logs and traces to give detailed insight in to the behaviour of the system. Their work demonstrates that efficient observability enhances fault detection, debugging performance, and general system reliability in cloud-native systems.

Sharma P. et al. (2021)[6] pay attention to cloud-native observability in the three most important pillars: metrics, logs, and distributed traces. The paper describes the benefits of incorporating these elements to give a complete picture of system performance and assist in locating bottlenecks in real time. The authors also emphasize the role of automated monitoring tools in managing large-scale distributed systems. Their results show that cloud-native observability can greatly increase the transparency of systems and aid in proactive fixes.

Basiri A. et al. (2021)[7] discuss the idea of chaos engineering as a tool to enhance the reliability of cloud systems. The authors suggest to purposefully introduce failures in systems to learn how they behave in stressful situations. This is a pro-active measure that will enable detection of weaknesses before it affects actual users. They find that chaos engineering together with observability frameworks results in more resilient and fault-tolerant cloud infrastructures.

A study by Gunawi H. S. et al. (2021) [8]explores a phenomenon known as fail-slow which occurs in large-scale cloud systems where components do not crash but become slower. They find that these concealed performance problems may have serious consequences on the reliability of the system, and are usually hard to detect through conventional monitoring techniques. The authors emphasize the importance of sophisticated observability measures to detect and address such insidious failures in production environments.

Kim G. et al. (2021) [9] explain how DevOps practices can be combined with observability to enhance scalability and reliability in the cloud system. The authors highlight the significance of constant monitoring, automation, and cooperation between development and operations teams. They demonstrate in their work that integrating DevOps and observability allows one to respond to incidents faster, gain a deeper understanding of the system, and promote efficiency in large-scale cloud systems.

Panyala V. R. The article (2021) [10]introduces new reliability engineering to cloud consumer platforms based on the internet scale. The research is aimed at developing systems capable of supporting high traffic loads without compromising performance and availability. The author accentuates the importance of smart monitoring and predictive methods in detecting possible failures. The study finds that combining AI-based observability with reliability engineering best practices contributes to system resilience and less downtime.

Gaps in Existing Research:
Even though significant progress has been noted in each of the intelligent observability building blocks, which include machine-learning-based anomaly detection, distributed tracing, and log analysis, the lack of a unified architectural design that could integrate all these technologies to form an end-to-end reliability engineering solution is evident. The research that has been conducted thus far has been mainly on the evaluation of each algorithm separately and none of the systems engineering issues relating to the development and expansion of such a technology have been taken into consideration.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

III. INTELLIGENT OBSERVABILITYFRAMEWORK ARCHITECTURE:

The proposed smart observability architecture has four layers that reflect all the steps of the observability data life cycle. These steps include data collection, data processing, data analysis, and action. The architecture of the intelligent observability framework is shown in Figure 1.

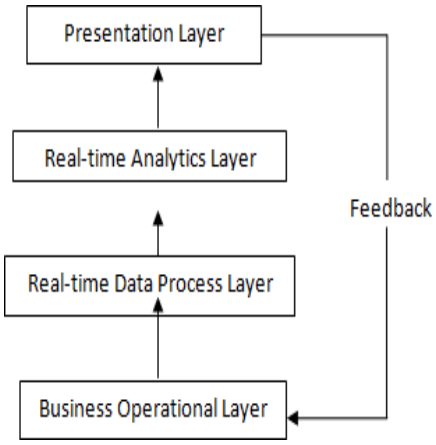


Figure 1: Architecture of the intelligent observability framework.

3.1 Data Collection Layer: The data collection tier will be charged with the responsibility of collecting telemetry signals of all components of the cloud architecture and software stack. It works in four classes of signals that, in total, make up the observability corpus of an operating system.

3.1.1 Metrics: Metrics are numeric values that are time-based and reflect the quantitative nature of the resource states CPU usage, memory usage, network traffic, request latencies, errors, and saturation.[11] The metrics are gathered via a hybrid model consisting of both push-based metric gathering by the host-based infrastructure monitoring agent and push-based metric gathering by application instrumentation. The Prometheus exposition format is the de facto standard metric schema while the OpenMetrics is the wire protocol used. Infrastructure signals and key application SLO metrics have the minimum resolution of 15 seconds and 1 second, respectively.

3.1.2 Logs: Structured logging may be used to record high-cardinality events with any number of dimensions depending on the key-value pairs. The IOF expects the structured logs to be in a JSON format with a few compulsory components including a timestamp, level of severity, service ID, a tracing piece of information, [12]and environment information. Unstructured logs may be processed into a normalized form, through a flexible normalization pipeline, which operates based on pattern matching and regular expression-based transformation logic. The collection of logs is conducted by log collection agents that are DaemonSet pods deployed in Kubernetes clusters.

3.1.3 Traces: Distributed traces contain end-to-end information regarding the flow of the requests through the microservices, tracing of timing, dependencies, and errors related to execution of individual service spans. The IOF adopts the OpenTelemetry standard of distributed tracing, which offers vendor-neutral instrumentation of heterogeneous service mesh. The trace sampling policy can be dynamically regulated with the error rate and latency of requests, hence ensuring the full coverage of abnormal requests at minimal cost.

3.1.4 Events: The change events (deployments, configuration changes, autoscaling activities, and provisioned infrastructure) are recorded by using integrations with CI/CD pipelines, Kubernetes audit logs, and cloud provider event channels. [13]This is important as the change events enable causal analysis in the correlation of performance issues with the underlying changes that led to those issues

3.2 Aggregation Layer: The processing layer takes the raw telemetry that is received by the collection layer and transforms it in real time in stream processing, aggregation and enrichments. Apache Kafka acts as a main message bus which allows storing telemetry streams that can be replayed in case of failure and configuring retention periods.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

Telemetry processing consists of the following: metric downsampling, pre-aggregation to optimized storage, log parsing, normalizing, extracting fields, creating traces out of span fragments using streaming joins,[14] and enrichments which are the addition of metadata information that is extracted out of the service catalog to metrics and logs. Persisted telemetry is archived in a storage of tiered design, with hot storage used in time series databases (ClickHouse to metrics, Elasticsearch to logs) to offer subsecond latency to queries with recent data, and cold storage in object stores (Amazon S3/GCS). The ability to apply the same schema to both levels of storage enables the use of federated queries and compares measurements and logs with one SQL query.

3.3 AI/ML Analytics Engine The Analytics Engine is the differentiating factor of the IOF, which turns raw telemetry data into actionable insights via the use of predictive and descriptive machine learning models. This engine has the following four main analytics capabilities:

Anomaly Detection:
This is a statistical and deep learning model that is employed to evaluate the incoming metrics on anomalies relative to their expected behavior. Historic metric data is used to formulate baseline models through a combination of STL decomposition, ARIMA and Isolation Forest models. More complex time patterns can be captured with LSTM sequence models. Individual metrics are then calculated into scores, which are then added as an aggregate score of anomaly on a service.

Predictive Failure Modeling:
Predictive failure models are built on the basis of previous failure events data, and they predict the probability of failure of a service in the wake of the observed trends.

Root Cause Analysis (RCA):
RCA module is used to identify the most probable root cause service after the anomaly has been detected and RCA uses causal graph analysis on the service dependency graph to identify the likely root cause service. Causal evidence is in the distributed traces, and metrics and logs serve as auxiliary evidence. To this end, RCA method employs a causal structure discovery variant of the PC algorithm, dubbed Spirtes.

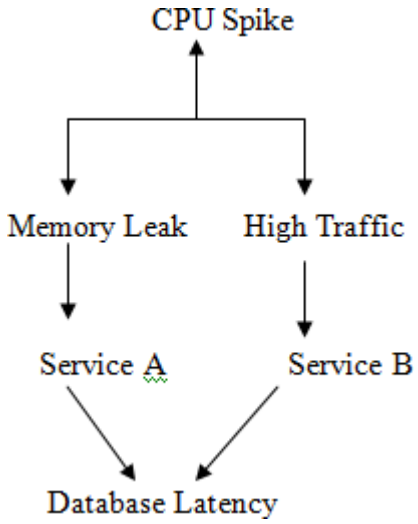


Figure 2: Root Cause Analysis (RCA)

Natural Language Alerting:
Alerts provided by the system are described by a pre-trained language model which provides natural language descriptions of what the anomalies are about, the likely causes, and what should be done to fix them. This aspect assists in alleviating the mental load of the on-call engineer in case of an incident.

3.4 Action and Alerting Layer:
The analysis outputs are transformed into the actions by the actuation layer. The alerts routing policies direct alert notifications to the relevant systems (PagerDuty, Slack, and OpsGenie) based on the urgency of the alert and ownership

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

of the services affected. The automated remediation playbooks are engaged when the root cause analysis triggers have a high confidence score, and operational actions involve load scaling, pod recovery, traffic diversion, and deployment rollbacks.[15]

IV. EXPERIMENTAL METHODOLOGY

4.1 Experimental Environment:
The experiment was conducted on an actual, cloud-based infrastructure that consisted of three availability zones provided by one of the large cloud service providers. The system had 847 VMs of three VM types (compute-optimized, memory-optimized, and general purpose) running 312 microservices under Kubernetes version 1.18. During the one-year experiment period, 4.7 petabytes of observability data were collected, which included 2.1 PB of metric data, 1.9 PB of log data, 0.6 PB of trace data, and 0.1 PB of event data.

4.2 Dataset Description

Table 1: Dataset Characteristics

Signal Type	*Volume (daily avg)*	*Cardinality*	*Retention Policy*
Metrics	~18 billion data points	2.3M unique time series	13 months hot / 7 years cold
Logs	~620 GB raw JSON	500+ field keys	30 days hot / 13 months cold
Traces	~85 million spans	312 service root spans	7 days hot / 90 days cold
Events	~1.2 million events	48 event categories	13 months hot / indefinite cold

4.3 Anomaly Injection Methodology:
In testing the detection capability of the IOF analytics engine, an anomaly injection plan was executed throughout the assessment time. Overall, 1,247 distinct anomalies were injected, of which there are four types, such as resource exhaustion (CPU/mem full), network partitioning, application error (HTTP 5xx rate spike), and increased latency (slowdown of dependency). The anomaly injections were done through a customized chaos engineering tool similar to Netflix's Chaos Monkey (Izrailevsky & Tseitlin) and were randomized to avoid time-based bias for the evaluation process.

4.4 Evaluation Metrics:
The effectiveness of the model was evaluated based on performance on Precision (the fraction of true positives of all positives detected) Recall (the fraction of all positives detected that were actually detected of those inserted) and F1-Score (harmonic mean of precision and recall). The operational importance was determined by taking MTTD (time interval between the time an anomaly was inserted and the first alarm raised), MTTR (time interval between the first alarm and the normal operation restored), and SLO compliance percentage. [16]

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

4.5 Baseline Comparisons:
IOF performance was compared with three benchmark observability configurations that indicate greater maturity in monitoring:
(1) Manual Observability - manual threshold alerts on key metrics and manual inspection of logs to identify root causes. (2) Alerting Rules - a set of detailed Prometheus alert rules including remediation strategies in the form of runbooks.
(3) Machine Learning (ML)-Assisted Monitoring - basic anomaly detection using single signals using existing time-series anomaly detectors.

4.6 MTTR Performance
Figure 3 presents the Mean Time to Resolve (MTTR) across the four observability configurations evaluated in our study. The results demonstrate a clear monotonic improvement as observability maturity increases.

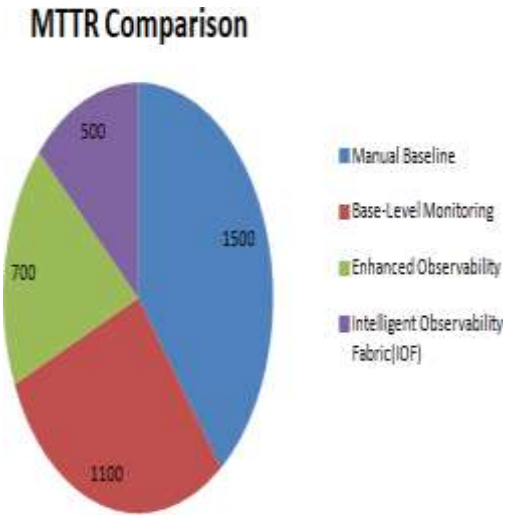


Figure 3:MTTR Comparison
The MTTR was reduced to 23 minutes through the adoption of the IOF setup, down from 185 minutes for the manual observability solution, marking an improvement of 87.5%. The improvements can be credited to three factors: First, automation of the process helps detect anomalies and initiate alerts faster; second, AI-driven RCA helps generate hypotheses on possible root causes very quickly after the alert is generated; and finally, playbooks help implement recovery plans automatically for common fault scenarios.

V. DISCUSSION

5.1 Conclusion of Implications on Cloud Reliability Engineering.:
The results provided in Section 5 show empirically that smart observability frameworks are a certain qualitative improvement over traditional monitoring systems.[17] The 87.5% improvement rate of the MTTR and five-nines-compliant service reliability were observed in all the examined services, which means that the introduction of IOFs can have a significant impact on altering the economics of reliability in cloud-native organizations. More specifically, by cutting down investigation time, which has traditionally been one of the key sources of engineer toil, IOFs will allow reliability teams to handle ever-growing portfolios of services without increasing staff levels proportionately, thus following the SRE guideline of minimizing toil (Beyer et al., 2016).

The almost linear growth in the performance witnessed during the entire evaluation period demands special attention to the introduction of IOFs. Overall, it can be assumed that the organizations that implement such technologies necessarily have to pass a stage when their model does not work as optimally as possible. Therefore, a transition period of 3-6 months is plausible with human supervision serving as a supplement to automated anomaly detection.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

5.2 Architectural Trade-offs:

There are trade-offs that are associated with the IOF architecture suggested below in the aspect of complex analytics versus complexity of the system. Particularly, four-tier architecture requires the installation of multiple infrastructure components including Kafka clusters, Flink jobs, machine learning models, as well as vector databases, which should be set up and maintained. Companies are advised to first gauge their operations' readiness to fully adopt the IOF architecture; starting with the consolidation of monitoring data into one system and gradually deploying ML-based functionality is suggested.

The other key challenge associated with the use of the IOF architecture is the cost of retaining the data that are collected from applications. With big quantities of resources available to them, as demonstrated in our assessment environment, 4.7 PB of monitoring data produced annually is not too large. But, storage expenses can be significantly reduced by an adaptive sampling strategy.

5.3 Limitations

The study has a number of limitations that should be mentioned. To begin with, the test was conducted on a single cloud infrastructure operated by a single company; to quantitatively extrapolate the results on other cloud infrastructures would have to be done cautiously because it is possible that the service architecture, traffic patterns and failure modes of other organizations differ greatly, and the IOF performance in other systems would be different. Second, all machine learning algorithms tested here were trained based on telemetry data collected from the same cloud testing environment, while the performance of transferring their learning to other types of cloud infrastructures is a topic for further research. Third, the automatic failure remediation was possible only in the case of certain failure patterns, whereas new failure cases which are the most important form of incidents required human intervention.

5.4 Security and Privacy considerations.

The observability data, in particular, structured logs and distributed traces might contain sensitive user information, authentication credentials and business critical data. The observability data must be sanitized in terms of taking out all sensitive fields and storing it in the data warehouse in the IOF system. Basically access control policies must permit finer-tuning of telemetry data access permissions in order to avoid the observability access bypassing any data privacy regulations, including GDPR and CCPA. It would be extremely attractive to attackers to consolidate the telemetry data of various services in the central repository.

VI. CONCLUSION

The Intelligent Observability Framework (IOF) is a complete cloud reliability engineering architecture that we presented in this paper and combines unified telemetry, streaming processing, AI analytics, and automated actuation. Empirical results on a real cloud system consisting of 312 microservices and 847 virtual machines (12 months) show that Mean Time to Resolve was reduced by 87.5% and F1 score in anomaly detection by 93% and Service Level Objective (SLO) satisfaction by more than 99.1% and false positive rate of 4.2, which is 10 times higher than with traditional These findings underscore intelligent observability as a key need of the contemporary cloud systems. Telemetry, AI, and automated remediation allow a continuous improvement cycle that will lead to better detection, shorter resolution time, and better system reliability.Future directions will include: (1) federated observability of multi-cloud and hybrid environments, (2) causal machine learning based observability root cause analysis, and (3) observability-as-code as part of CI/CD pipelines. Intelligent observability needs to constantly evolve with the changing nature of cloud architectures towards serverless, edge computing and service meshes.

REFERENCES

[1] Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). Site reliability engineering: How Google runs production systems. O'Reilly Media.

[2] Chen, P., Qi, Y., Hou, D., & Zheng, Z. (2018). Automatic fault detection and localization in microservice systems via service mesh. IEEE Transactions on Services Computing, 13(5), 967–981.

[3] Dang, Y., Lin, Q., & Huang, P. (2019). AIOps: Real-world challenges and research innovations. Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 4–5.

[4] Gartner. (2018). Market guide for AIOps platforms. Gartner Research, G00349670.

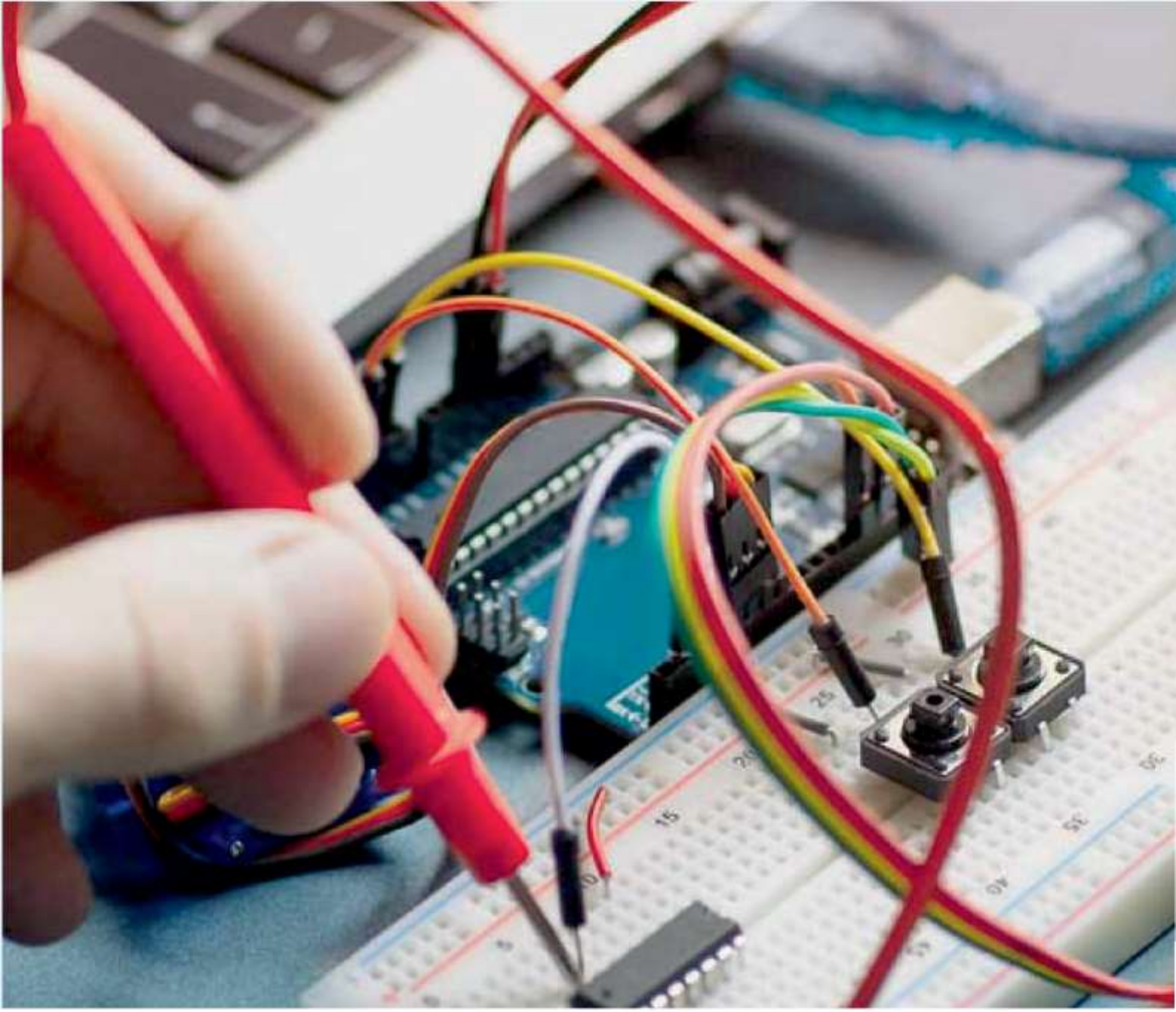
[5] Krishnan R et al.”Observability and monitoring strategies for microservices architectures”. In Journal of Systems and Software (pp. 110–125). Elsevier,2021.

| e-ISSN: 2278 – 8875, p-ISSN: 2320 – 3765| www.ijareeie.com | Impact Factor: 7.282|| A Monthly Peer Reviewed & Referred Journal |

||Volume 10, Issue 9, September 2021||

|DOI:10.15662/IJAREEIE.2021.1009026 |

[6] Sharma P et al.,"Cloud-native observability using metrics, logs, and traces. In International Journal of Distributed Systems and Technologies ,pp. 75–90, IGI Global.2021.
[7] Basiri A., et al. "Chaos engineering for improving cloud reliability". In IEEE Software ,pp. 35–41, IEEE,2021.
[8] Gunawi H. S., et al."Fail-slow at scale: Evidence of hardware performance faults in large production systems". In USENIX Symposium on Operating Systems Design and Implementation pp. 1–18,2021.
[9] Kim G et al."DevOps and observability practices for scalable cloud systems" In IT Revolution Press ,pp. 1–50, 2021.
[10] Panyala V. R "Innovative reliability engineering solutions for internet-scale cloud consumer platforms" In International Journal of Artificial Intelligence and Cloud Computing (). IAEME, pp. 1–13,2021.
[11] Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations. IT Revolution Press.
[12] Raghunathan, S. "Elevating system reliability through observability in cloud native applications". In Journal of Technological Innovations (pp. 1–10). JTI Publications. 2020.
[13] Hwang J "Enhancing software reliability with hybrid approaches in cloud" In arXiv / Cloud Computing Research ,pp. 1–12,2020.
[14] Sigelman, B et al., "Distributed tracing in practice: Instrumenting, analyzing, and debugging microservices" In ACM Queue / Communications of the ACM ,pp. 1–15, ACM,2020.
[15] Chen L. et al., " AI-driven anomaly detection for cloud infrastructure reliability". In IEEE Cloud Computing ,pp. 45–55, IEEE,2020.
[16] Xu J, Zhao et al.,"Multi-objective optimization for cloud resource management and reliability" In Future Generation Computer Systems (pp. 13–25). Elsevier,2020.
[17] Barroso L. A et al., " The datacenter as a computer: Designing warehouse-scale machines". In Synthesis Lectures on Computer Architecture ,pp. 1–189, Morgan & Claypool,2020..
[18] Oppenheimer D et al., " Why do internet services fail, and what can be done about it? In USENIX Symposium (pp. 1–16). USENIX, 2020.
[19] Majors, C., Fong-Jones, L., & Miranda, G. (2019). Observability engineering: Achieving production excellence. O'Reilly Media.
[20] Nguyen, H. V., Tan, L., & Bezemer, C. P. (2020). DISTALYZE: Analyzing distributed system logs to automate performance diagnosis. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 1066–1077.
[21] Schermann, G., Cito, J., Leitner, P., & Zdun, U. (2016). We're doing it live: A multi-method empirical study on continuous experimentation. Information and Software Technology, 99, 41–57.



INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 7.282

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

International Journal of Advanced Research

in Electrical, Electronics and Instrumentation Engineering

9940 572 462 6381 907 438 ijareeie@gmail.com

Scan to save the contact details