LLM-Enhanced Failure Localization in Microservices: Integrating Multi-Modal Data and Expert Interpretation

Zhengyu Zhong, Ruowei Fu, Minghua Ma, Shenglin Zhang, *Member, IEEE,*, Yongqian Sun, *Member, IEEE,*, Chetan Bansal, Dan Pei, *Senior Member, IEEE,*

***Abstract*—Failure localization in microservice environments is increasingly challenging. While large language models (LLMs) have shown promise in software engineering tasks, existing approaches struggle to effectively integrate multi-modal telemetry data (e.g., log, metric and trace) and provide interpretable results. This paper presents LocaleXpert, a novel failure localization system that combines specialized LLM-based agents with traditional AIOps methods to diagnose issues in microservice environments. LocaleXpert introduces three key innovations: (1) a modular pipeline that transforms metrics, logs, and traces into natural language descriptions that LLMs can effectively process, (2) specialized expert agents that analyze each data type and collaborate to identify root causes, and (3) an interpretation mechanism that produces clear, actionable explanations of its reasoning process. Evaluation results show that LocaleXpert significantly outperforming baseline approaches in both accuracy and interpretability. The system has been successfully deployed in Microsoft's AIOpsLab benchmark, demonstrating its effectiveness.**

***Index Terms*—Large language model, failure localization, microservice.**

I. Introduction

In today's rapidly evolving microservice platforms, system failures can have profound impacts on business operations, leading to significant downtime, data loss, and financial repercussions [1], [2], [3]. In this context, accurate and efficient failure localization has become a critical component of effective software management and operational resilience [4], [5], [6].

Failure localization, often considered as part of the broader root cause analysis (RCA) process, focuses specifically on identifying and pinpointing the source of a system failure. This process begins with observed system-wide symptoms and systematically narrows down the investigation to identify the underlying root cause [7], [8]. Based on the real-world failure diagnosis requirements from Microsoft's cloud operations, failure localization needs to identify root causes at two granularity levels: (1) the root cause instance level, which pinpoints the specific service or component that triggered the failure, and (2) the root cause metric level, which identifies the key performance indicators or metrics that exhibited anomalous behavior leading to the failure [9], [10]. This dual-level localization provides both the concrete failing entity and the underlying problematic measurements, enabling operators to quickly localize and resolve issues.

To effectively localize failures, microservice system operators typically collect three types of monitoring data: traces, logs, and metrics. Traces are tree-structured data that record the detailed invocation flow of user requests. Logs are semi-structured text that record hardware and software events of a service instance. Metrics are time series indicating service status, including system metrics (e.g., CPU utilization, memory utilization) and user-perceived metrics (e.g., average response time, error rate). Most automatic failure localization approaches of microservice rely on *single-modal* data to capture failure patterns. However, relying solely on single-modal data for localizing failures is not effective enough for two reasons. First, a failure can impact multiple aspects of a service instance, causing more than one modality to exhibit abnormal patterns. Using just one data source cannot fully capture these patterns and accurately distinguish between different types of failures. Second, some types of failures may not be reflected in certain modalities, making it difficult for methods relying on that modality to identify these failures [3]. These observations highlight the importance of incorporating *multi-modal* data for robust failure localization.

In recent years, the application of large language models (LLMs) in failure localization has shown promising results [11], [12]. LLMs have demonstrated potential in improving diagnostic efficiency and accuracy by leveraging their ability to process and understand vast amounts of textual data [13], [14]. However, current failure localization methods based on LLMs encounter several significant limitations. First, these methods struggle to effectively process and integrate multi-modal data that originates from a variety of sources. Since LLMs are primarily designed to process textual data, they face challenges in handling structured data such as metrics and traces. Second, there is a notable issue with the interpretability of the models used in failure localization. Many approaches based on LLMs tend to directly output root cause analysis results without providing systematic and comprehensive reasoning processes or proper explanations [12]. This lack of transparency undermines trust in the models'

Zhenyu Zhong, Ruowei Fu, Shenglin Zhang and Yongqian Sun are with Nankai University, China

Minghua Ma and Chetan Bansal are with Microsoft, USA

Dan Pei is with Tsinghua University, China

outputs, which is crucial for operators who need to make informed decisions based on these results. Third, existing methods which rely solely on LLMs may lack the ability to establish true cause-and-effect relationships between system components, events, and failures. While LLMs can identify correlations and patterns in data, they struggle to determine whether one event or condition actually caused another failure, rather than just occurring around the same time. This limitation is particularly important in microservice environments where complex chains of interactions between services can make it difficult to distinguish between true root causes and coincidental events that happen during a failure.

This paper proposes LocaleXpert, a novel failure localization system that leverages LLMs to diagnose issues in microservice environments. LocaleXpert introduces an efficient data transformation pipeline that naturally bridges structured and unstructured information. The log data is easy to be processed by the LLM-based experts in LocaleXpert. As for structured data like traces and metrics, our approach generates text descriptions from them, overcoming the inherent limitations of text-only models. To ensure accurate failure localization, LocaleXpert introduces a prompt engineering strategy where prompts are dynamically composed based on the observed failure patterns. These prompts incorporate domain-specific heuristics and operational knowledge, enabling the LLM to perform targeted reasoning about specific microservice components and failure propagation patterns. Furthermore, LocaleXpert integrates external failure localization modules. This hybrid approach combines the strengths of LLM-based reasoning with traditional failure analysis methods, ensuring robustness and flexibility in addressing complex microservice failures. Through deployment in Microsoft's AIOpsLab benchmark [9], [15] (more details can be found in Section IV), LocaleXpert has showcased remarkable practical value. When integrated into AIOpsLab's pipeline, the system successfully diagnosed various types of failures across different microservice applications. LocaleXpert also provided clear, actionable insights that helped operators implement targeted mitigations.

It is important to emphasize that LocaleXpert is not a new failure localization method in itself. Rather, its accuracy is determined by the external failure localization modules it integrates, which can be replaced as technology advances. Our primary contribution lies in automating and humanizing the failure localization process through interactive communication and interpretation modules that enhance understandability and trustworthiness.

In summary, this paper makes the following contributions:

1) We propose LocaleXpert[1], a novel agent-based framework that orchestrates and automates failure localization by integrating LLMs with traditional metric and trace-based anomaly detection and failure localization methods. Unlike most existing approaches, LocaleXpert is not a new failure localization method but rather a platform that enhances existing methods with automation, natural language interaction, and interpretability.
2) We implement an efficient multi-modal data transformation pipeline that converts structured operational data (metrics, traces) and unstructured data (logs) into natural language representations that LLMs can effectively process. The system includes fallback mechanisms to handle scenarios with missing or incomplete data modalities, addressing real-world operational challenges.
3) We develop an LLM agent interpretation mechanism that enhances the interpretability of failure localization results, enabling users to understand the reasoning process and build trust in the system's outputs. This mechanism provides detailed explanations for why certain components or metrics were identified as potential root causes.
4) We evaluate LocaleXpert through comprehensive experiments using microservice failure datasets and demonstrate its ability to improve the interpretability and user experience of failure localization. We pilot LocaleXpert into Microsoft's AIOpsLab, where representative cases have been collected, demonstrates the system's effectiveness in localizing complex software failures.

[1]Our implementation of LocaleXpert is available at https://anonymous.4open.science/r/LocaleXpert-2D6A.

II. Background and Related Work

A. Failure Localization

Failure localization in large-scale microservices is a key focus within the Artificial Intelligence for IT Operations (AIOps) community. This field aims to identify the root causes of system failures using observable data like logs, metrics, and traces. Numerous methods have been proposed for failure localization, leveraging these diverse data sources. Some approaches use logs to construct causality graphs, visualizing relationships between system components and failures [16], while others explore techniques to aggregate logs [17], [18]. Methods also analyze traces to uncover system topology and interactions between services [19], [5], [20]. Additionally, metrics-based methods identify faulty services or build causality graphs to diagnose failures [21], [22], [23], [24]. Single-modal data often fails to comprehensively reflect the state of software systems, as certain failures may only manifest anomalies in specific modalities. This limitation can lead to insufficient diagnostic accuracy in single-modal approaches. In contrast, multi-modal data provides multidimensional insights, offering a more holistic understanding of fault causes and impacts, effectively addressing the shortcomings of single-modal methods. Recent approaches integrate logs, metrics, and traces for improved accuracy and efficiency [25], [3]. Our research advances this by using large LLMs to enhance root cause analysis, offering greater accuracy and interpretability.

B. LLM Agents

LLMs are powerful tools for generating human-like text, excelling in natural language processing and reasoning. Recent advancements have turned them into "agents" that can interpret complex inputs, make decisions, and execute tasks through natural language understanding and structured reasoning [26], [27], [28]. LLM agents surpass traditional systems by accurately processing unstructured text, maintaining context for complex scenarios, and generating detailed explanations of

their reasoning, which enhances transparency for failure diagnosis. Our research focuses on developing specialized LLM agents to effectively diagnose and resolve microservice failures while ensuring transparency and reliability in their decisions.

C. LLMs in Failure Localization

In recent years, LLMs have shown significant promise in automating complex processes, including failure localization. For instance, Ahmed et al. [11] proposed fine-tuning the GPT-X model specifically for root cause identification and mitigating production incidents. However, the substantial size of large models demands significant GPU resources for fine-tuning, and as new data emerges, continuous fine-tuning is required. This process is both resource-intensive and time-consuming, which limits its scalability for real-world applications. In response to this, Zhang et al. [29] introduced an in-context learning approach for automated root cause analysis, allowing the model to avoid resource-intensive fine-tuning. By providing relevant historical events as contextual examples, this method equips the LLM with domain-specific expertise, enabling it to generate accurate root causes while reducing computational and maintenance costs. Furthermore, Chen et al. [12] developed RCACopilot, a system that collects multi-modal diagnostic data using predefined handlers. By systematically gathering and analyzing diagnostic information from various sources, it provides a comprehensive event perspective for root cause analysis, enabling the LLM to autonomously predict and explain root causes. Additionally, it reduces the cognitive load on operators and improves response speed and processing efficiency. However, all previous methods rely on powerful external API models and failed to address data privacy issues when using LLMs. To address this issue, Wang et al. [30] implemented a tool-augmented LLM on a locally deployed model, utilizing external tools as agents to enhance the model's effectiveness in RCA tasks. Despite advancements, challenges remain in improving the interpretability and explainability of LLMs in failure localization. The decision-making processes of existing methods often lack transparency. Additionally, better integration with multi-modal data (e.g., log, metric and trace) is essential for achieving comprehensive failure localization.

III. Methodology

A. System Overview

We propose a novel failure localization framework, LocaleXpert, that utilizes LLM-driven multi-agent collaboration. As depicted in Figure 1, LocaleXpert's workflow begins by processing an query for a failure localization task. First, the system pulls telemetry data from AIOpsLab [15] and filters it with a filter layer (Service/Instance Filter 1 in the figure) based on relevant dimensions such as time, service, or instance to isolate pertinent information. This filtered data is then routed through specialized pipelines tailored for traces, metrics, and logs. In the trace pipeline, the filtered trace data is analyzed to generate anomaly descriptions and perform failure localization. Both the anomaly and failure localization results are passed to the Trace Expert, an LLM designed to interpret trace-related issues. Similarly, in the metric pipeline, filtered metric data undergoes anomaly detection and failure localization, with these results being sent to the Metric Expert. Before the system processes the metric and log data, it undergoes an additional layer of filtering (Service/Instance Filter 2 in the figure) based on the root cause service identified by the trace module. This filtering ensures that only the monitoring data collected from the specified root cause service is retained, allowing the system to focus on the most relevant information. In some cases when the trace pipeline is unable to provide a valid root cause service, the system will then shift its focus to the log data. A specialized LLM is employed in Service/Instance Filter 2 to analyze the log data and identify potential root cause services. This fallback mechanism ensures that even when trace data alone is insufficient, the system can still effectively localize the failure. In the log pipeline, the system processes the filtered log data directly, as logs are already in natural language form, making them easily interpretable by the Log Expert. Finally, the Failure Localization Expert consolidates the results from all three data type experts to produce a comprehensive and accurate failure localization conclusion, which is returned to AIOpsLab for evaluation.

B. Multi-Modal Data Processing

1) Data Collection and Preprocessing: When a failure occurs in a microservice environment, operators can ask LocaleXpert to analyze the system's monitoring data to identify the root cause of the failure. The system first uses LLM to parse the user's query and determine the relevant time window for the analysis, as demonstrated in Figure 2. If no time information can be detected from the user's query, the system will prompt the user to provide a time window. If the time window is unclear, the system defaults to a predefined time window (e.g., the past 6 hours) from the moment the query is submitted. It then collects monitoring data from various sources in that time window, including metrics, logs, and traces. Each of these data modalities provides unique insights into the state and behavior of the microservice system. Metrics data typically consists of time-series information about system performance, resource utilization, and other quantitative measures. The preprocessing of metrics data involves several steps: normalization to ensure consistency across different scales, and time-series alignment to correlate metrics from different sources. Log data comprises textual records of system events, error messages, and other qualitative information, providing contextual information about system behavior. The preprocessing of log data includes filtering to remove irrelevant entries and tokenization to prepare the text for further analysis. Trace data represents the flow of requests or transactions through the various components of a distributed system. It provides insights into the dependencies between different services and the performance of individual components. Preprocessing of trace data involves reconstructing the complete path of requests, identifying service boundaries, and extracting relevant performance metrics.

2) Anomaly Detection Module: For metrics and traces, the system implements dedicated anomaly detection algorithms.

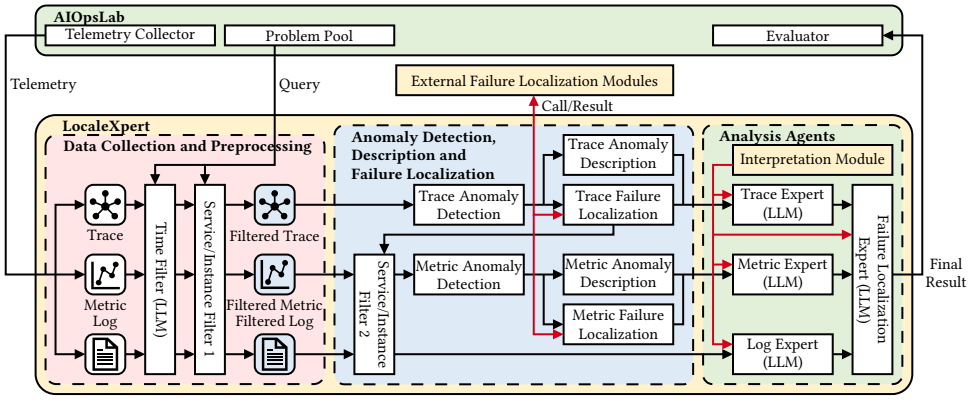

Fig. 1. The system architecture of LocaleXpert.

These anomaly detection algorithms are used to identify deviations from normal behavior in the monitoring data that may indicate the presence of a failure.

For metrics, we choose the 3-sigma rule as the anomaly detection algorithm [31]. This statistical rule states that for a normal distribution, approximately 99.7% of all data points fall within three standard deviations of the mean, making any points outside this range statistically significant outliers worth investigating. This algorithm is widely used in practice and provides a simple yet effective way to detect anomalies in metric data [32]. Experimental results have shown that 3-sigma can accurately identify outliers and unusual patterns in metrics, making it a suitable choice for our system. we also acknowledge its limitations for metrics that follow non-Gaussian patterns, such as error rates, which often exhibit long-tail distributions. To address this limitation, our modular design allows for easy substitution of alternative anomaly detection methods based on the distribution characteristics of specific metrics. For metrics with non-Gaussian distributions, operators can configure LocaleXpert to use alternative methods such as (but not limited to) the following: (1) Density-based approaches like DBSCAN, which can handle arbitrary distribution shapes; (2) Percentile-based methods that use quantiles rather than standard deviations; (3) Deep learning approaches such as OmniAnomaly [33] or USAD [34], which can effectively detect trend changes in addition to level shifts and spikes.

The trace anomaly detection module focuses on identifying unusual patterns in request flows or performance metrics derived from traces. This includes techniques for detecting long latencies, unusual service dependencies, or deviations from expected request patterns. We choose TraceAnomaly [35] as the anomaly detection algorithm for trace data, which leverages unsupervised learning techniques to identify and learn normal trace patterns during offline training. Its deployment in multiple online services has demonstrated exceptional recall and precision, significantly surpassing traditional methods and other baseline algorithms, making it the ultimate choice for robust trace anomaly detection in LocaleXpert.

In terms of log analysis, we discovered that normal log data surrounding the time of an incident can significantly aid LLMs in grasping the context of the failure. Providing the complete log data from the entire event window yields better results for fault localization compared to inputting only the anomalous logs into the model. Consequently, we opted not to perform anomaly detection on the logs.

We use these algorithms as our default anomaly detection approaches. However, LocaleXpert is designed as a modular framework rather than a single failure localization method. Like described earlier, if operators find that this method does not suit their specific data patterns, they can easily swap it out for other anomaly detection algorithms. The module's modular design supports straightforward integration of alternative methods, allowing teams to tailor the detection strategy to their unique monitoring needs.

3) Anomaly Description Module: A critical component of the system is the anomaly description module, which transforms the raw data and detected anomalies into a format that can be easily understood by LLMs using a series of predefined rules and deep learning models. This module acts as a bridge between the quantitative anomaly detection results and the qualitative interpretation needed for human understanding.

While transforming structured data (metrics and traces) to text descriptions inevitably involves some information abstraction, LocaleXpert employs several techniques to minimize information loss: (1) Temporal relationship preservation: When describing metric anomalies, our templates explicitly include timing information including start time, end time, and duration, preserving the temporal relationships that are critical for understanding failure propagation. (2) Relative magnitude context: Rather than only reporting absolute values, descriptions include relative changes (e.g., "increasing from X to Y, a Z% change") and comparisons to historical norms, helping maintain the context of the anomaly's severity. (3) Structural relationship encoding: For traces, the description templates preserve the hierarchical call paths and parent-child relationships between services, ensuring that error propagation patterns are not lost during the text conversion. (4) Multi–granularity descriptions: The system generates descriptions at multiple levels of detail, from high-level summaries to detailed explanations of specific anomaly characteristics.

To clarify the implementation specifics and how we minimize information loss during this transformation, we present the description generation process in Algorithm 1.

Algorithm 1 Anomaly Description Generation Process

Input: Metric Anomalies $\mathcal{A}_M$, Trace Anomalies $\mathcal{A}_T$, Metric Data $\mathcal{D}_M$, Trace Data $\mathcal{D}_T$
Output: Set of Textual Descriptions $\mathcal{S}$

Initialize description set $\mathcal{S} \leftarrow \emptyset$
for all metric anomaly $a \in \mathcal{A}_M$ **do**
  $f_{stat} \leftarrow$ EXTRACTSTATS($a, \mathcal{D}_M$) {Extract start/end time, peak, mean, std dev}
  $p_{type} \leftarrow$ PATTERNMATCHER($a, \mathcal{D}_M$) {Classify pattern: spike, dip, increase, decrease, level shift up/down, fluctuations, etc.}
  $t_{template} \leftarrow$ SELECTTEMPLATE(p_{type})
  $s_{desc} \leftarrow$ FILLTEMPLATE($t_{template}, f_{stat}$)
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{desc}\}$
end for
for all trace anomaly $a \in \mathcal{A}_T$ **do**
  $path_{curr} \leftarrow$ GETCALLPATH($a, \mathcal{D}_T$)
  if ISSTRUCTURALANOMALY(a) **then**
    $path_{diff} \leftarrow$ COMPARESTRUCTURE($path_{curr}$, BaselinePath)
    $s_{desc} \leftarrow$ FORMATSTRUCTUREDESC($a.traceID, path_{diff}$)
  else
    $nodes_{slow} \leftarrow$ IDENTIFYSLOWNODES($path_{curr}$, 3-Sigma)
    $s_{desc} \leftarrow$ FORMATTIMEOUTDESC($a.traceID, path_{curr}, nodes_{slow}$)
  end if
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_{desc}\}$
end for
return $\mathcal{S}$

In detail, for metrics, the description module generates textual summaries of the observed anomalies, including information about the affected metrics, the magnitude and duration of deviations, and any relevant contextual information. For the recognition of metric anomaly patterns, we use a deep learning-based anomaly pattern classification model called PatternMatcher [36], which classifies anomalies into 11 common patterns. For example, a CPU usage spike can be described as

> *The cpu_kernel_norm_pct metric for the service logservice2 is abnormal, displaying an anomaly pattern characterized by multiple spikes. This anomaly began on 2021-07-02 at 00:09:12 and ended at 00:23:42, reaching a peak of 0.002457, which is an increase from the previous value of 0.001223. The anomaly score is 1.16.*

For trace data, the description module generates narratives that explain the flow of requests through the system, highlighting any unusual patterns or performance issues. This can include descriptions of service dependencies, latency issues, or errors encountered during request processing. Considering that anomalies in traces can be categorized into two types: call path anomalies (such as interruptions caused by network failures, service crashes, or other unexpected events) and call timeouts (such as network request timeouts caused by network delays, packet loss, or target server failures), LocaleXpert designs two templates for describing trace anomalies. For example, a call path anomaly might be described as

> *There is a structural anomaly in the trace. The trace ID is d3e78653683c2cbc, with the trace call starting at 2021-07-02 00:18:44 and ending at 00:18:56.*

A call timeout anomaly is detected when the execution time exceeds the expected duration, calculated using the 3-sigma principle based on historical normal call path propagation times. It can be described as

> *The trace call with trace ID ea6799e3d0dfb480 timed out. It started at 2021-07-02 00:20:54 and ended at 00:20:56, following the call path: webservice2 -> logservice1 -> logservice2 -> dbservice2 -> redisservice1. Several services exhibited anomalies:*
> - *logservice2: Call time was 1788 ms, exceeding the normal upper limit by 1056 ms.*
> - *logservice1: Call time was 1866 ms, exceeding the normal upper limit by 1030 ms.*
> - *webservice2: Call time was 2150 ms, exceeding the normal upper limit by 976 ms.*
> - *dbservice2: Call time was 1743 ms, exceeding the normal upper limit by 1090 ms.*

As for log data, since logs are already in textual format, we do not need to generate additional descriptions. Instead, we preprocess the logs to remove irrelevant entries and input them directly into the LLM for further analysis.

While our anomaly description module uses predefined templates, we emphasize that these templates are designed with atomic, composable features rather than rigid, monolithic patterns. Each template extracts fundamental characteristics that can be freely combined to describe various anomaly types. For metrics, the atomic features include: anomaly pattern type, temporal boundaries, magnitude information, and statistical indicators. For traces, the atomic features comprise: structural properties, performance metrics, and anomaly classifications. These atomic features serve as building blocks that can be recombined to accurately describe both common and novel failure scenarios. For instance, a previously unseen failure pattern manifesting as a cascading timeout with intermittent spikes can be described by composing the atomic features of "timeout", "call path sequence", "spike pattern", and "temporal progression", without requiring a dedicated template for this specific combination, providing the flexibility to handle unforeseen failure patterns while maintaining consistency in the information provided to downstream LLM agents. Moreover, the modular design allows operators to extend the feature set or adjust the composition rules as new types of anomalies are encountered in production environments. Since our model does not require fine-tuning, modifying the feature set does not significantly impact performance or introduce additional training overhead.

4) Failure Localization Module: The failure localization module employs an agent that analyzes the generated anomaly descriptions to localize culprits of a failure and calls external failure localization methods to analyze trace and metric data more deeply. The output is then translated into plain text and fed into subsequent LLMs for further insights.

For trace data, we choose MEPFL [20] as the failure localization method. It extracts a set of features from system trace logs that reflect the dynamic environment and interactions of microservices, describing the state of the microservices system from configuration, resource, instance, and interaction perspectives. Based on these extracted features, it trains a series of models, including random forests, k-nearest neighbors, and multi-layer perceptron, to effectively analyze system behaviors under normal and failure conditions. This enables it to predict latent errors and identify faulty microservices within complex microservice architectures.

For metric data, we employ MicroCause [24] for failure localization. It uses a straightforward yet effective path condition time series (PCTS) approach to capture the time-lag characteristics that exist in the propagation of failure across metrics and incorporates a temporal cause-oriented random walk method. It integrates causal relationships, temporal order,

and priority information from metirc data to achieve the ranking of root cause components.

The output of the failure localization module is a ranked list of potential root causes, accompanied by supporting evidence and confidence levels for each hypothesis.

C. Analysis Agents

1) Agent Design: The analysis agents, including the Trace Expert, Metric Expert, Log Expert, and Failure Localization Expert, are designed to bridge the gap between the technical analysis performed by the anomaly detection and failure localization modules and the need for human-readable, actionable insights. These LLM agents leverages carefully crafted prompts to generate responses that are both technically accurate and easily understandable by system operators.

The design of the expert model focuses on creating a knowledge base that encompasses common failure scenarios, best practices in microservice management, and domain-specific terminology. This knowledge is input into the model through carefully designed prompt templates. For each data type (metrics, logs, and traces), specific prompt structures are developed to guide the model in generating relevant and insightful interpretations. These prompts are designed to elicit explanations that cover the what, why, and how of detected anomalies and proposed root causes.

For example, a prompt for interpreting metric anomalies might include instructions like:

> **Knowledge:**
> Top root cause metrics and their anomaly descriptions: [ANOMALY DESCRIPTIONS]
> Let's think step by step. Please provide your detailed observation of metric anomaly events and the root cause metrics, using at least six sentences. During the metric anomaly analysis, you can integrate the anomaly descriptions for reasoning and analysis. In the root cause metrics analysis, you should combine the anomaly descriptions with the top-5 root cause metrics for reasoning and analysis. You should provide detailed impact relationships between each metric and infer the final answer based on these relationships. The output format is:
> **Observation:** ... **Reasoning:** ... **Final Answer:** ...
> In this format, "Observation" involves using the anomaly descriptions to provide observation results. "Reasoning" is the process of thinking and reasoning based on observations to arrive at the final result. The "Final Answer" is the conclusion drawn from observation and reasoning. The final result of the metric anomaly analysis should identify the metrics that indicate the presence of anomalies, referring only to the information described in the anomaly, and providing three to five candidate metrics. The final answer for the root cause metric analysis should combine the top-5 root cause metrics to identify specific root cause metrics.
> Please refer to the following example when conducting root cause metric analysis: [EXAMPLE]
> Note that you must ONLY discuss your observations, reasoning, and suggestions, and do not discuss anything else!

Similar prompt structures are developed for log and trace data, each tailored to the specific characteristics and insights provided by these data types.

2) Interpretation Module: A key feature of the analysis agents is their ability to provide clear rationales for their analyses and recommendations. This transparency is essential for building trust in the system's outputs and empowering operators to make informed decisions. All of our LLM agents are equipped with an interpretation module that employs specially designed prompts to encourage them to explicitly state the evidence underpinning their conclusions. This evidence is drawn from anomaly descriptions, failure localization results, and their knowledge base of microservice infrastructure principles.

For each potential root cause identified, the interpretation module provides a detailed explanation of how this cause could lead to the observed anomalies across all data types (metrics, logs, and traces). For example, as illustrated in Figure 3, when multiple log anomaly events show panic errors across multiple services with messages indicating no reachable servers, alongside repeated calls to *main.initializeDatabase* function, the module explains how database or network connectivity issues could be the root cause leading to service-wide failures. The interpretation connects these observations by showing how connectivity problems would trigger both the unreachable server errors and the repeated database initialization attempts as services try to reconnect. This approach ensures that the system's outputs can help operators understand the reasoning behind the diagnoses better.

D. Handling Missing Data Modalities

In real-world operational environments, microservice monitoring systems often face challenges with incomplete data. Some services may not be fully instrumented, or certain data types may be temporarily unavailable due to collection failures. LocaleXpert is specifically designed to handle these scenarios through several mechanisms. First, each data type (logs, metrics, traces) has its own dedicated expert agent that can operate independently, allowing the system to generate insights even when certain data types are missing. Second, the Service/Instance Filter 2 (as shown in Figure 1) implements a fallback mechanism that can shift from trace-based filtering to log-based filtering when trace data is unavailable or insufficient for determining the root cause service. Even when both logs and traces are absent, the system can perform failure localization solely based on metrics, though the performance may degrade due to the lack of service-level filtering, resulting in a broader search space and potentially lower localization accuracy. Third, when making diagnoses with incomplete data, LocaleXpert's interpretation module explicitly communicates areas of uncertainty and the limitations of the analysis, helping operators understand the confidence level of the conclusions. Finally, if the LLM-based analysis is inconclusive, the system can fall back to traditional failure localization methods that do not rely on LLMs, ensuring that some level of analysis can still be performed.

For example, in scenarios where distributed tracing is not fully implemented across all services, LocaleXpert can still detect root causes in untraced services through log and metric analysis. The system leverages log patterns and metric anomalies to identify potential issues in services that do not appear in the trace data. Similarly, for gradual failures like memory leaks that may not trigger threshold-based anomaly detection in metrics, LocaleXpert can identify these through log analysis (e.g., detecting increasing garbage collection frequency) and through the external failure localization modules which employ more sophisticated detection methods beyond simple threshold-based approaches.

IV. Case Study

Through our ongoing collaboration with Microsoft, LocaleXpert has been actively deployed in Microsoft's AIOpsLab[2] [9], [15] to test and improve LocaleXpert's ability to diagnose various real-world application failures. AIOpsLab is an integrated framework designed to evaluate AI agents that support cloud automation, aiming to drive the full lifecycle automation of cloud operations through the implementation of the AgentOps concept. The framework integrates two microservice applications: SocialNetwork (containing 28 microservice components such as Memcached, MongoDB, and Redis) and HotelReservation (containing 20 microservice such as search and recommendation). To comprehensively assess the capabilities of AI agents, AIOpsLab provides two main types of faults: symptomatic faults (such as performance degradation and service interruptions) and functional faults (such as configuration errors and code defects). These refined fault designs and task benchmarks enable a comprehensive evaluation of agent performance in dynamic cloud environments.

Among numerous successful deployments and validations, we present a particularly illustrative case study that demonstrates LocaleXpert's effectiveness in resolving failures in enterprise settings. Specifically, we examine how the system diagnosed a buggy app image failure that manifested uniquely in metrics and logs while showing no significant trace anomalies. This case not only highlights LocaleXpert's capability to integrate and analyze multi-modal data effectively but also showcases its practical value in solving real industrial challenges.

As illustrated by Figure 2, the diagnostic process begins with LocaleXpert's data filtering mechanism, which operates across multiple dimensions including time, service, and instance parameters. This initial filtering stage is crucial for isolating the most pertinent information from the vast amount of monitoring data generated by the microservice infrastructure. What makes this filtering process particularly effective is its adaptive nature. After the log module identifies *geo* as the potential root cause service, the system applies an additional layer of filtering to metric data stream, ensuring that only the most relevant data points are considered for analysis.

The system then processes the filtered data through three specialized pipelines, each designed to handle a specific data type. In the log pipeline, the Log Expert leverages the LLM's advanced reasoning capabilities to analyze the filtered log entries directly. This analysis proves particularly valuable in this case, as the expert successfully identifies that the system failure is specifically related to no reachable servers in the geo service.

Concurrently, the metric pipeline conducts its analysis through multiple stages. The filtered metric data first undergoes anomaly detection and description generation, followed by a detailed failure localization process. The Metric Expert then receives this processed information and, through sophisticated analysis combining both the anomaly descriptions and failure localization results, confirms *kpi_container_cpu_usage_seconds_total* as the root cause metric of the failure. This conclusion further leads to the identification of specific performance bottlenecks in the root cause service (i.e, the geo service) allowing for targeted remediation strategies.

In the trace pipeline, although the system performs its standard sequence of anomaly detection, description generation, and failure localization, the Trace Expert's analysis proves less conclusive in this particular case. This is actually an important finding, as it demonstrates LocaleXpert's ability to handle scenarios where certain data modalities may not exhibit clear signs of failure. Instead of forcing incorrect conclusions from the trace data, the system appropriately acknowledges the lack of trace anomalies and relies more heavily on the insights gained from logs and metrics.

The final stage of the diagnostic process (Figure 3) showcases one of LocaleXpert's most important features: the integration and synthesis of multiple expert analyses. The Failure Localization Expert consolidates the findings from all three expert modules, weighing their relative contributions based on the strength and relevance of their insights. In this case, the expert produces a clear, actionable conclusion: "There is a panic error *no reachable servers* in the *geo* service." This concise yet precise diagnosis encapsulates both the location (*geo*) and nature (*no reachable servers*) of the failure, providing operators with the exact information needed to address the issue.

This case study demonstrates LocaleXpert's ability to handle complex scenarios where failures may not manifest uniformly across all data types. The system's sophisticated data processing pipelines, expert-driven analysis, and intelligent synthesis of findings enable it to produce accurate and actionable diagnoses even when working with incomplete or inconsistent data across different modalities.

V. Experiment

While our case study demonstrated LocaleXpert's effectiveness when integrated with Microsoft's AIOpsLab benchmark, a systematic experimental evaluation is still necessary to quantify the system's performance and understand the contribution of each component.

A. Experimental Setup

The primary focus of our experiments was on assessing LocaleXpert in failure localization and the interpretability of its reasoning process. The interpretability analysis concentrated on the results produced by three agents: the Log Expert, Metric Expert, and Trace Expert. In addition, an ablation study was conducted to isolate the contributions of specific LocaleXpert components, including the anomaly description module, the external failure localization module, and the interpretation module. Finally, we evaluated the trade-offs between model size, latency, and performance to inform practical deployment decisions.

1) Datasets: For our evaluation, we utilized two popular datasets:

Generic AIOps Atlas Dataset (GAIA Dataset)[3] was

[2]https://github.com/microsoft/AIOpsLab

[3]https://github.com/CloudWise-OpenSource/GAIA-DataSet

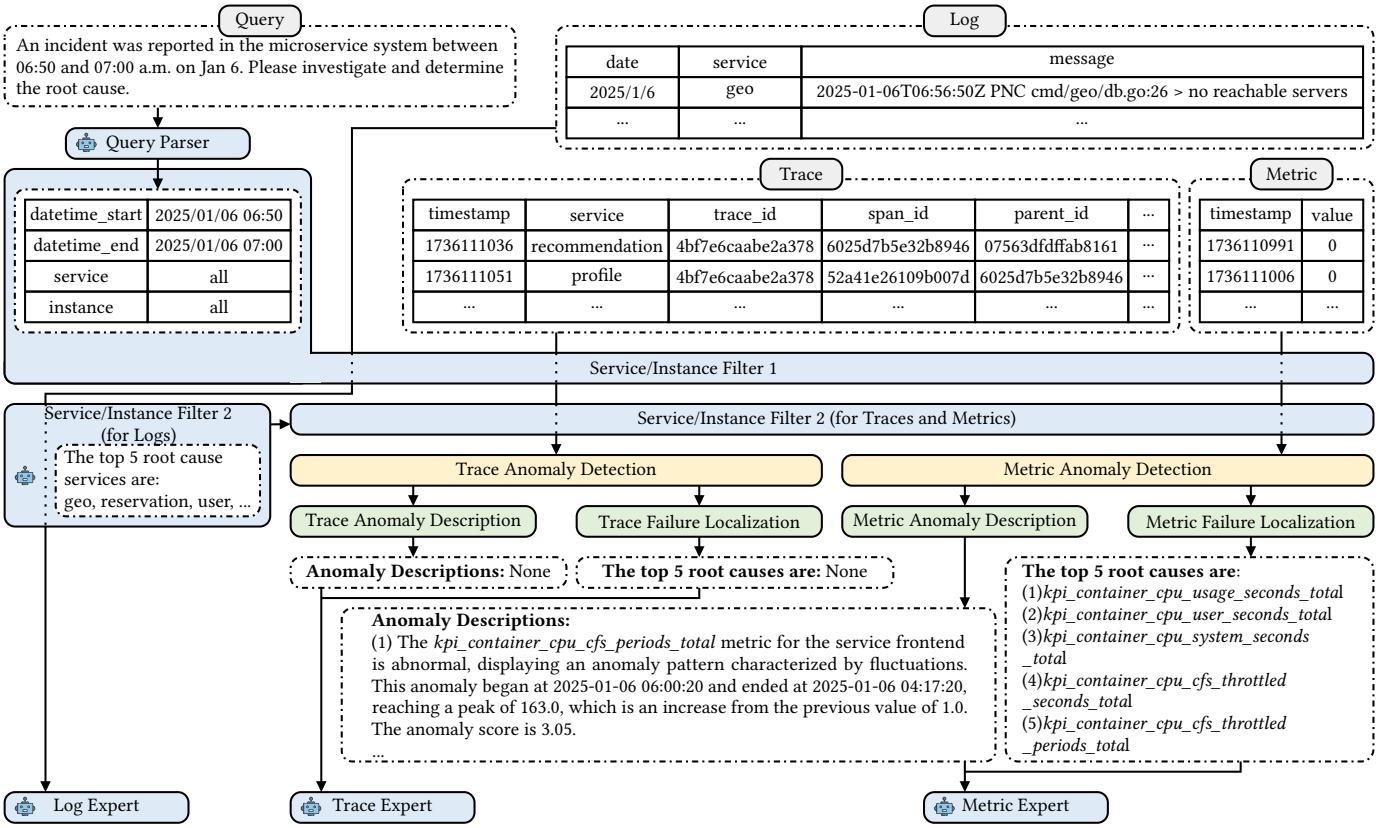

Fig. 2. The data processing workflow of LocaleXpert.

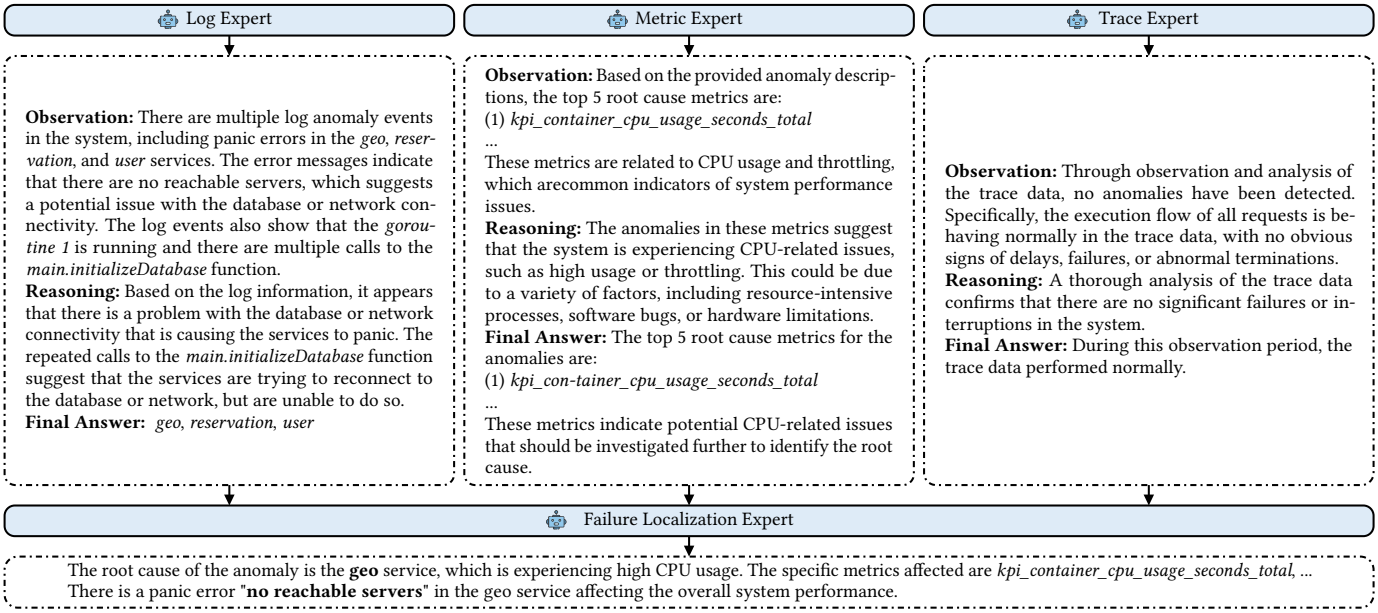

Fig. 3. The failure localization process of LocaleXpert.

sourced from CloudWise and comprises multi-modal data from a business simulation system, MicroSS. This system includes 10 instances and generated over 87 million logs, 0.7 million metrics, and 28 million traces over two weeks. The dataset encompasses five types of failures: system-level failures such as system stuck and process crashes, and service-level failures such as login failures, file missing, and access denied issues.

CCB Cloud Dataset[4] was collected from a real-world e-commerce system implementing a microservices architecture across 40 instances. Deployed on CCB Cloud (a large bank in China), the system's traffic patterns aligned with real business scenarios. The dataset incorporates 15 distinct failure types derived from real systems, including resource failures (CPU, memory, disk), network failures (packet loss and latency), and application failures (process termination).

Table I provides an overview of the two datasets, including the number of microservices, instances, failure rates, and data volumes. The diversity and complexity of these datasets present a robust test for evaluating the performance of failure localization and reasoning in multi-modal, large-scale systems.

TABLE I
DATASET INFORMATION

	GAIA	CCB Cloud
Microservices	5	10
Instances	10	40
% Failures	4.908	3.125
Log	87,974,871	26,035,188
Metric	734,165	18,497,600
Trace	28,681,438	44,858,393

While our current datasets do not match the scale of the largest cloud environments (such as Azure's 1000+ microservice deployments), they represent realistic production-like environments with sufficient complexity to validate our approach. We discuss scaling considerations for larger deployments in Section V-B3 and Section VI.

2) Baseline Methods: We organized the comparative methods into two categories:

(A) Traditional (non-LLM) failure localization methods. **Eadro** [2] uses all three types of multi-modal data for failure localization by integrating anomaly detection and root cause localization. It classifies failure cases and treats failure-free cases as a separate class. To better utilize monitoring data, it applies three specialized fusion methods and combines them via a graph attention network. **DiagFusion** [3] incorporates CMDB data for propagation paths and characteristics. It extracts, serializes, and converts events into vectors, then builds a dependency graph for GNN training using traces and deployment data. A neural network encodes events to learn their representation. **PDiagnose** [37] performs anomaly detection on each data type and builds an anomaly microservice queue. A voting system identifies the root cause.

(B) LLM-based approaches. **ReAct** [38] integrates chain-of-thought style reasoning with explicit actions (tool calls) so an LLM can alternate between reasoning and interacting with external analyzers or data sources. This enables flexible problem-solving and dynamic information gathering.

RCACopilot [39] and **RCAgent** [30] are also two popular LLM-augmented root-cause analysis systems that couples learned reasoning with automation and external tool use to improve diagnosis and explanations. However, they are not publicly available and their prompts/configurations are not published, which prevents faithful reproduction and fair evaluation in our controlled experiments. Consequently, RCACopilot and RCAgent could not be included as an evaluated baseline. For this study we therefore use ReAct as the LLM-based baseline and compare against the traditional methods described above.

3) Implementation: LocaleXpert was implemented using Qwen 3 8B Instruct as the core language model, with the entire setup developed in PyTorch 2 and accelerated by CUDA for GPU utilization. The experiments were conducted on a high-performance Linux server equipped with two Intel Xeon Gold 5416S CPUs and one NVIDIA A6000 GPU with 48GB of VRAM. We acknowledge that this configuration may impose higher demands on low-resource environments. The minimum hardware requirements align with the minimum runtime requirements of Qwen 3 8B. Additionally, general low-resource optimizations, such as quantization, can be applied to reduce overhead. If these measures are still insufficient, a lower-overhead model can be substituted. Specific considerations for model selection are discussed in Section V-B3.

4) Evaluation Metrics: We employed multiple metrics to assess the efficacy of LocaleXpert in failure localization and reasoning generation. For failure localization, we designed the agent to output its reasoning in a fixed format after completing the regular inference process. This fixed format explicitly includes the agent's identified root cause metrics. We then directly extracted the root cause candidates from this fixed format and compared them with the ground truth in the dataset to measure accuracy. The Top-k accuracy (A@k) was used, where A@k measures the probability that the identified root cause is among the top k results. We report results for $k = 1, 3,$ and 5.

For reasoning generation, we used a combination of lexical and semantic metrics to evaluate the similarity between the generated text and reference text. Since the original datasets do not contain reasoning text, we invited three experienced operations engineers to carefully construct and review the reference text to ensure its correctness and authority.

We use lexical metrics including BLEU-4 [40] and ROUGE-L [41] to evaluate the reasoning quality. BLEU-4 [40] is precision-focused, comparing 4-grams between generated and reference text, while ROUGE-L [41] emphasizes recall through the longest common subsequence. These metrics primarily assess textual similarity rather than diagnostic accuracy. They provide valuable insights into how well the system can articulate its reasoning process in a way that matches expert explanations, which is important for interpretability and trustworthiness.

Given that lexical metrics often fail to capture the semantic nuances of language, we employed a LLM-based evaluation. Specifically, GPT-4 was used to evaluate the clarity and

[4]The CCB Cloud Dataset is confidential due to their policy.

coherence of the reasoning outputs, providing a human-aligned judgment of the semantic congruence between generated and reference texts [42], [43], [30]. This was scored on a scale from 0 to 1, denoted as G-sim (GPT-similarity). Additionally, to ensure an accurate qualitative assessment, our invited operations engineers manually reviewed the outputs of all methods and identified, for each test case they evaluated, the answer that demonstrated the most reasonable and correct reasoning. Unfortunately, due to the large volume of original data, the invited engineers were unable to evaluate every test case. Therefore, we randomly sampled 10% of test cases for manual evaluation. For each sampled test case, each engineer independently voted for the method they considered best. The win rate (W-rate) for a method is calculated as the proportion of cases where it received the majority vote (at least 2 out of 3 votes). In cases where no method received a majority (i.e., all three engineers voted for different methods), the engineers discussed their perspectives and conducted a second round of voting to reach consensus. This situation occurred in 9.46% of the sampled evaluation cases.

B. Experimental Results

TABLE II
FAILURE LOCALIZATION PERFORMANCE (FL: EXTERNAL FAILURE LOCALIZATION)

Model	GAIA			CCB Cloud		
	A@1	A@3	A@5	A@1	A@3	A@5
Eadro	0.3027	0.5648	0.6117	0.1904	0.3729	0.5301
DiagFusion	**0.6070**	0.8956	**1.0000**	0.5000	0.8140	0.8837
PDiagnose	0.2970	0.5446	0.6733	0.1778	0.5111	0.5778
ReAct	0.2231	0.3345	0.4507	0.0834	0.1667	0.2144
LocaleXpert	0.4330	**0.9377**	0.9643	**0.7500**	**0.8750**	**0.9375**
w/o FL	0.3274	0.6753	0.8779	0.0834	0.2500	0.5000

TABLE III
TRACE EXPERT REASONING PERFORMANCE (AD: ANOMALY DESCRIPTION, FL: EXTERNAL FAILURE LOCALIZATION, IM: INTERPRETATION MODULE)

Model	GAIA				CCB Cloud			
	BLEU-4	ROUGE-L	G-sim	W-rate	BLEU-4	ROUGE-L	G-sim	W-rate
ReAct	0.0139	0.1802	0.5075	0.1544	0.0114	0.1963	0.4333	0.0000
LocaleXpert	**0.0448**	**0.3479**	**0.8125**	**0.3848**	**0.0654**	**0.3636**	**0.8556**	**0.5380**
w/o AD	0.0172	0.3188	0.7313	0.0772	0.0312	0.3217	0.6611	0.1544
w/o FL	0.0189	0.2820	0.7350	0.3076	0.0338	0.3385	0.6978	0.2304
w/o IM	0.0119	0.1804	0.6750	0.0760	0.0346	0.2357	0.6200	0.0772

TABLE IV
METRIC EXPERT REASONING PERFORMANCE

Model	GAIA				CCB Cloud			
	BLEU-4	ROUGE-L	G-sim	W-rate	BLEU-4	ROUGE-L	G-sim	W-rate
ReAct	0.0172	0.1844	0.4075	0.0772	0.0179	0.2494	0.4938	0.0000
LocaleXpert	**0.0528**	**0.4060**	**0.7460**	**0.6152**	**0.0759**	**0.4238**	**0.7625**	**0.3848**
w/o AD	0.0327	0.3390	0.6500	0.0772	0.0441	0.3785	0.7063	0.2304
w/o FL	0.0375	0.2804	0.6700	0.1544	0.0255	0.3484	0.6938	0.2304
w/o IM	0.0208	0.2144	0.6150	0.0760	0.0221	0.2661	0.6175	0.1544

TABLE V
LOG EXPERT REASONING PERFORMANCE

Model	GAIA				CCB Cloud			
	BLEU-4	ROUGE-L	G-sim	W-rate	BLEU-4	ROUGE-L	G-sim	W-rate
ReAct	0.0340	0.3334	0.7425	0.2005	0.0350	0.3338	0.7854	0.1544
LocaleXpert	**0.0632**	**0.4776**	**0.8562**	**0.7995**	**0.0401**	**0.2890**	**0.8409**	**0.8456**

1) Effectiveness: Table II presents the failure localization performance of different methods across the two datasets. The experimental results demonstrated LocaleXpert's superior performance across both datasets, with particularly notable improvements in handling complex data structures. In the GAIA dataset, LocaleXpert achieved significant improvements in failure localization, with A@3 accuracy reaching 0.9377 compared to ReAct's 0.3345. The performance gap widened further in the CCB Cloud dataset, where LocaleXpert achieved an A@3 accuracy of 0.8750 versus ReAct's 0.1667. Although DiagFusion outperforms LocaleXpert on some localization metrics (e.g., higher A@1/A@5 on GAIA), likely due to its graph-based propagation and CMDB integration, LocaleXpert remains competitive with only modest gaps and delivers substantially better interpretability.

Furthermore, LocaleXpert achieves comparable or superior results on several reasoning and human-aligned metrics, demonstrating a favorable accuracy-explainability trade-off. Table III presents the reasoning performance of the Trace Expert across the two datasets. Similarly, Table IV illustrates the reasoning performance of the Metric Expert, while Table V shows the reasoning performance of the Log Expert. A particularly noteworthy finding emerged in the handling of complex trace data. In the CCB Cloud dataset, which features more intricate call paths due to its larger instance count, LocaleXpert demonstrated exceptional capability improvements. The G-sim metric, which evaluates how close the generated reasoning is to the reference reasoning, showed a remarkable increase of 42.23 points compared to ReAct, substantially higher than the 30.5-point improvement observed in the GAIA dataset. The W-rate metric also reflected this trend, with LocaleXpert achieving the highest W-rate in both datasets, indicating that while no method produced the best reasoning in any case, LocaleXpert consistently outperformed ReAct in terms of reasoning quality. This performance differential underscores LocaleXpert's enhanced capability in processing complex data structures and producing more accurate and useful reasoning results. For simpler data types, such as logs, the performance improvements were more modest. The G-sim improvements of 11.37 and 7.5 points across the two datasets for log analysis suggest that while LocaleXpert maintains superior performance, the magnitude of improvement correlates with data complexity, and how well the "language model" can understand the multi-modal data.

2) Ablation Study: The ablation study, summarized in Tables II, III, IV, V reveal the importance of each component of LocaleXpert. Since the agents and modules in LocaleXpert communicate through dynamically generated prompts, we conduct the ablation experiments by removing the corresponding

prompt segments generated by each target module, while keeping the remaining system architecture intact. This approach allows us to isolate and evaluate the contribution of each module while maintaining the system's basic functionality.

The anomaly description module is designed to assist the agent in better understanding the anomaly data, while the interpretation module enhances the rationality of the model's output and ensures the correctness of the reasoning process, primarily for human interpretation. Since the accuracy of failure localization heavily relies on the accuracy of the external methods invoked by the agent, the experiments on failure localization accuracy do not include comparisons with methods that remove the anomaly description module or the interpretation module.

Without the anomaly description module (w/o AD). We observed a significant drop in all evaluation metrics, especially in reasoning about trace data. A trace refers to the call chain process of a request in a distributed system, capturing the sequence of calls and timing information across various services or instances. The complexity and volume of trace data pose challenges for LLMs in accurately interpreting and processing it, leading to a sharp decrease in reasoning performance. By producing clear and precise anomaly descriptions in natural language, this module helps LLMs better understand complex monitoring data, enabling them to deliver more accurate and contextually relevant insights.

Without the external failure localization module (w/o FL). We saw a noticeable drop across all metrics, such as BLEU-4 falling from +0.054 to +0.0224 in Trace Expert reasoning, though this still outperforms ReAct. By identifying the root cause service beforehand, these external modules provide LLMs with crucial context, allowing them to focus on anomalies related to specific services. This targeted approach minimizes the ambiguity that LLMs typically encounter when reasoning in complex systems. Additionally, as shown in Table II, relying solely on LLMs for failure localization yields relatively low accuracy, with an A@5 score of 0.4507 in ReAct. Combining LLMs with traditional failure localization techniques results in more accurate and reliable reasoning.

Without the interpretation module (w/o IM). Compared to other components in the ablation study, the metrics for eliminating the interpretation module showed the steepest decline. This experiment demonstrates that the interpretation module plays a pivotal role in guiding the agents' reasoning processes, with the quality of the prompt directly influencing the agents' ability to produce correct outputs, i.e., the correct and coherent reasoning results in the desired format. Poorly optimized prompts often lead to vague or inconsistent responses, limiting the agents' ability to handle complex data and generate the expected results. Thus, the interpretation module is essential for ensuring that LLMs generate reliable and high-quality outcomes.

3) Model Size, Latency and Performance Trade-off: We measured single-case end-to-end inference latency and per-agent reasoning quality for three model sizes (Qwen3 4B, 8B, and 14B) to evaluate scalability and the practical trade-off between latency and performance. This analysis is necessary for real-world deployment decisions, as operations teams must balance diagnostic accuracy with response time requirements. Selecting an appropriately sized model directly impacts both the mean time to recovery (MTTR) and the computational resources required in production environments. In this experiment, latency is reported as the average time per case (seconds). For performance we summarize average G-sim across the three experts (Trace, Metric, Log) and report Top-1 localization accuracy.

TABLE VI
AVERAGE SINGLE-CASE LATENCY AND AGGREGATED PERFORMANCE

Dataset	Model	Latency (s)	Avg G-sim	A@1
GAIA	4B	25.743	0.7074	0.410
	8B	43.437	0.7983	0.433
	14B	53.835	0.8105	0.436
CCB Cloud	4B	17.830	0.7396	0.712
	8B	28.722	0.8070	0.750
	14B	40.384	0.8163	0.721

As shown in Table VI, the results reveal distinct trade-offs between model size, inference latency, and diagnostic performance. The 4B model achieved the lowest latency but delivered the weakest reasoning quality and comparable localization accuracy. Scaling to 8B brought substantial improvements in reasoning quality with marginal gains in localization, at the cost of increased latency. The low accuracy gain is expected since localization relies more on external modules than LLM size. The 14B model further improved reasoning and localization, but with the highest latency, representing a 100% increase over the 4B baseline. Interestingly, the A@1 accuracy on the CCB Cloud dataset decreased slightly when moving from 8B to 14B, suggesting potential overfitting or diminishing returns at larger scales for this specific task and dataset.

We found several practical considerations for deployment through this analysis. First, the 4B model offers an attractive option for time-sensitive scenarios, accepting slightly reduced reasoning quality. Second, the 8B model provides a balanced trade-off, delivering substantial improvements in reasoning interpretability while maintaining acceptable latency for most operational contexts. Third, the 14B model should be reserved for scenarios where maximum diagnostic accuracy justifies the computational overhead, or when infrastructure can support parallel inference to mitigate latency concerns.

VI. DISCUSSION

A. Lessons Learned

Through extensive experimentation and real-world deployment of LocaleXpert, we have identified several valuable insights that can inform future research and development in LLM-based microservice failure localization.

First, we learned that LLM-based interpretation requires careful structure; the ablation studies revealed that removing the interpretation module led to the steepest decline in reasoning quality, indicating that LLMs need carefully structured prompts and interpretation frameworks to produce reliable reasoning about system failures. Second, despite the

power of LLMs, traditional failure localization methods remain valuable, as shown by the notable performance drop when the external failure localization module was removed. The system's accuracy fell from 0.9375 to 0.5000 for Top-5 results in the CCB Cloud dataset, suggesting that hybrid approaches combining LLMs with traditional tools may be more effective than pure LLM solutions. Third, proper data pre-processing proved essential, with the anomaly description module playing a crucial role in bridging the gap between structured operational data and natural language understanding.

Through experimentation with different foundation models, we found that the key requirement for the foundation model is its ability to handle structured technical data and maintain consistent reasoning chains. Moderate-sized models (like Qwen 3 8B) strike an optimal balance for failure localization tasks. While larger models or closed-source models offered marginal improvements in natural language understanding, they introduced significant latency that could impact real-time diagnosis.

Additionally, our experience also revealed insights about mitigating hallucination through careful prompt engineering. We discovered that decomposing the failure analysis into explicit steps and enforcing structured intermediate outputs significantly reduced model hallucination. Specifically, requiring the model to first summarize the observed anomalies, then analyze individual data modalities (metrics, logs, traces), and finally synthesize findings using pre-defined templates, led to more reliable outputs compared to direct root cause inference.

While our current implementation utilizes Qwen 3 8B on GPU hardware, the system's modular design allows deployment flexibility. LocaleXpert achieves reasonable latency for failure localization tasks, with most diagnoses completed in under 30 seconds in our test environment. This response time is acceptable for operational contexts where automated diagnosis significantly reduces the overall time-to-resolution compared to manual investigation. Additionally, the computational requirements can be adjusted by implementing quantization techniques or selecting alternative foundation models based on specific resource constraints.

B. Generality of Results

The findings and approach presented in this research have broad applicability beyond the specific context of our study. While our evaluations focused on specific datasets (GAIA and CCB Cloud) and deployment in Microsoft's AIOpsLab, the underlying principles and methodologies can be adapted to diverse microservice environments.

The multi-modal data processing approach we developed is inherently flexible and can be applied to any microservice architecture that generates standard observability data (logs, metrics, and traces). The system's modular design allows organizations to substitute their preferred anomaly detection algorithms and failure localization methods while maintaining the overall framework's effectiveness. This adaptability is particularly important as different organizations may have varying monitoring infrastructures and data collection practices.

While our evaluation utilized SocialNetwork and HotelReservation applications with 28 and 20 components respectively, we believe our approach scales effectively to larger environments. The architecture of LocaleXpert employs hierarchical filtering and modular analysis, which inherently distributes the computational load. This design enables the system to handle increased component counts without exponential growth in processing requirements through several key mechanisms. First, the initial Service/Instance Filter (Filter 1) reduces the data volume by focusing only on the relevant time window and affected services, effectively creating a manageable subset of the full monitoring data stream. Second, the subsequent Service/Instance Filter (Filter 2) further narrows the scope by leveraging the root cause service identified by the trace module, ensuring that only pertinent logs and metrics are analyzed in detail. This two-stage filtering approach means that even as the number of microservices grows from tens to hundreds, the actual data processed by each expert agent remains bounded by the localized failure scope rather than the entire system scale. Third, the modular design allows each data type (logs, metrics, traces) to be processed independently and in parallel, with each expert agent focusing solely on its specialized modality. This parallelization potential means that computational resources can be distributed across multiple processing units. Finally, the external failure localization modules are themselves designed for efficiency in large-scale systems, and LocaleXpert inherits their scalability properties while adding the interpretability layer. The improved performance observed in the more complex CCB Cloud dataset (40 instances) compared to GAIA (10 instances) suggests that the methodology becomes even more valuable as system complexity increases, as the hierarchical filtering becomes more effective at reducing the search space. However, we acknowledge that definitive scalability claims for environments with thousands of microservices (such as Azure-scale deployments) would require direct empirical validation, which represents an important direction for future work.

Furthermore, LocaleXpert demonstrates robust degradation capabilities when dealing with incomplete monitoring data - a common challenge in real-world deployments. As illustrated in our case study where trace data showed no anomalies, the system can still perform effective diagnosis by adaptively relying on available data modalities. The system achieves this through several design choices: First, each expert agent can operate independently, allowing the final diagnosis to be made even when certain data types are missing or unreliable. Second, the Service/Instance Filter's fallback mechanism automatically shifts focus to log analysis when trace-based filtering fails. Third, the Failure Localization Expert is told to synthesize findings across any combination of available modalities, rather than requiring all three types of data.

The actionability and practical impact of LocaleXpert's diagnoses are evaluated through our deployment in the AIOpsLab environment and feedback from domain experts. However, we acknowledge that full validation of actionability would ideally require comprehensive Mean Time To Resolution (MTTR) studies in live production environments, comparing incident resolution times with and without LocaleXpert.

Such controlled experiments in production systems were not feasible within the scope of this work due to the operational risks and resource constraints associated with deploying experimental systems in critical production environments. Future research should prioritize rigorous MTTR evaluations to quantify the end-to-end operational impact of LLM-based failure localization systems.

C. Privacy Considerations

Regarding data privacy in environments with restricted access policies, LocaleXpert's on-premise deployment capability allows the entire system to operate within the operator's security perimeter, avoiding transmission of sensitive data to external services. Model fine-tuning with sensitive data introduces additional risks, such as memorization and unintended leakage, which constitute an active research area with substantial recent progress. A detailed discussion is beyond the scope of this work. Importantly, LocaleXpert does not require fine-tuning: all experiments in this paper are conducted using off-the-shelf open-source models, and performance needs are met through prompt engineering and tool-augmented workflows alone. The system design also aligns with major regulatory principles such as GDPR, as operators maintain full control over data usage, retention, and auditing.

VII. Conclusion

This paper presents LocaleXpert, an innovative failure localization system for microservice maintenance that integrates LLM-based agents. Our system effectively handle failure localization tasks through components including the anomaly description module for processing multi-modal data, the external failure localization module for providing diagnostic context, and the interpretation module for guiding LLM reasoning. Experimental results show LocaleXpert achieves significantly higher accuracy and better interpretablility in failure localization compared to existing approaches.

References

[1] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang, "Automap: Diagnose your microservice-based web applications automatically," in *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Y. Huang, I. King, T. Liu, and M. van Steen, Eds. ACM / IW3C2, 2020, pp. 246–258.

[2] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Eadro: An end-to-end troubleshooting framework for microservices on multi-source data," in *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 2023, pp. 1750–1762.

[3] S. Zhang, P. Jin, Z. Lin, Y. Sun, B. Zhang, S. Xia, Z. Li, Z. Zhong, M. Ma, W. Jin, D. Zhang, Z. Zhu, and D. Pei, "Robust failure diagnosis of microservice system through multimodal data," *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 3851–3864, 2023.

[4] Y. Li, G. Yu, P. Chen, C. Zhang, and Z. Zheng, "Microsketch: Lightweight and adaptive sketch based performance issue detection and localization in microservice systems," in *Service-Oriented Computing - 20th International Conference, ICSOC 2022, Seville, Spain, November 29 - December 2, 2022, Proceedings*, ser. Lecture Notes in Computer Science, J. Troya, B. Medjahed, M. Piattini, L. Yao, P. Fernández, and A. Ruiz-Cortés, Eds., vol. 13740. Springer, 2022, pp. 219–236.

[5] G. Yu, P. Chen, H. Chen, Z. Guan, Z. Huang, L. Jing, T. Weng, X. Sun, and X. Li, "Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. ACM / IW3C2, 2021, pp. 3087–3098.

[6] Y. Zhang, Z. Guan, H. Qian, L. Xu, H. Liu, Q. Wen, L. Sun, J. Jiang, L. Fan, and M. Ke, "Cloudrca: A root cause analysis framework for cloud computing platforms," in *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, and H. Tong, Eds. ACM, 2021, pp. 4373–4382.

[7] Z. Li, N. Zhao, M. Li, X. Lu, L. Wang, D. Chang, X. Nie, L. Cao, W. Zhang, K. Sui, Y. Wang, X. Du, G. Duan, and D. Pei, "Actionable and interpretable fault localization for recurring failures in online service systems," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, A. Roychoudhury, C. Cadar, and M. Kim, Eds. ACM, 2022, pp. 996–1008.

[8] M. Li, Z. Li, K. Yin, X. Nie, W. Zhang, K. Sui, and D. Pei, "Causal inference-based root cause analysis for online service systems with intervention recognition," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, A. Zhang and H. Rangwala, Eds. ACM, 2022, pp. 3230–3240.

[9] M. Shetty, Y. Chen, G. Somashekar, M. Ma, Y. Simmhan, X. Zhang, J. Mace, D. Vandevoorde, P. Las-Casas, S. M. Gupta, S. Nath, C. Bansal, and S. Rajmohan, "Building AI agents for autonomous clouds: Challenges and design principles," in *Proceedings of the 2024 ACM Symposium on Cloud Computing, SoCC 2024, Redmond, WA, USA, November 20-22, 2024*. ACM, 2024, pp. 99–110.

[10] Z. Yu, M. Ma, C. Zhang, S. Qin, Y. Kang, C. Bansal, S. Rajmohan, Y. Dang, C. Pei, D. Pei, Q. Lin, and D. Zhang, "Monitorassistant: Simplifying cloud service monitoring via large language models," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, FSE 2024, Porto de Galinhas, Brazil, July 15-19, 2024*, M. d'Amorim, Ed. ACM, 2024, pp. 38–49.

[11] T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, and S. Rajmohan, "Recommending root-cause and mitigation steps for cloud incidents using large language models," in *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 2023, pp. 1737–1749.

[12] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C. Zhang, Q. Lin, S. Rajmohan, and D. Zhang, "Empowering practical root cause analysis by large language models for cloud incidents," *CoRR*, vol. abs/2305.15778, 2023.

[13] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, S. Yih, L. Zettlemoyer, and M. Lewis, "Incoder: A generative model for code infilling and synthesis," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[14] Z. Jiang, J. Liu, Z. Chen, Y. Li, J. Huang, Y. Huo, P. He, J. Gu, and M. R. Lyu, "Llmparser: A llm-based log parsing framework," *CoRR*, vol. abs/2310.01796, 2023.

[15] Y. Chen, M. Shetty, G. Somashekar, M. Ma, Y. Simmhan, J. Mace, C. Bansal, R. Wang, and S. Rajmohan, "Aiopslab: A holistic framework to evaluate ai agents for enabling autonomous clouds," 2024. [Online]. Available: https://www.microsoft.com/en-us/research/publication/aiopslab-a-holistic-framework-for-evaluating-ai-agents-for-enabling-autonomous-cloud

[16] P. Aggarwal, A. Gupta, P. Mohapatra, S. Nagar, A. Mandal, Q. Wang, and A. M. Paradkar, "Localization of operational faults in cloud applications by mining causal dependencies in logs using golden signals," in *Service-Oriented Computing - ICSOC 2020 Workshops - AIOps, CFTIC, STRAPS, AI-PA, AI-IOTS, and Satellite Events, Dubai, United Arab Emirates, December 14-17, 2020, Proceedings*, ser. Lecture Notes in Computer Science, H. Hacid, F. Outay, H. Paik, A. Alloum, M. Petrocchi, M. R. Bouadjenek, A. Beheshti, X. Liu, and A. Maaradji, Eds., vol. 12632. Springer, 2020, pp. 137–149.

[17] K. Julisch, "Clustering intrusion detection alarms to support root cause analysis," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 4, pp. 443–471, 2003.

[18] X. Zhang, Y. Xu, S. Qin, S. He, B. Qiao, Z. Li, H. Zhang, X. Li, Y. Dang, Q. Lin, M. Chintalapati, S. Rajmohan, and D. Zhang, "Onion: identifying incident-indicating logs for cloud systems," in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and*

Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021, D. Spinellis, G. Gousios, M. Chechik, and M. D. Penta, Eds. ACM, 2021, pp. 1253–1263.

[19] D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, and Z. Wu, "Microhecl: High-efficient root cause localization in large-scale microservice systems," in *43rd IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 2021, pp. 338–347.

[20] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, M. Dumas, D. Pfahl, S. Apel, and A. Russo, Eds. ACM, 2019, pp. 683–694.

[21] H. Nguyen, Z. Shen, Y. Tan, and X. Gu, "Fchain: Toward black-box online fault localization for cloud systems," in *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8-11 July, 2013, Philadelphia, Pennsylvania, USA*. IEEE Computer Society, 2013, pp. 21–30.

[22] P. Wang, J. Xu, M. Ma, W. Lin, D. Pan, Y. Wang, and P. Chen, "Cloudranger: Root cause identification for cloud native systems," in *18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2018, Washington, DC, USA, May 1-4, 2018*, E. El-Araby, D. K. Panda, S. Gesing, A. W. Apon, V. V. Kindratenko, M. Cafaro, and A. Cuzzocrea, Eds. IEEE Computer Society, 2018, pp. 492–502.

[23] M. Ma, W. Lin, D. Pan, and P. Wang, "Ms-rank: Multi-metric and self-adaptive root cause diagnosis for microservice applications," in *2019 IEEE International Conference on Web Services, ICWS 2019, Milan, Italy, July 8-13, 2019*, E. Bertino, C. K. Chang, P. Chen, E. Damiani, M. Goul, and K. Oyama, Eds. IEEE, 2019, pp. 60–67.

[24] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in *28th IEEE/ACM International Symposium on Quality of Service, IWQoS 2020, Hangzhou, China, June 15-17, 2020*. IEEE, 2020, pp. 1–10.

[25] Y. Sun, Z. Lin, B. Shi, S. Zhang, S. Ma, P. Jin, Z. Zhong, L. Pan, Y. Guo, and D. Pei, "Interpretable failure localization for microservice systems based on graph autoencoder," *ACM Transactions on Software Engineering and Methodology*, 2024.

[26] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang, and T. Gui, "The rise and potential of large language model based agents: A survey," *CoRR*, vol. abs/2309.07864, 2023.

[27] A. Ghafarollahi and M. J. Buehler, "Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning," *Advanced Materials*, vol. 37, no. 22, p. 2413523, 2025.

[28] ——, "Protagents: protein discovery via large language model multi-agent collaborations combining physics and machine learning," *Digital Discovery*, vol. 3, no. 7, pp. 1389–1409, 2024.

[29] J. Zhang, T. Mytkowicz, M. Kaufman, R. Piskac, and S. K. Lahiri, "Using pre-trained language models to resolve textual and semantic merge conflicts (experience paper)," in *ISSTA '22: 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, South Korea, July 18 - 22, 2022*, S. Ryu and Y. Smaragdakis, Eds. ACM, 2022, pp. 77–88.

[30] Z. Wang, Z. Liu, Y. Zhang, A. Zhong, L. Fan, L. Wu, and Q. Wen, "Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models," *CoRR*, vol. abs/2310.16340, 2023.

[31] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 56:1–56:33, 2022.

[32] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, "Automatic anomaly detection in the cloud via statistical learning," *CoRR*, vol. abs/1704.07706, 2017.

[33] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 2828–2837.

[34] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: unsupervised anomaly detection on multivariate time series," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 2020, pp. 3395–3404.

[35] P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, J. Yang, L. Mo, J. Zeng, W. Xue, and D. Pei, "Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks," in *31st IEEE International Symposium on Software Reliability Engineering, ISSRE 2020, Coimbra, Portugal, October 12-15, 2020*, M. Vieira, H. Madeira, N. Antunes, and Z. Zheng, Eds. IEEE, 2020, pp. 48–58.

[36] C. Wu, N. Zhao, L. Wang, X. Yang, S. Li, M. Zhang, X. Jin, X. Wen, X. Nie, W. Zhang, K. Sui, and D. Pei, "Identifying root-cause metrics for incident diagnosis in online service systems," in *32nd IEEE International Symposium on Software Reliability Engineering, ISSRE 2021, Wuhan, China, October 25-28, 2021*, Z. Jin, X. Li, J. Xiang, L. Mariani, T. Liu, X. Yu, and N. Ivaki, Eds. IEEE, 2021, pp. 91–102.

[37] C. Hou, T. Jia, Y. Wu, Y. Li, and J. Han, "Diagnosing performance issues in microservices with heterogeneous data source," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), New York City, NY, USA, September 30 - Oct. 3, 2021*. IEEE, 2021, pp. 493–500.

[38] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[39] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C. Zhang, Q. Lin, S. Rajmohan, D. Zhang, and T. Xu, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024*. ACM, 2024, pp. 674–688.

[40] C. Lin and F. J. Och, "ORANGE: a method for evaluating automatic evaluation metrics for machine translation," in *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*, 2004.

[41] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Annual Meeting of the Association for Computational Linguistics*, 2004.

[42] L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[43] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun, "Toolllm: Facilitating large language models to master 16000+ real-world apis," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.