RCInvestigator: Towards Better Investigation of Anomaly Root Causes in Cloud Computing Systems

Shuhan Liu, Yunfan Zhou, Jue Zhang, Shandan Zhou, Weiwei Cui,
Qingwei Lin, Thomas Moscibroda, Haidong Zhang, Di Weng, Yingcai Wu

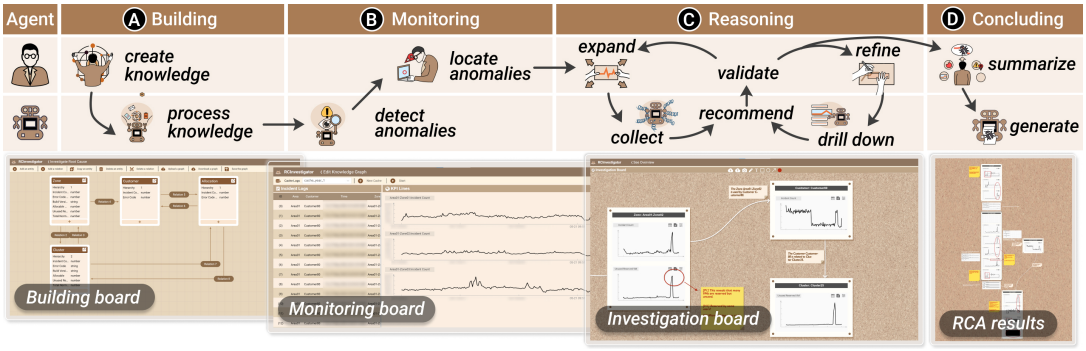

Fig. 1: This is a new human-machine-collaborative workflow of root cause analysis in cloud computing systems and an interactive visual analysis system named RCInvestigator. The workflow includes four stages: (A) building, (B) monitoring, (C) reasoning, and (D) concluding. Two agents collaborate via RCInvestigator, consisting of building, monitoring, and investigation boards.

Abstract—Root cause analysis (RCA) is critical for maintaining the availability and efficiency of cloud computing systems. However, identifying root causes from the large-scale, high-dimensional monitoring data generated by these complex environments is a significant challenge. Current approaches often rely on time-consuming manual analysis to ensure flexibility and reliability, while recent automated methods lack the crucial insights provided by domain experts. To bridge this gap, we propose RCInvestigator, a visual analytics system that facilitates interactive root cause investigation by establishing a tight collaboration between human experts and machine analysis. Our approach addresses three key challenges: a) modeling databases for the root cause investigation, b) inferring root causes from large-scale time series, and c) building comprehensible investigation results. We demonstrate the effectiveness and utility of RCInvestigator through two real-world case studies, which received positive feedback from domain experts.

Index Terms—Complex system visual diagnosis, root cause analysis, time-oriented data, cloud computing systems

1 Introduction

Cloud computing has emerged as a pervasive infrastructure technology, profoundly influencing a broad spectrum of application domains [19, 22]. As the significance of cloud computing systems escalates, its anomaly management becomes imperative since irregularities not only pose potential security risks and performance issues [40] but can also lead to substantial economic losses [55, 56]. Despite this urgency, root cause analysis (RCA), a critical aspect of anomaly management, continues to pose a significant challenge.

RCA is a reactive diagnostic process used to determine ***what*** caused a system anomaly and ***how*** it propagated [36]. Its goal is to guide engineers in rectifying the immediate fault and preventing future recurrences [20]. Unlike monitoring (e.g., Grafana, Splunk) or anomaly detection, which flag a *symptom* (*a surge in failed user requests*), RCA investigates the underlying *cause* (*cluster resource exhaustion*). This is challenging because a single symptom can stem from disparate root causes. For example, *a network misconfiguration* can also cause a failed spike, but requires a distinct mitigation. Pinpointing the true causal pathway is therefore a non-trivial task demanding domain expertise, especially given the architectural complexity and vast data volumes.

Numerous researchers are endeavoring to resolve the RCA problem. With the volume of data becoming extremely large, many methods leverage machine learning techniques, such as graph mining [4, 9] and logistic regression [8, 39], in identifying potential causes, effectively answering the question of ***what*** causes a given anomaly. However, machine-learning-based approaches are limited in explaining ***how*** the root cause exerts its influence due to its black-box nature. The sparsity of *(anomaly, root cause)* pairs also significantly limits the learning-based methods. Consequently, the prevalent RCA practice employed in the industry is predominantly dependent on labor-intensive manual processes. Engineers typically run a series of database queries to gather relevant factors, analyze potential root causes, and deduce conclusive results. Such an analytical procedure is inherently time-intensive, frequently spanning several hours to multiple days to finalize.

The limited efficiency of manual RCA and the poor interpretability of "black-box" learning methods raise an urgent need for an approach that synergizes human expertise with automated analysis. Visual analytics is particularly well-suited to bridge this gap by enabling powerful human-in-the-loop investigation. Drawing on insights from existing visual analytics (VA) studies on cloud computing [32, 62], we propose an approach that synergizes human-machine collaboration. However, since VA for RCA is emerging, a successful solution must be grounded in the practical realities. To identify these, we conducted a six-month study with domain experts following Sedlmair et al.'s methodology [45]. Based on empirical findings, we identified three core design challenges.

- *S. Liu, Y. Zhou, and Y. Wu are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China. E-mail: {shliu, yunfzhou, ycwu}@zju.edu.cn.*
- *D. Weng is with School of Software Technology, Zhejiang University. E-mail: {dweng}@zju.edu.cn. D. Weng is the corresponding author.*
- *J. Zhang, W. Cui, Q. Lin, H. Zhang are with Microsoft, Beijing, China. E-mail: {jue.zhang, weiwei.cui, qlin, haidong.zhang}@microsoft.com*
- *S. Zhou and T. Moscibroda are with Microsoft, Redmond, United States. E-mail: {shazho, moscitho}@microsoft.com*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

C1:Modeling complex factor relations for root cause investigation. Investigating cloud anomalies requires modeling relationships between factors dispersed across multiple platform components. This is challenging not only due to the scale and complexity of the data but also because the nature of these relationships is dynamic. For example, the relation between *a user* and *a virtual machine* could be one of *allocation* for a capacity anomaly or *access* for a security anomaly. Therefore, an interactive modeling approach is needed to incorporate expert knowledge and adapt to different analytical intents.

C2:Inferring obscure root causes from large-scale time series. Anomalies often manifest as spikes or drops in a key performance indicator (KPI) time series. Identifying the root cause requires correlating this KPI with temporal changes in other factors. This presents a dual challenge: analysts must first identify the few relevant factors from a massive dataset, like finding needles in a haystack; and then conduct a comprehensive analysis to understand their relationships and influence. Therefore, the proposed approach should not only be capable of detecting relevant factors but also providing reasoning support based on massive time series data.

C3:Building intuitive presentations of intertwined investigation findings. Creating intuitive presentations of investigation findings is essential for enabling timely action and transforming them into reusable knowledge, transcending the current practice of copy-pasting raw query results. However, the fundamental challenge lies in synthesizing this complex and often non-linear investigative process into a single, coherent representation. Distilling the intricate web of influence links, analytical steps, and expert insights into an intuitive and communicable form, without losing critical context, is a formidable barrier to transforming raw investigation findings into actionable, reusable knowledge.

In this study, we collaborate closely with the experts from a large cloud service provider and propose an interactive investigation framework based on human-machine collaboration for RCA. Our framework comprises four stages: building, monitoring, reasoning, and concluding, and respectively delegates different tasks to the human and machine agent (see Fig. 1), whereas the machine agent is responsible for laborious data collection tasks and the human agent is responsible for decision-making. First, knowledge graphs are employed to model the complex factor relationships across multiple data sources. Then, we design intuitive visualizations and a layout algorithm to organize the factors reasonably for visual analysis. Finally, we summarize two types of reasoning interactions and design corresponding reasoning models to enable steerable reasoning in finding the root causes of anomalies.

Our main contributions are listed as follows.

- We identify the tasks for analyzing anomaly root causes in cloud computing systems and formulate a human-machine-collaborative root cause investigation framework.
- We propose a novel knowledge-graph-based interactive approach for root cause analysis and develop RCInvestigator, an interactive system that supports steerable reasoning of the root causes.
- We evaluate RCInvestigator with two cases based on a real-world dataset and collect feedback from experts.

2 Related Work

2.1 Root Cause Analysis

Root cause analysis is a critical task in anomaly management, with extensive literature spanning both general surveys [20, 49] and specific domain reviews (e.g., software [57], industry [16]). Our work focuses specifically on RCA within IT operations for large-scale cloud environments. According to Natoro et al. [36], we discuss two categories: root cause localization (*what*) and root cause diagnosis (*how*).

Localization studies aim to pinpoint the system components harboring the root cause. These approaches primarily fall into two categories: machine-learning-based approaches [4, 28] that build inference models, and search-based strategies [30, 34, 35, 46] that leverage the correlation of Key Performance Indicators (KPIs) and employ search algorithms along with pruning techniques to expedite the localization process.

Diagnosis studies seek to elucidate the pathways that cause precipitate anomalies. Some researchers employ machine learning techniques [9–11, 27] to categorize and explain common root cause mechanisms. However, the effectiveness of these models is often limited by data sparsity and a predefined set of category labels. Additionally, other researchers trace anomalies on the established graph to facilitate the diagnostic process. The built graphs include topology graphs [23], causality graphs [9, 10], and knowledge graphs [67]. The walking strategies can be Breadth-First-Search [25], Markov analysis [44], KPI-correlation-oriented [51], and random walk [60].

While automated RCA offers efficiency, it often lacks the explainability and controllability essential for real-world applications, and current research typically focuses on specific components. RCInvestigator introduces a human-machine collaborative framework that leverages machine-scale data processing alongside human reasoning and control. This approach aims to not only locate root causes across system levels but also explain their impact.

2.2 Time-oriented Data Visualization

Numerous studies discuss temporal data visualizations from different perspectives, such as general time-oriented data [1, 2], discrete sequential events [6, 18], and continuous time series [15]. Data from cloud environments are primarily linear, consisting of event intervals and time-stamped states. As understanding the relations between these events and states is fundamental to RCA, we mainly discuss techniques for visualizing event sequences, time series, and their interrelations.

For visualizing time-oriented data, techniques can be categorized into three types: chart-, timeline-, and tree-based. Chart-based methods (e.g., RetainVIS [24]) visualize the distribution of event or point instances by basic bar charts or scatter plots. Timeline-based techniques (e.g., PlanningVis [50], VIVA [7]) present the order of events and values consistently along a timeline and emphasize the temporal features. Tree-based methods usually display the aggregation patterns (e.g., VisRuption [42]) or the hierarchical patterns (e.g., VizTree [26]).

For visualizing relations and evolutions, there are general techniques and special designs. General high-dimensional data visualizations (e.g., PCP [52]) display relations between different attributes but fail to describe temporal features. Special-design plots (e.g, [3, 37, 58]) intuitively reveal the evolution but are for events or time series specifically.

Based on extensive literature review, RCInvestigator adopts the most easy-to-understand visualizations: Gantt charts for event sequences and line charts for time series. We further design visualizations and frameworks that intuitively highlight the relations among event sequences and time series, supporting RCA tasks.

2.3 Complex System Visual Diagnosis

Complex system visual diagnosis integrates intuitive visual representations, flexible interactions, and effective models to enable a comprehensive understanding of failures and large-scale data. It is effective in solving complex systems problems, covering various domains, like industry [61, 68], urban computing [13], and dynamic network analysis [5, 48, 54]. We discuss aspects of pipeline risk management via the lens of a general workflow [33], specifically addressing monitoring, anomaly detection, and root cause analysis.

As for **monitoring**, the main challenge of visualizing such data is to organize large-scale and high-dimension data reasonably. The commonly used visualization techniques include static and dynamic ones. Many static approaches not only utilize cluster algorithms [47] but also adopt high-dimension visual representations, such as small multiples [14, 32]. Dynamic approaches introduce interactions (e.g., Traveler [43], XR [59]) and animations (fish eyes [64], dynamic PCP) in visualizations and enable level-of-detail displaying, making the monitoring process more flexible than static ones.

Visual analysis has been proven effective for **anomaly detection** across multiple domains [65, 69]. In the industrial sector, ViDX [63] highlights anomalies using Marey charts, while ECoalVis [29] enables interactive extraction of anomalies with trend events. In software analysis, CloudDet [62] employs clustering algorithms and glyph-based visualization for multidimensional anomaly detection, while GRANO [53] locates anomalous components by a knowledge graph. ViSRE [21] uses causal inference to predict anomalies proactively.

Despite visual analysis's utility in failure management, interactive **root cause analysis** for complex systems remains underexplored. Compared with anomaly detection, RCA not only requires for locating the cause but also emphasizing explaining how the cause intrigue anomalies. Therefore, we propose and develop the RCInvestigator framework.

3 Pain Points Analysis

In our study, the method development involved an iterative design process that was guided by Sedlmair et al.'s design study methodology [45]. Such a design process unfolded over six months and involved three steps:(1) We conducted a comprehensive review of existing commercial tools and visual analytics systems to understand the current research landscape; (2) Over the following two months, we engaged in multiple interview sessions with four domain experts (EA-ED) to obtain their nuanced insights and summarized pain points and user requirements in the RCA processes; (3) In the final three months, we engaged in an iterative design process, completing three rounds of refinement. Starting with a general design, we incrementally optimized it through four versions, focusing on visualization simplification, interaction exploration, plot layout, metaphorical enhancement, and knowledge modeling. This process involved using scenario-based discussions with competing mockups to methodically resolve conflicting expert feedback. Our goal was to ensure the final system design was user-centric and aligned with expert feedback. Details of iterations and expert comments are presented in the supplementary material. All of the experts were employed by a leading cloud computing provider. EA (female) and EB (male) were data scientists with years of experience in conducting cloud platform incident analyses, while EC (male) and ED (male) were senior researchers specialized in AIOps, facilitating IT operations in cloud computing systems with AI-based approaches.

3.1 Motivating Usage Scenario

During months of collaboration with experts, we have gained insights into the process of analyzing root causes in cloud computing systems. In this section, we illustrate the limitations of existing workflows (Fig. 2) via a representative scenario. Consider an analyst, Sabrina, tasked with prioritizing unresolved incidents from an office automation system.

First, Sabrina reviews the incident alert list. She struggles to take the first analysis step because there are so many KPIs conveying incidents. Typically, she generates some line plots to visualize data center-level metrics, aiding her selection of an entry point. However, this requires her to first merge similar alerts and write individual query scripts for each. Sometimes she can copy and paste existing query scripts and make a little modification if the incident only involves frequently investigated components. Otherwise, she spends a long time writing new KPI queries. Sabrina visualizes each queried metric and selects one to start the investigation (**P1.1:** ***cumbersome alert consolidation***).

After narrowing her focus to several specific metric spikes, Sabrina starts hunting for the cause, but the fragmented nature of the system complicates her investigation. To trace the anomaly, Sabrina uses her experience to write custom database queries targeting common clues (e.g., cluster load, user traffic spikes). She runs these queries, studies the results, and forms new guesses to test (**P1.2:** ***cumbersome clue collection***). However, each query returns hundreds of irrelevant results, forcing her to repeatedly refine her filters, a slow and risky process because she is not sure which variables truly matter and should make decisions carefully based on her knowledge (**P2.1:** ***heavy mental strain from numerous ambiguous variables***). Worse, some clues confuse her (e.g., unusual API error codes), as her knowledge of the whole complex computing system architecture is limited. When this happens, she emails other teams (e.g., network ops, developers) to ask for help, organizes discussions, and starts new queries based on their feedback (**P2.2:** ***heavy mental strain from knowledge gaps in vast architectures***). Additionally, Sabrina often references similar historical investigation results from other analysts. However, since these records typically capture only outcomes rather than the complete investigative process, she spends additional time reconstructing context to proceed effectively (**P3.1:***hard-to-share insights missing context*).

Finally, Sabrina compiles an RCA summary by sifting through scattered query results, emails, and interview logs from the investigation. As it is difficult to trace back all key clues, she only manually picks out necessary results and organizes them into a shared forum with notes explaining their relevance (**P3.2:***hard-to-form context-aware investigation results*).

We identified three pain points in such a workflow.

- **P1.** The collection process of potential clues is cumbersome. Analysts need to write query scripts manually to retrieve relevant data from databases, but it is hard to have a comprehensive understanding of such complex and large-scale databases. Furthermore, there is much repetitive work like writing numerous similar and lengthy scripts, making the process time-consuming and labor-intensive.
- **P2.** The reasoning process causes a heavy mental burden. Analysts must delve into the collected clues and compare them one by one to make inferences, which incurs a high memory cost. Moreover, existing representations of clues are mainly raw tabular data, which is not intuitive, causing additional cognitive difficulty.
- **P3.** The investigation results are hard to share. Current analysis summaries are mainly composed of scattered information like text and query results, which fails to present the analysts' reasoning logic in an organized and intuitive manner. Additionally, the arbitrary format of the results makes it difficult to share the knowledge.

We distilled pain points into three corresponding design challenges (**C1-C3**), which were introduced from design-level perspective in Sec 1.

4 Workflow and Task Abstraction

To address the pain points (**P1-P3**) and identified design challenges (**C1-C3**), we designed a human-machine collaborative workflow.

4.1 RCA Workflow

Our main ideas include:(1) delegating repetitive collection tasks and computing tasks to the machine agent (**P1**); (2) abstracting reasoning logics into direct interactions and intuitively visualizing the reasoning process (**P2**); (3) building all these in a structured format to facilitate sharing (**P3**). The new workflow (Fig. 1) has one preparation stage (*building*), three investigation stages (*monitoring*, *reasoning*, and *concluding*) and two agents (*human agent* A_h and *machine agent* A_m). This design of workflow is grounded in our empirical findings and prior literature. While the typical Monitoring, Reasoning, and Concluding stages are from literature [36], the Building stage and the human-machine task division are derived directly from our expert feedback. The workflow systematically addresses our three design challenges (**C1-C3**):

- **Building (C1)**: a consensus that describes how two agents collaborate should be defined. It guides how to share knowledge of databases, anomalies, and root causes. A_h dominates the definition of consensus to ensure its interpretability and reliability, while A_m follows it to adapt data loading.
- **Monitoring (C2)**: A_m *detects* potential anomalies and alarms users, and then A_h *locates* a detailed anomaly, whose root cause users want to investigate.
- **Reasoning (C2)** is an iterative process. First, A_h proposes hypotheses based on the selected anomaly, and then A_m executes and *recommends* relevant clues from the dataset, which A_h *validates*

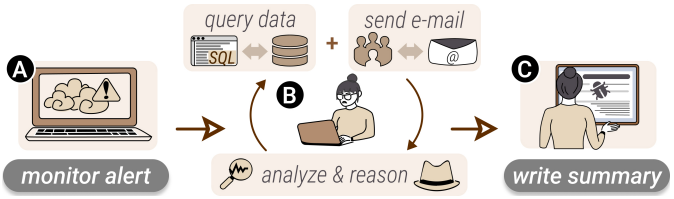

Fig. 2: Three stages of the existing pipeline: (A) Analysts screen the alert list to select targets. (B) Analysts write scripts to query data, e-mail relevant teams for additional information, and then reason potential causes. They iterate until root cause found. (C) Analysts write a summary.

to confirm as evidence. New evidence can inspire the formation of new hypotheses, repeating the cycle until the root cause is found. There are two kinds of hypotheses: When A_h *expands* a clue, A_m *collects* as many relevant clues as possible to generalize the analysis. When A_h *refines* a clue, A_m *drills down* the clue and eliminates unnecessary factors to detail the investigation.

- **Concluding (C3)**: A_h *summarizes* final root causes, and A_m *generates* structured and visual format of conclusions for sharing.

In the new workflow, Sabrina first loads and supplements the knowledge base. With institutional expertise consolidated in the base, she retrieves critical clues more independently. She then reviews a prioritized alert list augmented with contextual metric visualizations, enabling her to swiftly identify urgent anomalies. During reasoning, A_m autonomously ranks highly relevant clues, eliminating the need for manual scripting. This allows Sabrina to focus entirely on decision-making. Besides, A_m merges and intuitively organizes clues, so Sabrina can better control the investigation's direction and depth. Finally, the process is automatically documented in a structured, visual format.

4.2 Data and Concepts

This study is based on the data obtained from the cloud service provider where the experts were employed. Data can span multiple sources, including system-level, network-level, and application-level monitoring.

In the building stage, two anomaly types are involved:(1) *Incident logs* record a log that a user failed to request resources. Each incident log is a tuple of number and string; (2) *Key performance indicators* record states of components and are numerical time series. Both anomalies are detected by platform existing monitors.

In the reasoning stage, there are two clue types: (1) Time series: record numerical changes in components and performance metrics; (2) Event sequences: record categorical changes in options. Validated clues are regarded as pieces of evidence. Data types include *number* (a time series), *string* (an event sequence), *set* (multiple event sequences), and *bag* (multiple time series).

4.3 Design Tasks

Based on three design challenges and the four-stage workflow, we detailed user requirements into five design tasks.

T1. (Building) Build persistent investigation knowledge interactively . The system must provide an interactive way for analysts to build, edit, and persist their investigative knowledge about system components and their relationships. This structured knowledge serves as the foundation for the machine's automated clue collection.

T2. (Monitoring) Obtain an overview of cloud computing anomalies. The system must provide an interactive way for analysts to build, edit, and persist their investigative knowledge about system components and their relationships. This structured knowledge serves as the foundation for the machine's automated clue collection.

T3. (Reasoning) Extend analysis scope based on recommendations. This task focuses on the human agent's role in reasoning. The system must present machine-recommended clues in an interpretable manner, allowing the analyst to validate them as evidence. Based on this evidence, the analyst must be able to easily form new hypotheses and direct the next step of the investigation.

T4. (Reasoning) Explore time-oriented data based on investigation knowledge. This task defines the machine agent's role. Guided by the analyst's current hypothesis and the knowledge model, the system must automatically search the vast dataset to collect and recommend the most relevant causal clues. It should rank these clues based on their correlation to the anomaly to intelligently prune the search space.

T5. (Concluding) Share intuitive investigation results. The system must facilitate the creation of an intuitive and expressive summary of the entire investigation. This involves capturing the final root cause, key evidence, and reasoning steps in a structured format for sharing and future reference.

5 RCInvestigator

Based on the workflow and tasks, we proposed RCInvestigator, an interactive approach for human-machine collaborative root cause analysis in cloud computing systems. The system features a multi-page frontend, a backend model, and a data store. The Vue [66] frontend consists of three distinct pages corresponding to the workflow stages: Building, Monitoring, and Investigation Board. The Flask [38] backend handles knowledge processing, anomaly retrieval, clue recommendation, and summarization The data store connects to the Kusto [31] databases and manages data caches to ensure interactive performance. The following subsections will introduce the system design for four stages.

5.1 Build Investigation Knowledge

To support the creation of persistent investigation knowledge (**T1**), RCInvestigator utilizes a knowledge graph due to its flexibility and expandability. This section details the knowledge graph model and the design of the corresponding Building Board used to construct it.

5.1.1 Model investigation knowledge graph

A knowledge graph is typically defined by three components: entities, relations, and attributes. To model existing investigation knowledge into a knowledge graph, we map two categories of investigation knowledge to these components:(1) Cause clues (*what*): anomalous attributes that indicate potential root causes; (2) Reasoning logic (*how*): factual rules that guide how to collect cause clues.

Cause clues. After observing many root cause examples, we found that a cause clue can be defined as an observation on a tuple of (entity, attribute, time range). For example, a root cause like *"The nodes of cluster A were exhausted during 2:00 to 3:00 so anomalies happen"* can be simplified to the tuple *(cluster A, node count, 2:00-3:00)*. Because analysts typically investigate a fixed time range preceding an anomaly, this model can be further simplified to *(entity, attribute)*. However, there are a large number of entities in cloud platforms, and establishing a knowledge graph for each entity would be extremely costly. Therefore, we abstract specific entities (*cluster A*) into entity concepts (*clusters*). This creates a reusable knowledge graph blueprint, which significantly reduces storage and modeling costs.

Formally, an entity concept e_i belongs to the set of all concepts E and has a corresponding set of attributes $A_i = \{a_{ip}\}$. A specific cause clue is therefore a tuple (e_{iq}, a_{ip}, t), where e_{iq} is an individual entity instance of concept e_i observed at a time range t. To further refine the analysis, our model supports filtering on attributes; by default, attributes with a string data type are treated as categorical filters.

Reasoning logic reveals the factual connections between cause clues. For example, the fact that *"a cluster belongs to a zone"* guides an analyst to collect cause clues from the corresponding zone when a cluster-level anomaly occurs. This reasoning logic is captured as a relation in the knowledge graph, with the example being recorded as (*cluster, belong, zone*). Formally, we define a set of relations $R = \{r_k\}$, and a set of facts $F = \{f_k\}$, where each fact f_k is a tuple (e_i, r_k, e_j) that relation r_k connects two entities e_i and e_j. The complete investigation knowledge graph is thus represented as $G = (E, R, F)$.

Justification. Many alternative methods can support the representation of investigation knowledge. They span a spectrum from highly structured (e.g., annotated database schemas) to fully unstructured (e.g., natural language notes). We chose a knowledge graph blueprint because it avoids the rigidity and poor scalability of schemas while providing more control and clarity than free-form text. This approach yields a flexible, governable, and lightweight representation of knowledge.

5.1.2 Building board

From the knowledge model, we designed the building board (Fig. 3), an interactive interface for intuitively constructing and editing the knowledge graph. The design is based on a card-based visual metaphor.

Entities and attributes (Fig. 3A). Each card displays the entity's name and a table of its attributes, specifying each attribute's name and data type (see Sec. 4.2). The interface supports direct manipulation, allowing users to add new attributes or perform in-place editing of existing ones. Furthermore, users can define a Kusto query template [31] for each entity to specify its data retrieval logic (Fig. 3C).

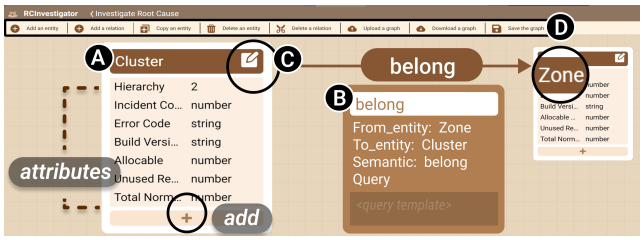

Fig. 3: The building board, showing the creation of *'cluster belongs to a zone'*. (A) Entity card (attributes/query). (B) Relation card (semantic). (C) Query editor. (D) Toolbar.

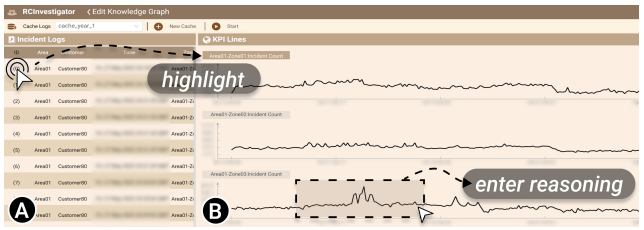

Fig. 4: The monitoring board, featuring (A) an incident log panel and (B) a KPI panel with corresponding line charts for key metrics.

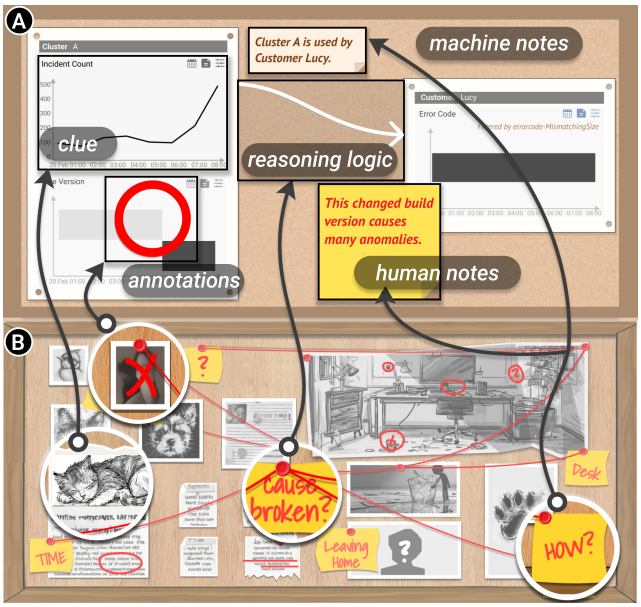

Fig. 5: This shows the investigation element design. There are four types of typical elements: clues, reasoning logic, annotations, and notes.

Relations (Fig. 3B). Relations are visualized as labeled directed edges linking two entity cards. Clicking a relation's label discloses a relation card, which contains detailed properties. The relation's semantic label and its corresponding query template are directly editable.

Interactions. A toolbar provides essential graph management functions (Fig. 3D). Beyond creating and editing entities and relations, it allows users to *Upload* an existing knowledge graph or *Download* the current one in JSON format. Clicking *Save* commits the graph to the monitoring board. With this design, a graph is constructed once and then reused or incrementally updated for subsequent analyses.

The machine agent plays an interactive validation partner. After a user defines entities, attributes, and relations, it actively validates this knowledge against the data sources. For example, if the corresponding query template does not cover an attribute defined on an entity card, an alert will be raised. Similarly, if a defined relation does not map to valid, joinable columns in the dataset, it will warn the user.

Justification. We considered a spectrum of knowledge authoring paradigms, from fully automated generation to an analyst-driven approach. While automated knowledge extraction offers speed, it fails on complex operational data where variable aliases create massive, unmanageable graphs. We thus adopt an analyst-driven approach, granting users full authorship to ensure the graph is reliable, interpretable, and accurately grounded in its data sources.

5.2 Monitor Anomalies

The overview provides a summary of system anomalies by coordinating incident logs with Key Performance Indicator (KPI) visualizations (**T2**). It comprises two main components: an incident log panel and a KPI panel. To ensure interactive performance when handling large-scale database queries, the system relies on a data caching mechanism.

The incident log panel (Fig. 4A). This presents incident logs from a selected time period in a tabular format. Each row corresponds to a single incident, with columns showing its specific attributes, providing essential context for an investigation.

The KPI panel (Fig. 4B). This panel visualizes important KPIs selected by users. To ensure data fidelity, KPIs inherit their types directly from the source system. We visualize these KPIs using synchronized line charts for continuous data and Gantt charts for discrete events, allowing analysts to correlate performance.

Interactions. The two panels are tightly coordinated. Clicking an incident in the log highlights the corresponding metric in the KPI panel. An analyst can then brush a time range on a KPI chart to select an anomalous pattern. This selection automatically transitions the user to the investigation board for detailed reasoning. Additionally, an "Edit Knowledge Graph" button allows users to refine the underlying semantic model without disrupting the analytical workflow.

The machine agent is responsible for automated anomaly detection. Our system does not propose a new anomaly detection algorithm, as these are well-studied and typically pre-deployed in cloud environments. Instead, it automatically queries the platform's monitors to retrieve the list of detected incident logs as well as pre-defined KPIs.

Justification. While many techniques exist for visualizing time-series and event data [1, 18], our initial design using streamgraphs produced severe visual clutter due to complex temporal dependencies. To prioritize analytical focus and clarity, we adopted the widely interpretable formats of line charts and Gantt charts.

5.3 Investigate Root Causes

The nvestigation board is for collaborative root cause reasoning, integrating our reasoning model with interactive visualizations (**T3**, **T4**).

Visual design. The investigation board not only features investigation elements, including cause clues and reasoning logic, but also serves as an interactive medium for human-machine collaboration. Our design provides a spatial canvas where analysts organize and connect evidence to form a cohesive understanding (Fig. 5A); this organization of clues is visually represented using a physical detective board metaphor (Fig. 5B). Each clue card visualizes a piece of evidence as a small chart. To handle diverse data, we use line plots for numerical time-series (number, bag types) and Gantt charts for event sequences (string, set types). All attributes are organized into different entity cards. Every entity card has the title "Entity Concept: Entity", like *"Area: Asia"*. The reasoning process connecting entities is visualized as directed arrows, which can be annotated with notes from either the analyst or the system to explain the logical steps.

Hypothesis interaction overview. RCInvestigator supports three hypothesis interactions: *expanding* and *refining* for clues; *annotating* for reasoning logic. ***Expanding*** broadens the investigation by collecting a wider set of alternative or more generalized clues. ***Refining*** drills down into a specific clue to reveal more granular details. ***Annotating*** allows analysts and the system to collaboratively add high-level interpretations of visual patterns. Next, we will introduce interactions and our model.

5.3.1 Expand Relevant Clues

We present five expanding hypothesis interactions, powered by an expanding model. We also introduce a dynamic layout algorithm that organizes clues to preserve semantic meaning and reduce visual clutter.

Interactions and visual designs. We offer five ***structured*** expansion interactions for exploring clues: *upward*, *downward*, *leftward*, *rightward*, and *inward*. These are derived from our knowledge model, where each clue is a tuple defined as $c_{i_p} = (e_i, a_{i_p})$. To ensure a systematic exploration, each interaction is designed to modify only one dimension of this tuple. Specifically, the inward interaction expands on the attribute, while the other four expand on the entity.

Upward expansion (f_U) facilitates root cause generalization by collecting clues from an entity at a higher level in the knowledge hierarchy. For example, an analyst can expand from a specific cluster to its parent zone. This interaction helps in assessing the broader context and potential scope of an incident.

Downward expansion (f_D) enables specialization by allowing an analyst to "drill down" to entities at a lower level in the knowledge hierarchy. For example, after observing a pattern in a zone, a user can expand downward to examine a specific cluster. This interaction is crucial for differentiating between distinct root causes that may manifest as a single, high-level symptom.

Rightward expansion (f_R) broadens an investigation by traversing a relationship to an entity of a different type. For example, an analyst investigating a customer entity can expand to that customer's associated orders. This helps reveal potential causal links by exploring evidence across related but distinct categories.

Leftward expansion (f_L) supports analogical reasoning by collecting clues from sibling entities, that is, different instances of the same entity concept. For example, after identifying a root cause affecting clusters in the US, an analyst can expand to find corresponding clues from clusters in Canada, accelerating the discovery of similar incidents. This interaction is designed for comparative analysis.

Inward expansion (f_I) allows an analyst to pivot from one attribute of an entity to another. For example, after examining a cluster's stability, a user can expand inward to correlate its utilization metrics. This multi-faceted view helps uncover complex root causes that arise from the interplay of different attributes within the same entity.

The expanding model automates the discovery of relevant clues by performing a best-first search over the knowledge graph. It is designed to automatically identify which of the countless potential clues (e.g., *Customers, Orders*) are most related to a given anomaly (e.g., in a *Zone*) . Our model is designed to answer this question automatically. The core of our model is a relevance metric, $R(c_1, c_2)$, based on the insight that related clues will ***"change"*** at roughly the same time. To capture this, the model first detects significant change points in each clue's time series using Bayesian Change Point Detection [41]. We then define a distance metric $d(S,T)$ Eq. 1, measuring the average temporal offset between the change points of two clues.

$$d(S,T) = \sum_{x \in S, y(x) \in T} |x - y(x)| \tag{1}$$

S and T are sets of change points for two clues, while $y(x)$ is the nearest change point in T for x in S. This distance is then normalized into a relevance score as Eq. 2, ranging from 0 to 1, where a score of 1 indicates perfect temporal alignment of their change points.

$$R(c_1,c_2) = 1 - \frac{1}{2N}\left(\frac{d(S_1,S_2)}{|S_1|} + \frac{d(S_2,S_1)}{|S_2|}\right) \tag{2}$$

, where N means the number of change points. The relevance score powers two strategies: (1) Inward expansion finds relevant attributes of the same entity. The model queries all attributes of the current entity and ranks them by their relevance score to the starting clue, returning the top-k results. (2) Other expansion finds relevant entities by traversing the graph. The model performs an iterative, best-first search. Explore one-hop neighbors, calculate the relevance of their clues, and maintain a ranked list of the best clues found so far. In Fig. 7, *Cluster*, *Customer*, and *Area* are one-hop neighbors for *Zone*. The search then expands recursively from the entities that produced the highest-relevance clues, pruning paths that do not improve the results. The process ends when no new high-relevance clues are found or a time limit (2s) is reached.

Layout. To organize the investigation board and preserve the semantic meaning of the reasoning directions, we designed a two-step hierarchical layout algorithm based on DAGre [12]. First, all entity cards connected by hierarchical (upward or downward) links are partitioned into groups. Within each group, DAGre computes a vertical layout, establishing a local coordinate (x_1,y_1) for each card. Next, each group is treated as a single node. These group nodes are then arranged horizontally across the canvas, establishing a group-level coordinate (x_2,y_2). The final position of each entity card is the sum of its local and group coordinates: $(x_1 + x_2, y_1 + y_2)$.

Parameters. Our model's parameters are set by domain-specific heuristics and principles for interactivity. We set the BCPD [41] hazard parameter lam to 360 ($24 \times 30/2$). This is a domain heuristic for our hourly data, reflecting an expert expectation of < 2 significant monthly updates (e.g., version pushes). We set $k = 3$ as an initial UI default for inward expansion to manage cognitive load. It is not an algorithmic filter and all attributes remain accessible. The 2s timeout is a strict HCI requirement to ensure interactivity and prevent the 10s+ delays from existing tools that break an analyst's concentration.

Justification. Our design addresses a key gap in existing methods for clue analysis. Our literature review and expert observations revealed three prevalent clue-collection methods: enumeration-based (KPI correlation analysis), search-based (graph traversal), and learning-based methods. The other two lack semantic organization and controllability, while learning-based methods face data scarcity issues. We designed the directional interactions to address the lack of semantic control in typical search interfaces. Each direction explicitly maps to a fundamental analytical primitive: upward/downward for generalization and specialization, leftward for comparison, and rightward for correlation. This transforms clue collection from a "black box" query into a human-steered dialogue with the reasoning model. Similarly, our hybrid (ranked + clustered) layout was developed because our previous, purely ranked-list design failed to provide necessary reasoning context.

5.3.2 Refine Current Evidence

Initial clues must be refined because a single anomaly can mask multiple distinct problems. For example, a cascading failure might conflate a primary *OS error* with secondary *network* congestion. Furthermore, a root cause may only manifest in a specific data subset, such as on resources of a particular *version* or *size*. To address this, our system provides a clue refining capability. We support analysts with an interactive filtering interface for manual exploration, coupled with a refining model that automatically mines and recommends high-value filters to accelerate the discovery of these hidden patterns.

The filter card **(Fig. 6)** is a dedicated interface for both manual and automated clue refinement, consisting of a selection panel (Fig. 6A) for selecting filters and a preview panel (Fig. 6B) for visualizing different option combinations' effects. Each filter (e.g., *OS Type*) has many options (e.g., *Linux*). The key analytical challenge is that effective option combinations are often non-obvious (e.g., *Linux* with *TypeError* might be significant, while *Linux* with *OverLimit* is not). To overcome this, a "Generate" button invokes our refining model to automatically discover and suggest insightful option combinations. The preview panel visualizes these suggested combinations using a parallel coordinate plot, while the corresponding filtered data is displayed as line or Gantt charts.

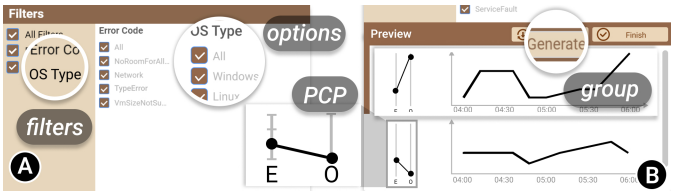

Fig. 6: The filter card consists of two parts: (A) users can select filters and options in the selection panel; (B) the preview panel displays alternative groups of options and filtered attributes. e.g., the PCP shows **E**rrorcode-TypeError (3^{rd}) with **O**SType-Linux (2^{nd}).

The refining model automatically discovers filter combinations that make an attribute's trend most relevant to existing clues. We adapted one of the latest and most relevant tools, MID [17], which adopts meta-heuristic search and maximizes only increasing trends. Our modified model searches within user-selected filters for combinations that produce both increasing and decreasing trends, as a root cause can manifest as either. The model then ranks resulting groups by temporal relevance to the clues already on the board (using Eq. 2), ensuring suggestions are semantically aligned with the ongoing analysis.

5.3.3 Annotate Reasoning Logic

RCInvestigator supports a hybrid approach to annotating reasoning logic, combining automated machine-generated notes with flexible, human-authored annotations.

Machine annotations. The system automatically adds key analytical actions using templates. For clue collection, a template summarizes the graph traversal path that was followed. E.g., "*<Cluster A> [belongs to] <Zone A>*" For clue refinement, another template explicitly states the filter chain that was applied, ensuring that every data transformation is recorded. E.g., "*Filtered by <OS Type> and <Error Type>*"

Human annotations. To support expressive sensemaking, analysts are provided with a flexible canvas. They can freely add shapes (e.g., circles, arrows) to highlight anomalies or visually connect evidence. Additionally, styled text can be used to distinguish between personal hypotheses, formal conclusions, or mitigation suggestions.

5.4 Output Results

After users find the root cause and make annotations, they may click the camera icon to save the analysis process as a PNG image, which will help users share the results through discussion boards or email. Users can also download the anomaly analysis log in JSON form and share it with others for better collaborative analysis (**T5**).

6 Case Study

This section presents two cases and expert interviews[1].

6.1 Case 1: Updated Inconsistent Strategy

EA received an alert email: *"[outage] Area01 incidents ..."* Without RCInvestigator, EA would reconstruct investigation context from scattered tribal knowledge about *Area01*. Instead of this ad-hoc process, her investigation now begins by formalizing this structural understanding, as described in the first step.

S1. Build a knowledge graph (Fig. 7A). EA first confronted the combinatorial complexity of mapping cross-component dependencies. She abstracted the platform's physical hierarchy into 3 structural layers (*Area, Zone, Cluster*) and 2 application layers (*Customer, Order*). She defined their relations as well, for example, connecting *Zone* and *Area*. This helps her compress hundreds of instances into 5 entity concepts and 9 relations. She added necessary attributes like *Incident Count (the number of hourly incidents)* and *Unuse Reserved VMs (the number of virtual machines reserved by customers but unused)*. She then defined query templates for data retrieval logic in the building board. This one-time effort is the crucial step that addresses **P1**. The built reusable graph now empowers the machine agent to autonomously collect relevant data, eliminating the need for analysts to write complex, ad-hoc queries during a live incident. EA saved the graph as a persistent asset.

S2. Find that many incidents happen in a short period (Fig. 7B). EA proceeded to the monitoring board to find an analytical entry point. This step is typically blocked by **P1.1**, forcing analysts to manually merge alerts and write query scripts just to get started. Instead, RCInvestigator had already populated the view with detected incident logs and key KPIs. While scanning the log panel, EA observed numerous *Failed/ComputeFailed* incidents all tied to a single user, *Customer80*. EA double-clicked one, the board instantly highlighted the corresponding *Area01-Zone02-Incident Count* KPI. The KPI line clearly displayed an anomalous peak. This tailored connection design allowed her to efficiently locate the analysis entry point. She then brush-selected this peak to transition to the investigation.

S3. Investigate why there are a large number of incidents. Investigating this spike manually would immediately force the analyst into a cycle of **P1** and **P2**, letting EA try to guess which of the hundreds of metrics to query first. Instead, upon entering the investigation board, EA was immediately presented with multiple clues in *Area01-Zone02* (**P1.2**). She noticed that the number of *unused reserved VMs* and *error code count* peaked while *allocable nodes* decreased (Fig. 8A_2) at nearly the same time as *the incident count* (Fig. 8A_1). This led EA to her first hypothesis: a user jam was exhausting resources. Validating this would normally require more manual queries to dig into error codes, adding cognitive load (**P2.1&2.2**). Here, EA simply used the "refine" interaction on the *Error Code Count*(Fig. 8A_3). The system's refining model automatically analyzed the data and surfaced that the dominant error code was *No Room for Allocation*. This evidence directly contradicted her hypothesis, so EA negated her first hypothesis about the customers.

EA formed the second hypothesis (Fig. 8B): a flaw in the reservation process itself. She decided to investigate a customer, the corresponding order, and its reservation for further insight. To perform this complex cross-component reasoning, EA used a simple "rightward expansion". This automatically added *Customer80* and *Cluster25*. The clues in *Cluster25* immediately presented contradictory facts. The *utilization* (Fig. 8B_2) was low, indicating there were sufficient physical resources, yet the *unused reserved VMs* was high, aligning with *No Room For Allocation* error. This strongly supported her hypothesis. Besides, the expanding model correlates events and time series, helping EA to gain a change in the *build version* (Fig. 8B_3) about two days before the incidents caught her attention. EA found that this latest update included a reservation strategy, so she ensured that RC was hidden behind the new build version, specifically related to the reservation update.

Finally, EA investigated the influence range. She applied the same investigation logic to the parent *Zone02* (Fig. 8C) and *Area01* (Fig. 8D) via clicking instead of adapting query scripts. In summary, EA determined that the root cause was the reservation strategy.

S4. Annotate and save. Finally, EA needed to document her findings. This is where manual RCA fails with **P3.1&3.2**, as analysts must manually assemble scattered query results, losing the reasoning context. Instead, EA used the investigation board to create a visual summary. She annotated her reasoning directly, circling key evidence and adding text notes for both her negated hypothesis ("[X] Mismatching error code") and her confirmed one ("[V] The Reservation Strategy changed..."). She then exported the entire canvas as a single, intuitive image. This creates a structured visual narrative of the full investigation.

6.2 Case 2: Normal Nodes Error

The alert email EC received informed him that the allocable nodes in *Area07-Zone01* decreased dramatically.

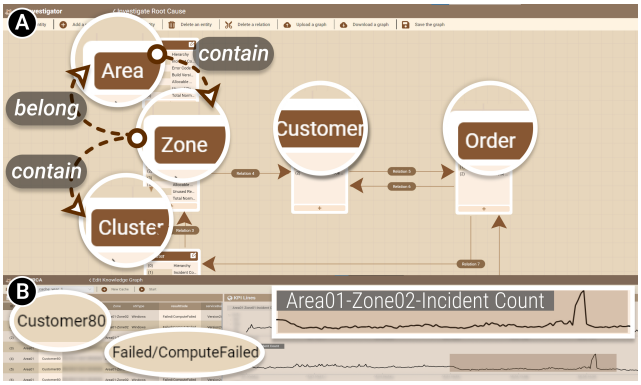

Fig. 7: The first and second steps of Case 1. (A) EA built a knowledge graph with 5 entities and 9 relations based on her domain knowledge, such as each "Area" contains many "Zone"s. (B) EA observed many incidents were related to *Customer80* and happened in *Area01-Zone02*.

[1]This study has been approved by State Key Lab of CAD&CG.

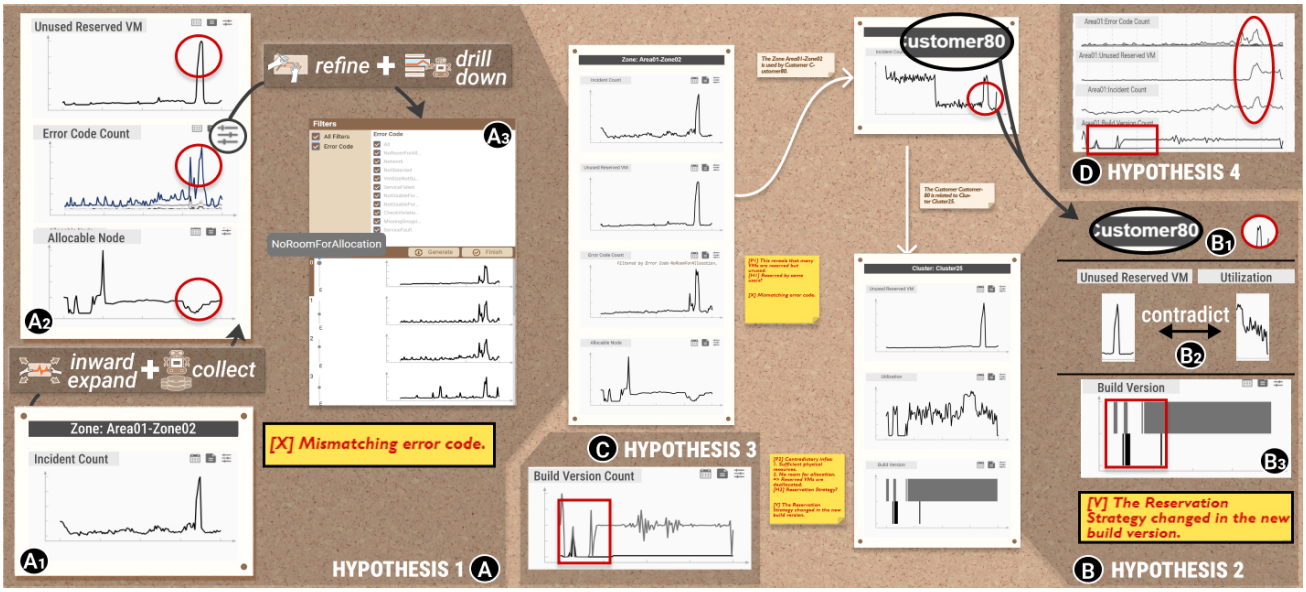

Fig. 8: The investigation process of Case 1. EA proposed four hypotheses. (A) In Hypothesis 1, EA used inward expansion to collect three clues and refined the error code clue. She negated this due to the mismatching error code. (B) In Hypothesis 2, EA collected clues and found a contradicting pattern. She confirmed this due to a change in the build version. Both (C) and (D) show the influence range of such a root cause.

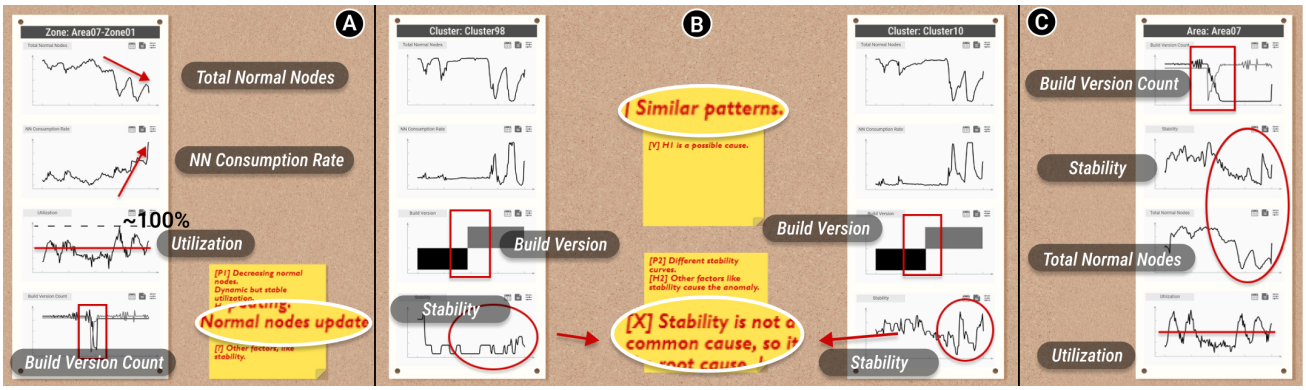

Fig. 9: The investigation process of Case 2. EC investigated clues from three different hierarchies. (A) EC found a drop in normal nodes and a high consumption rate. He thought it might be caused by a recent update. (B) EC further checked his guess: excluded the stability factor on clusters and confirmed the build version update. (C) EC investigated the influence range of the root cause.

S1&2. Find that the number of normal nodes drops dramatically. EC began by loading the knowledge graph EA had previously built and saved. This usable asset directly addresses **P2.2**, as it allowed him to leverage his colleague's formalized domain knowledge. In the monitoring board, EC observed the KPI lines for a while. He quickly found that the number of *total normal nodes* in *Area07-Zone01* had been dropping for a few days, and the trend had become steeper over time. EC brush-selected the time range for further analysis.

S3. Investigate why the normal nodes decrease. EC first decided to understand anomaly at the zone level. He used an "inward expansion", allowing him to efficiently gather multiple factors at once (**P1.2**). The expanding model collected and surfaced two clues: the total number of *normal nodes* was decreasing while the *consumption rate* of normal nodes was consistently increasing and approaching 100% (Fig. 9A). It also highlighted a *build version* update that was temporally related with this, leading EC to suspect this update was the cause.

Next, EC needed to validate the hypothesis at the cluster level, a cross-hierarchy analysis that can cause **P2.2**. Instead of a manual search, he used a "downward expansion". The system recommended *Cluster10* and *Cluster98* (Fig. 9B). By comparing these siblings, he ruled out *stability* (which only changed in *Cluster10*) as a common cause and confirmed the *build version* update, which affected both, was the true root cause. Finally, an "upward expansion" to *Area07* (Fig. 9C) confirmed the widespread impact.

S4. Conclude the summary of this investigation. Finally, EC complemented his findings. He added his reasoning logic and a summary directly on to the board, which describes that RC was related to *normal node* judgment and recycling. The overall across-hierarchy reasoning process was now stored as a single three-level image for easy sharing (**P3**). EC then exported this analysis report.

6.3 Expert Interview

We collected feedback from four domain experts (EA, EB, EC, and ED) via one-on-one structured interviews. The whole process consists of three stages: (1) **[10min]** introduction of RCInvestigator; (2) **[50min]** free investigation of anomalies' root causes; (3) **[15min]** a structured interview. We summarized the feedback from three aspects as follows. We provided a one-year dataset and tasked experts (EA, EB) with investigating Area01 and (EC, ED) with Area07, allowing them to choose their own specific anomalies and reasoning paths. The case studies are the detailed results from this stage.

Investigation framework. All experts praised the proposed investigation framework. EA said the building stage facilitated investigation experience forming and sharing and was useful for formulating knowledge basis. Besides, EB tried to edit existing investigation knowledge and said it was easy to extend. EC and ED both commented that RCInvestigator enabled easy reuse of the investigation experience. Moreover, all experts highlighted that RCInvestigator reduced the labor

effort needed in data collection. It is effective as they do not need to switch between databases and analysis panels during the investigation, so the investigation becomes more *"fluent"* and *"coherent"*.

Visualization and interactions. Experts agreed that visualizations and interactions adopted in RCInvestigator were intuitive and helpful. All experts said that the semantic layout of clues helped them organize their minds logically as well as inspire further investigation. Besides, EA and ED praised the collaborative summaries and annotations. EA said, *"Direct conclusions given by the machine agent help me value clues quickly."* Moreover, all experts commented that the design was expressive and easy-to-understand. *"Through simple charts, I can fully understand what happened and thus devote my effort all in the investigation instead of reading the chart"*.

Comparing with the existing workflow. All experts mentioned that RCInvestigator performs better than the existing workflow in the time aspect. They all think the most improved process is the reasoning stage and give different reasons. Some experts mentioned the labor aspect, thinking that RCInvestigator frees them from laborious coding tasks and helps them focus on the reasoning. EA said she can concentrate more on thinking and inferring without code issues. ED and EB shared similar positive opinions on reducing data collection efforts. Some experts also mentioned the improvement brought by reducing mental burden. EB said, *"it (the new workflow) opens up my thinking by showing me different angles and possible factors"*. EC thought that RCInvestigator helped uncover previously ignored KPIs. ED mentioned that the new workflow reduces the workload of organizing and understanding clues.

Suggestions. Experts also give many valuable suggestions. EA said that we should allow deleting unexpected clues. EB and EC suggested adding a mini-map of the knowledge graph during the investigation because it provides an overview and facilitates a comprehensive understanding of the whole reasoning. ED said we could add a dashed line while hovering on charts as it would be easier to align time across charts. We improved RCInvestigator based on these suggestions.

6.4 Quantitative Evaluation

We conducted two quantitaive experiments. Further details are discussed in the supplementary material. (1) We tested our model's efficacy on semi-synthetic datasets (100 and 1000 clues). The dataset contains 10% of relevant clues to the cause. Our model outperformed a pure correlation baseline: on the 1000-series test, our model achieved a Precision/Recall of 0.94/0.93, while the baseline only achieved 0.80/0.79. (2) We tested our layout's performance by measuring the latency for graphs of varying complexity. The results confirm the system is highly interactive for its intended scope: "medium" graphs (<50 nodes), typical of our case studies, averaged 556ms. Our two-step hierarchical layout also inherently prevents entity overlap, with rendered examples provided in the supplementary material.

7 Discussion

This section presents the significance and generalizability, lessons learned, limitations, and future work of RCInvestigator.

Significance and generalizability. We discuss the significance of RCInvestigator in the following two aspects.

Investigation framework. We proposed a root cause investigation framework based on human-machine-collaborative schema. With this framework, analysts can be freed from labor-intensive tasks such as data querying and information gathering and instead focus on thinking, reasoning, and analysis. Meanwhile, machines collect data, recommend clues, and summarize information controllably. Though this framework is designed for cloud computing systems, we argue that it can be applied to other scenarios, like smart manufacturing RCA. Especially in scenarios where RCA requires analyzing large amounts of data from multiple sources and heavily relies on domain knowledge.

Techniques. RCInvestigator comprises a novel interactive root cause reasoning model and a real-world investigation board-inspired time-oriented data visualization, which helps to identify the complex underlying causes behind anomalies. This paper proposes two types of hypothesis interactions and corresponding models for time-oriented data exploration. We argue that these structured interactions for generalization, comparison, and specialization are fundamental patterns for reasoning with temporal evidence and are generalizable to a wide range of temporal analysis tasks.

VA design for complex scenarios. Our work contributes a visualization design centered on semantic simplicity. Instead of designing novel or complex charts, we intentionally use familiar visualizations like line and Gantt charts, which are highly interpretable. The novelty lies in our system's ability to organize these simple plots into a meaningful structure that reflects the reasoning process. By preserving the semantic relations between clues (e.g., hierarchy, comparison), our layout transforms a potentially overwhelming collection of individual charts into a cohesive and intelligible visual narrative of the investigation.

Lessons learned. Through our collaboration with experts, we have learned valuable lessons. First, providing the necessary textual summaries when recommending visual elements is crucial. We initially presented recommended clues as a ranked list of small charts, but experts found this cognitively overwhelming, even with simple visuals. We learned that when presenting a large volume of evidence, visual plots must be complemented by natural language summaries that classify and abstract the key findings to be effective. Second, semantic layouts are better than just clean layouts. We first used a force-directed layout, which was clean but semantically meaningless to our experts, causing them to get lost. We learned that for complex reasoning, a layout's alignment with the user's mental model is far more important than generic visual optimization.

Learning curve of RCInvestigator. RCInvestigator's learning curve varies across its distinct operational stages. The building stage represents the primary initial investment. This is a deliberate tradeoff where the structuring effort required to build the graph replaces the high, continuous cost of repeated manual investigation. Although it requires building an initial graph, this "cold-start" is mitigated as the graph formalizes the expert's existing mental model and can be built progressively, allowing analysts to gain immediate value. The resulting knowledge base becomes a stable, reusable asset that amortizes the one-time setup cost. Besides, the monitoring stage has a minimal learning curve due to its familiar UI to experts' frequently used table form, while the Reasoning Stage is designed for rapid adoption, centered on two simple interactions (expanding and refining) that directly align with an analyst's natural problem-solving process.

Limitations and future work. RCInvestigator has some limitations that should be considered. (1) The current reliance on a human-constructed knowledge graph, while ensuring high reliability, requires for an initial "cold-start" effort. Future work should explore semi-automated approaches to streamline this process without sacrificing the quality of the domain knowledge. (2) The machine agent is good at finding direct relations but struggles with indirect ones, like identifying a fault when two metrics suddenly stop correlating. (3) Our evaluation was limited by challenges with data compliance and analyst recruitment. We only conducted an initial validation using case studies and expert interviews. Future work will focus on an in-situ study.

To further expand the capabilities of RCInvestigator, we outline two key areas for future work. (1) We will enhance knowledge injection by investigating semi-automated graph construction. We plan to integrate LLMs to extract entities and relations from unstructured historical logs and business knowledge, supported by interactive human-in-the-loop validation. (2) We plan to extend RCInvestigator's functionality to support collaborative analysis among multiple analysts simultaneously, thereby increasing collaborative efficiency.

8 Conclusion

In this paper, we proposed RCInvestigator, an interactive system for investigating anomaly root causes in cloud computing systems. Through the collaboration with domain experts, three challenges were identified in investigating root causes. We designed a novel human-machine-collaborative framework consisting of four stages (building, monitoring, reasoning, and concluding) to address these challenges. We built RCInvestigator based on this framework and evaluated the system through two real-world use cases, receiving positive feedback from experts.

Acknowledgments

The work was supported by NSFC (62402421, U22A2032, 62421003), and Zhejiang Provincial Natural Science Foundation of China under Grant No. LD25F020003.

References

[1] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing time-oriented data — A systematic view. *Computers & Graphics*, 31(3):401–409, 2007.

[2] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. 2023.

[3] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic. Time Curves: Folding time to visualize patterns of temporal evolution in data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):559–568, 2016.

[4] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. *SIGCOMM Comput. Commun. Rev.*, 37(4):13–24, aug 2007.

[5] T. Baumgartl, M. Petzold, M. Wunderlich, M. Hohn, D. Archambault, M. Lieser, A. Dalpke, S. Scheithauer, M. Marschollek, V. M. Eichel, N. T. Mutters, H. Consortium, and T. V. Landesberger. In search of patient zero: Visual analytics of pathogen transmission pathways in hospitals. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):711–721, 2021.

[6] J. Bernard, C.-M. Barth, E. Cuba, A. Meier, Y. Peiris, and B. Shneiderman. Ivesa – visual analysis of time-stamped event sequences. *IEEE Transactions on Visualization and Computer Graphics*, 31(4):2235–2256, 2025.

[7] J. Bernard, M. Solen, H. N. Lauscher, K. Stewart, K. Ho, and T. Munzner. Viva: Virtual healthcare interactions using visual analytics, with controllability through configuration. *IEEE Transactions on Visualization and Computer Graphics*, 31(12):10384–10402, 2025.

[8] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen. Fingerprinting the datacenter: automated classification of performance crises. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, page 111–124, New York, NY, USA, 2010. Association for Computing Machinery.

[9] P. Chen, Y. Qi, and D. Hou. CauseInfer: Automated end-to-end performance diagnosis with hierarchical causality graph in cloud environment. *IEEE Transactions on Services Computing*, 12(2):214–230, 2019.

[10] P. Chen, Y. Qi, P. Zheng, and D. Hou. CauseInfer: Automatic and Distributed Performance Diagnosis with Hierarchical Causality Graph in Large Distributed Systems. In *Proceedings of the IEEE Conference on Computer Communications*, pages 1887–1895, 2014.

[11] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system history. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, page 105–118, New York, NY, USA, 2005.

[12] Dagre. Dagre: Directed graph layout for JavaScript. https://github.com/dagrejs/dagre. Last accessed: Nov 22, 2023.

[13] Z. Deng, D. Weng, S. Liu, Y. Tian, M. Xu, and Y. Wu. A survey of urban visual analytics: Advances and future directions. *Computational Visual Media*, 9(1):3–39, 2023.

[14] J. Dong, H. Zhang, M. Cui, Y. Lin, H.-Y. Wu, and C. Bi. Tcevis: Visual analytics of traffic congestion influencing factors based on explainable machine learning. *Visual Informatics*, 8(1):56–66, 2024.

[15] Y. Fang, H. Xu, and J. Jiang. A survey of time series data visualization research. In *Proceedings of the IOP Conference Series: Materials Science and Engineering*, volume 782, pages 1–10, 2020.

[16] Z. Gao, C. Cecati, and S. X. Ding. A Survey of Fault Diagnosis and Fault-Tolerant Techniques – Part I: Fault Diagnosis with Model-Based and Signal-Based Approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767, 2015.

[17] J. Gu, C. Luo, S. Qin, B. Qiao, Q. Lin, H. Zhang, Z. Li, Y. Dang, S. Cai, W. Wu, Y. Zhou, M. Chintalapati, and D. Zhang. Efficient incident identification from multi-dimensional issue reports via meta-heuristic search. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 292–303, 2020.

[18] Y. Guo, S. Guo, Z. Jin, S. Kaul, D. Gotz, and N. Cao. Survey on visual analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):5091–5112, 2022.

[19] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. Ullah Khan. The Rise of "Big Data" on Cloud Computing: Review and Open Research Issues. *Information Systems*, 47:98–115, Jan. 2015.

[20] S. P. Kavulya, K. Joshi, F. D. Giandomenico, and P. Narasimhan. Failure Diagnosis of Complex Systems. In *Resilience Assessment and Evaluation of Computing Systems*, pages 239–261. 2012.

[21] P. Kayongo, J. Hoffswell, S. Saini, S. Garg, E. Koh, H. Wang, and T. Jacobs. ViSRE: A unified visual analysis dashboard for proactive cloud outage management. In *Proceedings of the Working Conference on Software Visualization*, pages 5–16, 2022.

[22] A. Keshavarzi, A. T. Haghighat, and M. Bohlouli. Research Challenges and Prospective Business Impacts of Cloud Computing: A Survey. In *Proceedings of the IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, pages 731–736, 2013.

[23] M. Kim, R. Sumbaly, and S. Shah. Root Cause Detection in A Service-Oriented Architecture. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems*, pages 93–104, 2013.

[24] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo. RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):299–309, 2019.

[25] J. Lin, P. Chen, and Z. Zheng. Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In *Proceedings of the International Conference on Service-Oriented Computing*, pages 3–20, 2018.

[26] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. VizTree: A tool for visually mining and monitoring massive time series databases. In *Proceedings of the International Conference on Very Large Data Bases*, pages 1269–1272, 2004.

[27] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen. Log clustering based problem identification for online service systems. In *Proceedings of the 38th International Conference on Software Engineering Companion*, page 102–111, New York, NY, USA, 2016.

[28] 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 *Proceedings of the IEEE International Symposium on Software Reliability Engineering*, pages 48–58, 2020.

[29] S. Liu, D. Weng, Y. Tian, Z. Deng, H. Xu, X. Zhu, H. Yin, X. Zhan, and Y. Wu. ECoalVis: Visual analysis of control strategies in coal-fired power plants. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1091–1101, 2023.

[30] C. Luo, J.-G. Lou, Q. Lin, Q. Fu, R. Ding, D. Zhang, and Z. Wang. Correlating events with time series for incident diagnosis. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 1583–1592, 2014.

[31] Microsoft. Kusto Query Language (KQL) Overview. https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/. Last accessed: Sep 11, 2023.

[32] C. Muelder, B. Zhu, W. Chen, H. Zhang, and K.-L. Ma. Visual Analysis of Cloud Computing Performance Using Behavioral Lines. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1694–1704, 2016.

[33] W. K. Muhlbauer. Risk: Theory and Application. In *Pipeline Risk Management Manual (Third Edition)*, pages 1–19. 2004.

[34] H. Nguyen, Z. Shen, Y. Tan, and X. Gu. FChain: Toward black-box online fault localization for cloud systems. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 21–30, 2013.

[35] H. Nguyen, Y. Tan, and X. Gu. PAL: Propagation-aware anomaly localization for cloud hosted distributed applications. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 1–8, 2011.

[36] P. Notaro, J. Cardoso, and M. Gerndt. A survey of aiops methods for failure management. *ACM Trans. Intell. Syst. Technol.*, 12(6), nov 2021.

[37] A. Offenwanger, M. Brehmer, F. Chevalier, and T. Tsandilas. Timesplines: Sketch-based authoring of flexible and idiosyncratic timelines. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):34–44, 2024.

[38] Pallets. Welcome to Flask. https://flask.palletsprojects.com. Last accessed:

Nov 22, 2023.

[39] A. Podgurski, D. Leon, P. Francis, W. Masri, M. Minch, J. Sun, and B. Wang. Automated support for classifying software failure reports. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, page 465–475, USA, 2003. IEEE Computer Society.

[40] C. Raj, L. Khular, and G. Raj. Clustering Based Incident Handling For Anomaly Detection in Cloud Infrastructures. In *Proceedings of the International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 611–616, 2020.

[41] Redpoll. A Bayesian change point library. https://pypi.org/project/changepoint/. Last accessed: Nov 22, 2023.

[42] P. Rosenthal, L. Pfeiffer, N. H. Müller, and P. Ohler. VisRuption: Intuitive and efficient visualization of temporal airline disruption data. *Computer Graphics Forum*, 32:81–90, 2013.

[43] S. A. Sakin, A. Bigelow, R. Tohid, C. Scully-Allison, C. Scheidegger, S. R. Brandt, C. Taylor, K. A. Huck, H. Kaiser, and K. E. Isaacs. Traveler: Navigating task parallel traces for performance analysis. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):788–797, 2023.

[44] A. Samir and C. Pahl. DLA: Detecting and localizing anomalies in containerized microservice architectures using markov models. In *Proceedings of the International Conference on Future Internet of Things and Cloud*, pages 205–213, 2019.

[45] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012.

[46] H. Shan, Y. Chen, H. Liu, Y. Zhang, X. Xiao, X. He, M. Li, and W. Ding. ε-Diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In *Proceedings of the World Wide Web Conference*, pages 3215–3222, 2019.

[47] Shilpika, T. Fujiwara, N. Sakamoto, J. Nonaka, and K.-L. Ma. A visual analytics approach for hardware system monitoring with streaming functional data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2338–2349, 2022.

[48] H. Shiravi, A. Shiravi, and A. A. Ghorbani. A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1313–1329, 2012.

[49] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada. Survey on Models and Techniques for Root-Cause Analysis. *CoRR*, abs/1701.08546, 2017.

[50] D. Sun, R. Huang, Y. Chen, Y. Wang, J. Zeng, M. Yuan, T.-C. Pong, and H. Qu. PlanningVis: A visual analytics approach to production planning in smart factories. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):579–589, 2020.

[51] J. Thalheim, A. Rodrigues, I. E. Akkus, P. Bhatotia, R. Chen, B. Viswanath, L. Jiao, and C. Fetzer. Sieve: Actionable insights from monitored metrics in distributed systems. In *Proceedings of the ACM/IFIP/USENIX Middleware Conference*, pages 14–27, 2017.

[52] C. Tominski, J. Abello, and H. Schumann. Axes-based visualizations with radial layouts. In *Proceedings of the ACM Symposium on Applied Computing*, pages 1242–1247, 2004.

[53] H. Wang, P. Nguyen, J. Li, S. Kopru, G. Zhang, S. Katariya, and S. Ben-Romdhane. GRANO: Interactive graph-based root cause analysis for cloud-native distributed data platform. *Proceedings of the VLDB Endowment*, 12(12):1942–1945, 2019.

[54] T. Wang, Z. Li, and J. Zhang. Egocentric visual analysis of dynamic citation network. *Journal of Visualization*, 25(6):1343–1360, 2022.

[55] S. Wastie. The real cost of downtime, the real potential of DevOps. App Dynamics, Jul 2018. Available: https://www.appdynamics.com/blog/engineering/idc-devops-cost-downtime/ (Last accessed: Nov 22, 2024).

[56] S. Wolfe. Amazon's one hour of downtime on Prime Day may have cost it up to $100 million in lost sales. Business Insider, Jul 2018. Available: https://www.businessinsider.com/amazon-prime-day-website-issues-cost-it-millions-in-lost-sales-2018-7 (Last accessed: Mar 30, 2024).

[57] W. E. Wong, R. Gao, Y. Li, R. Abreu, and F. Wotawa. A Survey on Software Fault Localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016.

[58] K. Wongsuphasawat and D. Gotz. Outflow: Visualizing patient flow by symptoms and outcome. In *Proceedings of the IEEE VisWeek Workshop on Visual Analytics in Healthcare*, pages 25–28, 2011.

[59] H. Wu, H. Zhang, J. Cheng, J. Guo, and W. Chen. Perspectives on point cloud-based 3d scene modeling and xr presentation within the cloud-edge-client architecture. *Visual Informatics*, 7(3):59–64, 2023.

[60] L. Wu, J. Tordsson, E. Elmroth, and O. Kao. MicroRCA: Root cause localization of performance issues in microservices. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, pages 1–9, 2020.

[61] W. Wu, Y. Zheng, K. Chen, X. Wang, and N. Cao. A visual analytics approach for equipment condition monitoring in smart factories of process industry. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 140–149, 2018.

[62] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, and H. Qu. CloudDet: Interactive Visual Analysis of Anomalous Performances in Cloud Computing Systems. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1107–1117, Jan. 2020.

[63] P. Xu, H. Mei, L. Ren, and W. Chen. ViDX: Visual diagnostics of assembly line performance in smart factories. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):291–300, 2017.

[64] F. Yan, Y. Wang, X. Yue, K.-K. Wong, K. Mao, R. Zhang, H. Qu, H. Zhu, M. Zhu, and W. Chen. Fundselector: A visual analysis system for mutual fund selection. *Visual Informatics*, 9(4):100258, 2025.

[65] Y. Yang, X. Yi, Y. Jin, S. Li, K. Ma, S. Liu, D. Deng, D. Weng, and Y. Wu. Pvesight: Dimensionality reduction-based anomaly detection and visual analysis of photovoltaic strings. *Visual Informatics*, 9(3):100243, 2025.

[66] E. You. Vue.js: The Progressive JavaScript Framework. https://vuejs.org/. Last accessed: Nov 22, 2023.

[67] 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 *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 4373–4382, 2021.

[68] F. Zhou, X. Lin, C. Liu, Y. Zhao, P. Xu, L. Ren, T. Xue, and L. Ren. A Survey of Visualization for Smart Manufacturing. *Journal of Visualization*, 22:419–435, 2019.

[69] Z. Zhou, Y. Li, Y. Ni, W. Xu, G. Hu, Y. Lai, P. Chen, and W. Su. Visci: A visualization framework for anomaly detection and interactive optimization of composite index. *Visual Informatics*, 8(2):1–12, 2024.