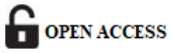
International Journal of Artificial Intelligence Research and Development (IJAIRD)
Volume 1, Issue 4, January-December 2022, pp. 13-27, Article ID: IJAIRD_01_04_002
Available online at https://iaeme.com/Home/issue/IJAIRD?Volume=1&Issue=4
Impact Factor (2022): 11.98 (Based on Google Scholar Citation)
Journal ID: 234A-56Z1; DOI: https://doi.org/10.34218/IJAIRD_01_04_002



AI-POWERED OPERATIONAL INTELLIGENCE FOR MANAGING HIGH-SCALE CLOUD-NATIVE DISTRIBUTED SYSTEMS

Venkatramana Reddy Panyala
Production Engineer, Yahoo, USA.

ABSTRACT

The explosion in cloud-native architectures and microservice-based distributed systems has brought about a level of complexity that cannot be effectively handled using traditional monitoring systems and manual incident handling strategies. This paper introduces a novel operational intelligence framework, based on the artificial intelligence, streaming, and auto-remediation facilities, to manage the high-scale cloud-native distributed systems. It comprises anomaly detection, predictive analysis, smart alerting, and auto-remediation. By using the data produced by containers, service mesh technologies, and cloud infrastructure, the system can deliver operational intelligence by employing machine learning, stream processing, and autonomous actions, and minimal human interaction is necessary. The proposed system architecture is vendor-neutral and extensible, which allows it to be compatible with popular observability products on the market. The paper addresses the system architecture, data flows, AI/ML model development, and deployment issues, putting a particular focus on how operational intelligence is applied in DevOps and SRE.

Keywords: Cloud-native systems, operational intelligence, anomaly detection, distributed systems, machine learning, observability, auto-remediation, microservices, AIOps, DevOps

Cite this Article: Venkatramana Reddy Panyala. (2022). AI-Powered Operational Intelligence for Managing High-Scale Cloud-Native Distributed Systems. *International Journal of Artificial Intelligence Research and Development (IJAIRD)*, 1(4), 13–27.
DOI: https://doi.org/10.34218/IJAIRD_01_04_002

1. Introduction

The advent of cloud computing models has brought about a paradigm shift in the way companies develop, provide and execute their software systems. Containers, microservices, dynamic orchestration, and declarative APIs are the characteristics of cloud-based systems that enable companies to achieve new scalability, resilience, and deployment agility. [1] Technologies like Kubernetes, Istio, and serverless platforms have become integral elements in modern software development lifecycles, supporting continuous delivery in a distributed computing environment.

However, this change has resulted in complicated issues. A cloud system may include thousands of microservices that constantly emit logs, metrics and distributed tracing data. The amount and speed of telemetry data produced by these systems are beyond the human brain's ability to analyze them in real time. Such events like latencies, resource exhaustion, cascade failures, and service degradation can happen within milliseconds.[2]

These classic monitoring platforms have several significant drawbacks, which make them not applicable in this case: they have a huge number of false positives, they do not comprehend context, and impose a significant cognitive burden on an operations staff member. SREs and platform engineers have come to be overwhelmed by the complex nature of their work environments, suffering from alert fatigue, slow incident response, and high Mean Time to Recover.

Interestingly enough, AIOps (Artificial Intelligence in IT Operations) is a term coined by Gartner in 2017 and offers an interesting way of addressing these issues. An AIOps system combines the use of Big Data analytics, machine learning and automation technologies to supplement and automate some operations management functions such as event correlation, root cause analysis and remediation activities.[3] The entire process of an autonomous AIOps platform is able to handle multiple streams of telemetry in real-time, identify anomalies, anticipate possible failures, and trigger remedial measures without human intervention.

The paper is an elaborated architecture of operational intelligence based on the use of artificial intelligence that is specific to the needs of large scale cloud native distributed systems.

This involves a set of real-time stream processing, unsupervised and supervised machine learning algorithms, intelligent event correlation and automatic remediation pipelines. [4]A summary of the suggested architecture focusing on the data ingestion, feature extraction, model inference, alerting, and feedback loops is presented. Finally, important aspects of the design process such as scalability, fault tolerance, interpretability, and toolchain integration are considered.

The remainder of this paper is organized in the following way: In Section 2, the literature that is relevant to the area of AIOps, observability of distributed systems, and anomaly detection are discussed. Section 3 presents the general framework of the proposed system and elaborates on it. Sections 4 and 5 discuss methods of AI and machine learning that were used. In Section 5, issues relating to implementation and deployment are considered.

2. Related Work

Distributed system intelligent operations management has been an area of intense research interest in the past few years. The next section discusses some of the historical work, which is relevant to the present work in the overlapping fields of observability, anomaly detection, and incident management.

Observability as the property of inferring the internal state of a system, given observations of external outputs, has evolved into a significant research area to design complex distributed systems. There are typically three big components of the concept of observability: metrics, logs, and distributed traces. Sigelman et al. (2010) [5]introduced the first trace-based system, Dapper, at Google; tracing systems such as OpenTelemetry and Jaeger are based on it. In their follow-up work, Mace & Fonseca (2018) explained the idea of end-to-end distributed tracing and discussed issues related to propagating context asynchronously between services[6]

Ma et al. (2020) developed a holistic observability of container-based microservice architectures and discovered that a combination of metrics, logs, and traces can provide a lot more valuable information than a single source of data. The results inspired the creation of many observability tools like Datadog, New Relic, and Honeycomb.

Anomaly detection on a time series data has been given considerable attention in the field of it in IT operations. Classical techniques such as Seasonal Decomposition of Time Series by Loess and ARIMA have been used to analyze system metrics (Hyndman and Khandakar, 2008[7]. More recently, deep learning-based methods have been demonstrated to be superior to traditional approaches based on operational data.

Hundman et al. (2018)[8] have discussed the option of LSTM-based anomalies in spacecraft telemetry demonstrating that RNNs can be used to model the dynamics of multivariate time series. Audibert et al. (2020)[1] proposed USAD (UnSupervised Anomaly Detection), an autoencoder-based method of anomaly detection in time series, which were tested on various industrial data sets. Another variational autoencoder-based anomaly detector method, DONUT, was developed by Xu et al.,[9] however, the method was focused on KPI anomaly detection on the internet services

AIOps has been implemented by vendors as well as academic researchers. Lim et al. (2014) [10]demonstrated some of the first attempts to predict failure in machine learning classifiers with the aim of demonstrating that natural language processing of log messages can predict system failures minutes before they occur. Chen et al. (2019)[11] proposed an alert management system in clouds that is based on correlations to minimize the level of noise in alerts by up to 90 percent compared to threshold-based systems.

Some research teams have looked into automated remediation of issues within cloud infrastructure. Benomar et al. (2021) [12]proposed an autonomy control loop on Kubernetes-based systems which uses reinforcement learning to establish optimal remediation actions given the current states of the system. The first research on the application of logs in detecting and automatically fixing issues in systems was carried out by Syer et al. (2014) [13]establishing some general principles of automating operations. Most innovations have been made in other fields of operation management individually but there are not many solutions that offer a unified solution to deliver all components of operational intelligence simultaneously, as well as data ingestion, multi-modal anomaly detection, intelligent correlation, and automated remediation, in a single solution. The gap is the one that is tried to be bridged in this paper.

3. System Architecture:

The proposed operational intelligence system based on the artificial intelligence is designed as an event-driven multi-level architecture, which analyzes telemetry data in the cloud-native environment in real-time.[14] The architecture is comprised of five key tiers such as data intake, stream processing, AI/ML analytics, operational intelligence, and observability/dashboarding. Figure 1 represents the system’s architecture.

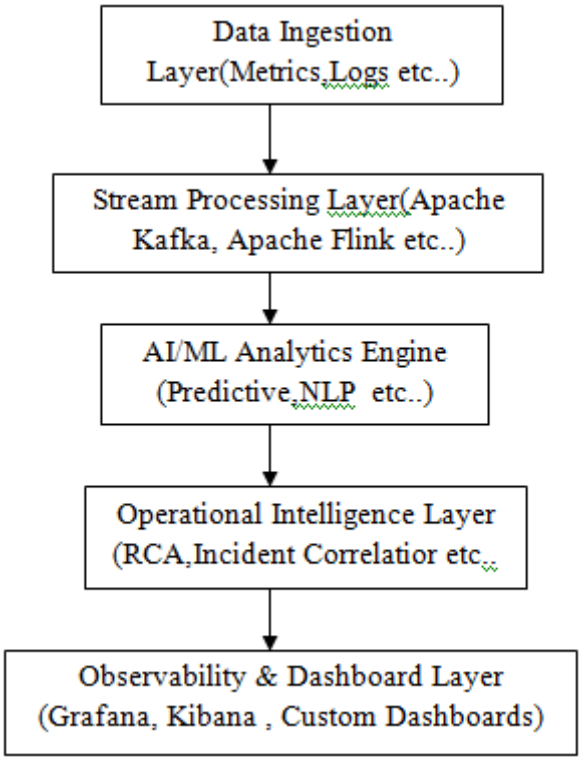


Figure 1: AI-Powered Operational Intelligence Architecture

3.1 Data Ingestion Layer :

The data ingestion layer will be charged with the responsibility of gathering telemetry data in all aspects of the cloud-native ecosystem. It accomplishes this by retrieving telemetry information across a wide range of sources, such as infrastructure level telemetry (CPU, RAM, network, disk I/O) of Kubernetes nodes and pods, application level telemetry (Prometheuses or StatsD interfaces) and structured and unstructured logs of container runtimes and application code, distributed traces of instrumented microservices and synthetic telemetry of external probing agents. [15]The ingestion layer relies on a pulling approach to gathering telemetry data and a pushing approach to dealing with monitoring tools log streams and events. Any data ingested is converted to a single telemetry format, and this format includes uniform field names, milliseconds time stamps, service specific IDs, and environment data. This conversion is essential to enable correlation among various signals of telemetry in downstream processing pipelines.

3.2 Stream Processing Layer:

Normalized telemetry events are issued into a distributed message broker that is typically referred to as Apache Kafka which serves as the central element of the streaming pipeline architecture. Kafka provides a trustworthy storage of ordered and partitioned events with a

loose-coupling between data publishers and subscribers that enables scalability to millions of events per second. Apache Flink is used to perform the stream processing, and offers exactly-once stream processing guarantees, stateful stream processing, and low-latency windowed aggregation operations. This layer calculates sliding window aggregations of statistical baselines, joins telemetry events based on time proximity, filters and routes them to appropriate downstream consumers, and adds contextual information to events by a service registry.

3.3 AI/ML Analytics Engine:

The analytics engine forms the central intelligence for the architecture. It consists of a number of various model pipelines that receive pre-processed telemetry streams. These are the anomaly detection pipeline, which relies on statistical and learned models to detect anomalies, the predictive analytics pipeline, which makes future predictions about resource utilization and failures, the log analysis pipeline, which uses natural language processing to extract structured events of log messages and the event correlation engine, which clusters alerts and anomalies to create incidents. The deployment of models is through a model serving mechanism, which makes quick inferences. The model version and the corresponding performance of the model versions are kept in the model registry of this engine.[16] This allows the model to be easily retrained using new data and deployed using shadow deployments and A/B testing.

3.4 Operational Intelligence Layer:

The Operational Intelligence Layer transforms AI-driven insights into operational actions. The major functions of this layer are root cause analysis where the possible cause of a service dependency graph incident is found; intelligent alert routing whereby the appropriate on-call or automated handling team is notified about an incident based on its priority and domain classification; and lastly, automated remediation whereby the incident is resolved using playbooks or policies without human intervention. [17]The remediation is implemented in control loop, where it monitors system state, makes a decision based on remediation, implements decision through the Kubernetes or cloud provider API and verifies the effect. Remediation tasks which are risky or new have an approval step.

3.5 Observability and Dashboard Layer:

The observability layer provides the user interface to the system, allowing the user to view the system state, incident history, AI model explanations, and system health by using dashboard interfaces. Libraries such as Grafana and Kibana are utilized and custom panels that show the information produced by the AI, such as anomaly scores and audit logs of remediations. The observability layer is RESTful and can be used as a webhook to integrate with incident management platforms such as PagerDuty, Opsgenie, and ServiceNow.

4. AI and Machines Learning procedures.

This section explains the particular AI and machine learning methods that are used in the proposed framework. The methodology aims to compromise between detection accuracy, computational efficiency, and explainability of operations.

4.1 Anomaly Detection

The system has an anomaly detection mechanism which operates on three types of signals, i.e. time series metrics, logs, and traces. Another form of hierarchical anomaly detection is applied in the system in the case of time series metrics. [18]An isolation forest algorithm is used in univariate anomaly detection to identify global outliers in single streams of metrics. Multivariate anomaly detection uses a Variational Autoencoder (VAE) which is trained on past windows of metrics and produces reconstruction errors to be used as continuous anomalies. VAE is very efficient in detecting non-linear dependencies among related metrics, such as the dependency of request latency, error rate, and CPU usage Figure 2 shows the process flow in anomaly detection.

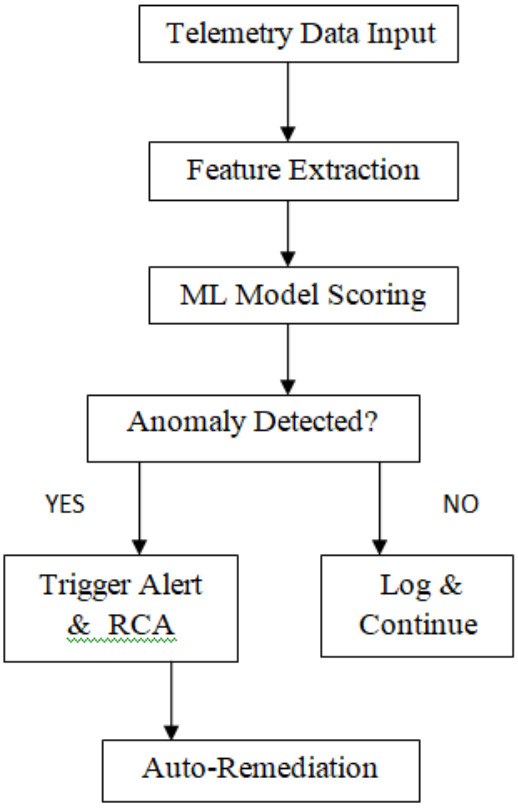


Figure 2: Anomaly Detection and Alert Workflow

4.1 Anomaly Detection

The system has an anomaly detection mechanism which operates on three types of signals, i.e. time series metrics, logs, and traces. Another form of hierarchical anomaly detection

is applied in the system in the case of time series metrics. An isolation forest algorithm is used in univariate anomaly detection to identify global outliers in single streams of metrics.[19] Multivariate anomaly detection uses a Variational Autoencoder (VAE) which is trained on past windows of metrics and produces reconstruction errors to be used as continuous anomalies. VAE is very efficient in detecting non-linear dependencies among related metrics, such as the dependency of request latency, error rate, and CPU usage

The seasonal decomposition via STL (seasonal and trend decomposition using loess) technique is used during pre-processing in order to filter out seasonal patterns in time series of metrics prior to calculating their anomaly scores. That will make sure that any models employed do not give a false-positive alert because of normal diurnal/weekly traffic patterns. An exponentially weighted average is used to maintain baselines dynamic.

4.2.1 Log Analysis and Event Parsing.

The issue with the analysis of logs is complicated in nature due to their unstructured nature and quantity. The framework uses a log parsing pipeline, which realizes Drain (He et al., 2017), a tree-based online log parsing technique to extract log templates in real times by just reading through the raw log messages without any information about the log format. This enables the framework to be used on logs of heterogeneous systems using different logging tools.

The system uses a multi-label classification model to classify log events through a set of annotated log events. In particular, this model categorizes errors, matches events to failure scenarios, and calculates their frequency anomalies in a sliding window manner. Additionally, the semantic similarity of log events to cluster events into the same logical failures even when they are worded differently is computed with the help of a language model built on BERT and trained on the operational logs.

4.3 Predictive analytics/capacity forecasting.

The ability to perform predictive analysis enables the system to be capable of predicting issues beforehand. The predictive system involves Facebook Prophet model, which forecasts future trends of resource metrics based on predictions of univariate time-series based on Facebook. This enables the capacity planning to be planned at varying degrees of prediction of 15 minutes to 7 days. Facebook Prophet model employs a model structure through additive model that can cope with multiple seasonalities, and missing/outlier points[20].

The issue of forecasting multiple steps in advance of intricate metric designs can be addressed with the aid of Temporal Convolutional Network (TCN). The model is efficient in modeling long term temporal dependencies than other models like LSTM. TCN model is trained

on a sliding window of historical metric values, and re-trained on online mode with current metric values.

4.4 Event Correlation and Incident Grouping.

Identification of related anomalies which can be combined into incidences is one of the most crucial functions of the analytics engine. The event correlation module holds a graph of all service dependencies based on distributed tracing data and determines any clusters of anomalies that have causal relationships by analysing the graph.

The temporal correlations of the anomalies are examined by the help of sliding event windows that are put in place based on a set time interval. The spatial correlation algorithm takes into account the closeness of the service dependency and forwards an anomaly in an upstream service to its probable root cause service.

4.5 Automated Remediation Intelligence

The automated remediation component detects and takes corrective action on the basis of a hybrid strategy that combines rule-based playbooks and learned policies. The playbooks are a repository of domain knowledge about common operational processes such as restarting pods, horizontal scaling processes, rolling back configurations, and routing traffic. Reinforcement learning model learns to select remedial actions in a virtual setting, where reward mechanism is comprised of a mixture of reducing MTTR and obeying SLOs.

Figure 3 shows a pipeline of intelligent auto-remediation framework including identification of a problem, choice of playbooks, and taking actions.

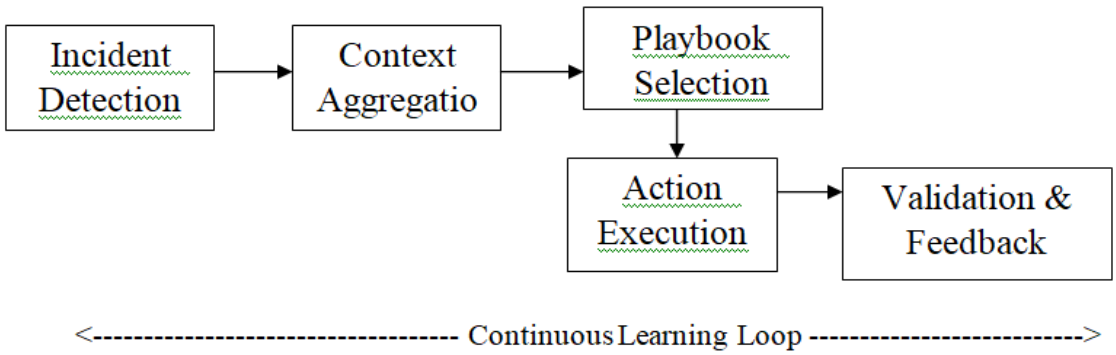


Figure 3: Intelligent Auto-Remediation Pipeline

The system is designed with a number of fail-safes that will help minimize chances of chain failures resulting from automation processes. Circuit breakers are used in the system to restrict the use of automation during major events. All the activities carried out in the

automation are captured so that they can be tracked of the situation before and during automation.

5. Implementation Considerations

Putting this framework into practice in a production cloud-native ecosystem calls for caution regarding scalability, resilience, security, and compatibility with current toolchains. This is what will be explored in the following section.

5.1 Data Pipeline Architecture

The overall data flow of the cloud-native observability system, which illustrates how telemetry data of various microservices are gathered, processed, and analyzed..The pipeline must be such that it can tolerate bursts of telemetry and not lose data. Partitioning for Kafka topics needs to be consistent with service boundaries so that they can be processed in parallel but in order per service. It has backpressure capabilities throughout the pipeline to make sure that there is no memory starvation in times of traffic burst. The cost of collecting metrics and traces to the network is minimized by the Collector running as a DaemonSet on each node of Kubernetes, and log agents are sidecars in application containers.

5.2 Model Training and Deployment

Kubeflow Pipelines are used to deploy the model training pipelines, which can be reused to repeat a machine learning process in a containerized system that benefits the Kubernetes scheduling system. Training processes are instigated by either a planned retraining, or the occurrence of data drift, which implies that there are significant changes in the statistical properties of the input data. The MLflow experimentation service manages the artifacts, hyperparameters and evaluation metrics of the training processes.

The process of serving is done with KServe (formerly KFServing), which is a Kubernetes native offering with the ability to scale itself and canary deploy and support both REST/gRPC endpoints and inference requests. One of the major operational constraints is inference latency, with the objective of having P99 inference latency of less than 50 milliseconds, so that there is no introduction of further delays in the scoring of anomalies.

5.3 Scalability and High Availability.

The operational intelligence system in itself must be available as much as possible, since failure to do so will render it hard to track and keep other systems being tracked healthily. Every node is configured with multiple replicas and the failover is automatically activated. At least three replica factors make the Kafka cluster withstand the failure of the brokers without any

data loss. The Flink streaming jobs are configured to have checkpoints to restart the processing in the event of failure.

For horizontal scaling, all components that do not have state can scale independently using Kubernetes Horizontal Pod Autoscalers (HPAs), which use custom metrics. Components of the analytics engine are the heaviest, and they need to be run on nodes with GPUs.

5.4. Security and Data Governance.

The data that is contained in the telemetry may be sensitive data like credentials, IDs, and critical business operational data. This is done by using end-to-end encryption of data-in-transit with TLS, role-based access control on all platform APIs and dashboards, data masking and redaction pipelines to remove sensitive fields prior to the analytics pipeline, and auditing logs to all accesses and remediations.

Federated solutions may be used where data sovereignty is needy and thus enable analytics operations in geographic areas without necessarily aggregating the telemetry data. Differential privacy in model training pipelines aids in the prevention of inference attacks against sensitive operational data.The integration with the existing toolchains is another feature of the new toolchain.

The implementation of this model on the enterprise relies on its successful integration with their existing working tool chains. This incorporates integrations in the tools listed below. The Incident Management integrations are available as webhook and API integrations with PagerDuty, Opsgenie, and ServiceNow to perform ticket creation, ticket enhancement, and ticket close operations automatically. Configuration Management has integrations with Terraform, Helm, and Gitops toolsing such as ArgoCD to run automated configuration remediation operations. Communication Integrations to communicate with the Slack and Microsoft teams and notify about incidents automatically and generate a summary of the incidents with the help of AI.

6. Problems and Future.

6.1 Alert Fatigue and Model Calibration.

The most significant issue of operational AI systems is maintaining the detection models in tune with the environment in which it will be used in. A trained detection model based on one set of data in one environment may not be effective in a different environment that is marked by variation in traffic, infrastructures or applications. To reduce the amount of labels used to train anomaly detectors, it is possible to use few-shot learning to ensure the best results in new settings.

The problem of alert fatigue cannot be overlooked in the use of AI in operations. The AI algorithm is significant in ensuring that the number of alerts is reduced but the level of precision should remain high because this way, the operators will not lose trust in the system. The mechanisms that can be explained are relevant in making sure that this trust is retained.

6.2 Handling Non-Stationarity

The systems based on cloud computing are subject to continual transformation due to constant software updates, changes in the infrastructure, and changing usage patterns. Under these non-stationary situations, some severe issues are associated with the application of models of machine learning that assume a certain degree of stationarity of the data distributions. It is necessary that concept drift detectors keep track of the statistical properties of the input data all the time and initiate the process of model retraining whenever they detect substantial changes in data distribution.

6.3 Explanation and Causation.

The existing anomaly correlation techniques rely on statistical associations and proximity, as opposed to causality. Recent advances in causal inference, such as causal graph learning with observational data and counterfactuals, can provide improved attribution of root causes. The application of causal graph models in the framework of anomaly correlation can help to address the issue of misdirected attribution of root causes in multifaceted, multiservice anomalies.

Explainability is one of the most important features of operational AI systems as operations staff has to make decisions on the basis of AI insights without full confidence of the model results. Additional areas of future research involve model-agnostic approaches, which can describe anomaly score and remediation predictions using SHAP and LIME feature-level explanations.

6.4 Edge and Multi-Cloud Deployments.

With the current popularity of edge computing and multi-cloud strategies, centralized operation intelligence platforms are experiencing new challenges. The data transmitted through the telemetry on edge nodes may be limited by bandwidth, which prevents real-time transfer of data to the centralized analytics system. This can be addressed with federated learning methods that learn machine learning models on edge nodes and synchronize the updates of the models at a central hub. The architecture in a multi-cloud environment must be able to connect transparently with the different cloud service provider APIs and their proprietary telemetry data representations.

6.5 Independent Work and Human Control.

The future of AI-enabled operational intelligence is the possibility of complete automation of all functioning operations, which will create a system that will handle itself with minimum human involvement. However, in order to reach this vision, there has to be answers to certain difficult questions in trust, responsibility and safe automation. Future research would involve devising methods to establish and verify safety in automated remediation plans, in order to avoid autonomous systems exacerbating an incident by making poorly-informed remedial decisions.

7.Conclusion

The paper proposes a complete stack of AI-based operational intelligence to manage large-scale distributed systems using the cloud-native. It overcomes the shortcomings of traditional monitoring and incident management by unifying machine learning-based anomaly detection, event correlation, forecasting, and automated remediation into a single and scalable system. The five-layer architecture proposed is used to convert the telemetry data into actionable insights. The data ingestion layer gathers metrics, logs, traces and events in various sources. Real-time data enrichment is facilitated by the stream processing layer. The AI/ML analytics engine does anomaly detection, log analysis, forecasting and root cause identification. The operational intelligence layer translates insights into automated actions and the observability layer lets users visualize and monitor. Autoencoders and isolation forests to detect anomalies, deep learning and NLP to analyze logs, time-series forecasting, graph-based root cause analysis, and reinforcement learning to optimize remediation strategies are some of the key AI techniques. Such methods are used to cope with the complexity of contemporary distributed systems. The issues are related to the design of the data pipeline, model accuracy, scalability, and security. Further research will deal with model adaptation, dynamic environment, causal analysis, and completely autonomous functions. In general, this framework will improve the AIOps vision of building intelligent, autonomous cloud-based infrastructure that involves less human intervention and improves operational efficiency.

References

[1] H. Chen, et al " Mobility-Aware Offloading and Resource Allocation for Distributed Services Collaboration," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 10, pp. 2428–2443, 2022.

[2] Guo Q et al.," " CausaL: A Causality-Based Service Fault Diagnosis Framework in Distributed Systems" In Proceedings of the 30th International Symposium on Software Reliability Engineering (ISSRE) (pp. 12–23). IEEE, 2020.

[3] S. Choochotkaew et al " AutoDECK: Automated Declarative Performance Evaluation and Tuning Framework on Kubernetes," in 2022 IEEE 15th International Conference on Cloud Computing (CLOUD).IEEE, 2022, pp. 309–314

[4] Meng Y et al., "Localizing Failure Root Causes in a Microservice through Causality Inference"In Proceedings of the 28th IEEE/ACM International Symposium on Quality of Service (IWQoS) (pp. 1–10). IEEE,2020.

[5] Sigelman et al " Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Google Technical Report."

[6] N. Wang, R. Zhou, L. Jiao, R. Zhang, B. Li, and Z. Li, "Preemptive Scheduling for Distributed Machine Learning Jobs in Edge-Cloud Networks," IEEE Journal on Selected Areas in Communications, 2022.

[7] Ma et al. " AutoMAP: Diagnose Your Microservice-Based Web Applications Automatically" In Proceedings of the World Wide Web Conference (WWW) (pp. 246–258). ACM.2020.

[8] Hyndman R J et al. "Automatic Time Series Forecasting: The forecast Package for R. Journal of Statistical Software", 27(3), 1–22,2008.

[9] Xu H et al., "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications" In Proceedings of the World Wide Web Conference (WWW) (pp. 187–196). ACM,2018.

[10] Hundman K et al."Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding."In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 387–395). ACM,2018.

[11] Lim C et al., " A Log Mining Approach to Failure Analysis of Enterprise Telephony Systems. In Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 557–566). IEEE,2014.

[12] Chen Y et al.,"Outage Prediction and Diagnosis for Cloud Service Systems" In Proceedings of the World Wide Web Conference (WWW) (pp. 2659–2665). ACM,2019.

[13] Benomar Z et al.,"Autonomous Self-Healing for Cloud-Native Applications Using Reinforcement Learning and Kubernetes Operators" In Proceedings of the IEEE International Conference on Cloud Engineering (IC2E) (pp. 112–123). IEEE,2021.

[14] Z. Luo et al " "Efficient pipeline planning for expedited distributed dnn training," arXiv preprint arXiv:2204.10562,

[15] Zhou Xet al., "Benchmarking Microservice Systems for Software Engineering Research" In Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings (ICSE-C) (pp. 323–324). ACM,2018.

[16] M. Yu et al ", "Gadget: Online resource optimization for scheduling ring-all-reduce learning jobs," in IEEE INFOCOM 2022-IEEE Conference on Computer Communications.IEEE, 2022, pp. 1569–1578.

[17] Wang W et al., " Automatic Fault Detection for Deep Space Exploration Using Structural Models and ML Techniques" In Proceedings of the IEEE International Symposium on High Assurance Systems Engineering (HASE) (pp. 165–172). IEEE,2018.

[18] Gulenko A et al., "Detecting Anomalous Behavior of Black-Box Services Modeled with Distance-Based Online Clustering" In Proceedings of the 2018 IEEE International Conference on Cloud Computing (CLOUD) (pp. 912–916). IEEE,2018.

[19] Nair V et al., "Finding Faster Configurations Using FLASH" IEEE Transactions on Software Engineering, 46(7), 794–811,2018.

[20] S. N. A. Jawaddi, M. H. Johari, and A. Ismail, "A review of microservices autoscaling with formal verification perspective," Software: Practice and Experience, 2022.