algorithms

MDPI

Article

Fault Location Method Based on Dynamic Operation and Maintenance Map and Common Alarm Points Analysis

Sheng Wu [1,2] and Jihong Guan [1,*]

[1] College of Electronic Information and Engineering, Tongji University, Shanghai 201804, China; 1711022@tongji.edu.cn
[2] ICBC Data Center, Shanghai 200131, China
* Correspondence: jhguan@tongji.edu.cn

Abstract: Under a distributed information system, the scale of various operational components such as applications, operating systems, databases, servers, and networks is immense, with intricate access relationships. The silo effect of each professional is prominent, and the linkage mechanism is insufficient, making it difficult to locate the infrastructure components that cause exceptions under a particular application. Current research only plays a role in local scenarios, and its accuracy and generalization are still very limited. This paper proposes a novel fault location method based on dynamic operation maps and alarm common point analysis. During the fault period, various alarm entities are associated with dynamic operation maps, and alarm common points are obtained based on graph search addressing methods, covering deployment relationship common points, connection common points (physical and logical), and access flow common points. This method, compared with knowledge graph approaches, eliminates the complex process of knowledge graph construction, making it more concise and efficient. Furthermore, in contrast to indicator correlation analysis methods, this approach supplements with configuration correlation information, resulting in more precise positioning. Through practical validation, its fault hit rate exceeds 82%, which is significantly better than the existing main methods.

Keywords: fault location; dynamic operation and maintenance map; common alarm points

Citation: Wu, S.; Guan, J. Fault Location Method Based on Dynamic Operation and Maintenance Map and Common Alarm Points Analysis. *Algorithms* **2024**, *17*, 217. https://doi.org/10.3390/a17050217

Academic Editor: Frank Werner

Received: 10 April 2024
Revised: 11 May 2024
Accepted: 14 May 2024
Published: 16 May 2024

1. Introduction

In distributed information systems, the dependencies and connectivity access relationships between data center infrastructure, servers, storage, networks, systems, databases, and applications become exceedingly complex and invisible due to factors such as resource pooling and insufficient cross-discipline operational and maintenance data sharing. Each professional monitoring system runs independently, resulting in a prominent silo effect and insufficient linkage mechanisms, making it difficult to accurately locate the underlying objects that cause exceptions under a particular application. Current research focuses on knowledge graph and indicator correlation analysis methods. However, knowledge graph technology has a high complexity and is only effective in local scenarios such as networks, and therefore, it is still not ready for practical use. Indicator correlation analysis lacks configuration correlation information between objects, thus its accuracy is limited. This paper proposes a basic fault location method based on dynamic operation maps and alarm common point analysis. First, a distributed information system dynamic operation map is constructed, including topology information, node mutual visits, application high-availability deployment, covering applications, systems, and networks, providing topological relationships for fault location. Meanwhile, the troubleshooting process triggered by a single abnormal business flow is studied, including database access flows, read/write I/O flows, and basic component mutual visits. During the fault period, various alarm entities are associated with dynamic operation maps, and alarm common points are

obtained based on graph search addressing methods, covering deployment relationship common points, physical and logical connection common points, and access flow common points, thus positively locating the possible root cause of the fault. This method is simpler and more efficient than knowledge graph methods and more accurate than indicator correlation analysis methods due to the addition of configuration correlation information.

2. Research Background

Due to the complexity of distributed information system architecture, when a business application encounters an exception, a large number of alerts from different levels such as the operating system, network, and database are usually received at the infrastructure level, making it difficult to focus on the cause of the failure in the first place, and it is difficult to drill down from the abnormal application to analyze and locate the faulty operation and maintenance object at the infrastructure level.

There are currently two main types of research. One is based on knowledge graphs [1–4], involving multiple steps such as knowledge graph construction, knowledge learning, and knowledge reasoning. Key technologies include ontology modeling of domain knowledge, graph mining algorithms, and logical reasoning algorithms based on fast graph search. The technical complexity is high. This method has strong generalization but insufficient accuracy, and there are no mature cases of practical application.

The knowledge graph technology has been used by Tsinghua University in the field of basic-level fault location in the vertical direction, applied to alarm traceability research [5–7].

Another type is based on root cause similarity analysis, such as the CoFlux algorithm [8] and FluxInfer algorithm [9]. Due to the lack of topological relationship knowledge, only based on algorithm analysis of the shape changes of indicator data, the accuracy is not good in the distributed architecture.

A comparison of commonly used basic-level fault location methods is shown in Table 1.

Table 1. Comparison of commonly used basic-level fault location methods.

Method Type	Method Core Idea	Existing Problems
Knowledge Graph	Build a knowledge graph based on operation experience and topology relationships, and achieve location through knowledge fusion and fault reasoning.	The construction of the knowledge graph is relatively complex and effective in local scenarios, but there is still a gap from practical use.
1-3 Similarity Analysis	After applying anomalies, directly compare the mutation patterns of a large number of indicators in the basic layer to locate.	Lack of configuration relationships in the basic layer, limited accuracy.

It can be seen that there is insufficient accuracy in basic-level fault location technology, which is effective in local scenarios but still has a gap from practical use.

Take the distributed financial information system as an example. Considering the extremely strict stability guarantee and regulatory requirements brought by the characteristics of a financial business, the ideal emergency response time requirement is to detect anomalies within 1 min, provide the scope of the object of fault location within 5 min, and complete isolation, switching, expansion, restart and other emergency response operations within 10 min. The overall business impact time is controlled within 20 min. Against this background, fault location requires accuracy and efficiency, firstly, the coverage of fault types is sufficiently complete, and secondly, it meets the time requirement of fast location in minutes.

As the financial distributed information system architecture evolves towards infrastructure cloudification, containerization of runtime environments, and microservice-oriented

business systems, the mutual invocation of IT components has become increasingly common and complex. However, the current widely established Configuration Management Database (CMDB) primarily focuses on the management of configuration items, with relational databases serving as the primary means to store static configuration information, making it difficult to dynamically reflect the topological relationships between configuration items. Graph databases, on the other hand, are inherently suited for expressing topological relationships between objects. Therefore, there is a need to establish a dynamic operational maintenance map based on a graph database to reflect the configuration topology between various operational maintenance objects in real-time.

This map encompasses topological information, node inter-access, application high-availability deployment and more, covering the dynamic operational maintenance map of distributed information systems across applications, systems, and networks. It provides real-time updated topological relationships for fault localization. The map construction primarily involves the establishment of a unified data model, targeting various types of information such as CMDB, PaaS cloud configuration centers, and specialized configuration ledgers. The primary approach is rule-based processing, supplemented by machine learning methods, to achieve entity alignment and association relationship construction. Compared to knowledge graph construction methods, this approach is more concise, eliminating the need for complex methods such as knowledge annotation and ontology construction. The dynamic map reflects more precise associations, without the partial ambiguity introduced by knowledge graph reasoning, making it more suitable for the precision requirements in the operational maintenance field.

The focus of this research is on the construction of a dynamic operational maintenance map and a fault localization method based on this map.

3. Model Architecture

Due to the extremely large scale of operation and maintenance objects involved in the same business, the method of directly analyzing the fault location conclusion from the data source has become feasible. It is necessary to rely on the accurate abnormal perception of each operation and maintenance object to carry out analysis and location for the abnormal alarm of each operation and maintenance object. Therefore, this method relies on rich and complete alarm sources, including the system, network, application, equipment, and other professional monitoring alarms. The key technical point is to screen multiple source alarms and construct key features. Based on the complete and accurate dynamic operation and maintenance map data and various professional alarm data, this method studies the troubleshooting process triggered by an abnormal business flow, mainly including database access flow, read-write I/O flow, and microservice access flow. The business flow is shown in Figure 1. Through the analysis of the commonalities of various data flows (deployment relationship commonalities, connection commonalities (physical, logical), access flow commonalities), triggered by abnormal business flow, the possible root cause of the failure is located forwardly.

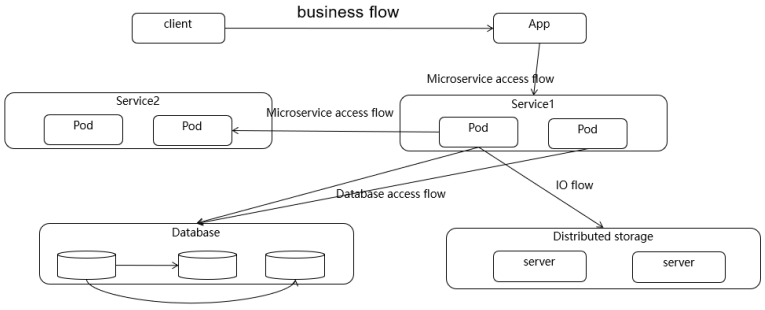

Figure 1. Business flow.

The model architecture is shown in Figure 2.

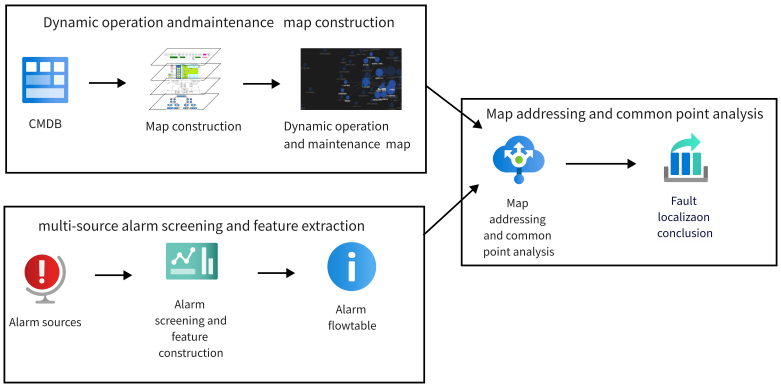

Figure 2. Model architecture.

The model is divided into three main modules: dynamic operation and maintenance map construction, multi-source alarm screening and feature extraction, and map addressing and common point analysis. This method is based on dynamic operation and maintenance maps and various professional alarm source data, with complete fault coverage types. Compared with methods that lack topological information and indicator correlation analysis, it has obvious advantages in accuracy. At the same time, due to the aggregation and simplification of alarm redundancy information in the alarm feature extraction process, the common point analysis logic itself is relatively concise, leading to this method having obvious advantages over knowledge graphs and other methods.

4. Dynamic Operation and Maintenance Map Construction

The overall architecture of the dynamic operation and maintenance map is shown in Figure 3, which is divided into the data layer and service layer.

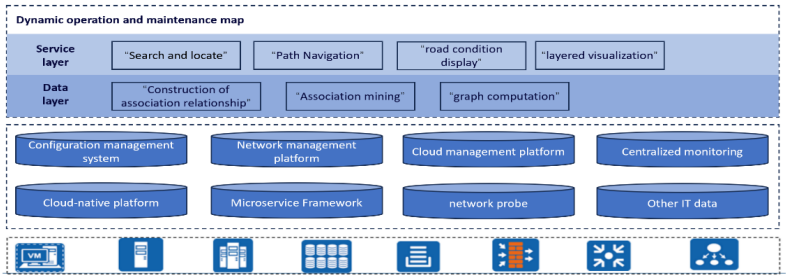

Figure 3. Dynamic map architecture.

The data layer includes data modeling, data access, data processing, and other steps, while the service layer provides data interfaces and visual displays.

1. Unify map data modeling [10–13]: The data model is divided into node and edge design. In terms of the node model, clarify the node types and attributes of each node type, including unique KEYs (this method uses IPs for alarm point attachment). Node types are obtained by refining and combing the operation and maintenance object directory of distributed information systems, covering applications, systems, networks, devices, etc., including but not limited to applications, application groups, middleware, PODs, virtual machines, physical machines, network devices, etc. In terms of the relationship model, it is mainly divided into two categories, one is the application access level topology, as shown in Figure 4. It mainly describes the call relationship between applications, groups, and services. The other is the vertical resource dependency vertical topology, as shown in Figure 5. It mainly describes the deployment and operation dependency relationship from large to small at the resource level.

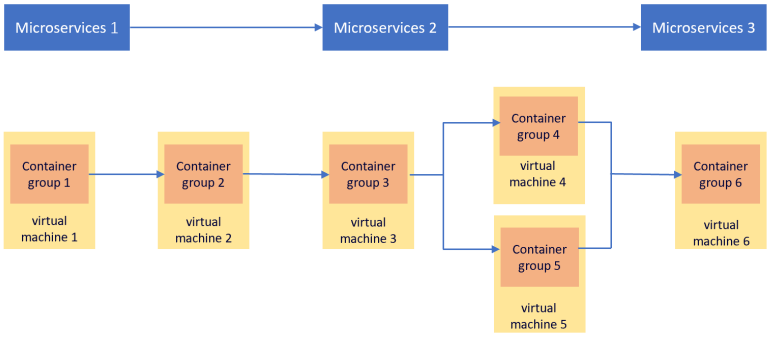

Figure 4. Application access level topology data model.

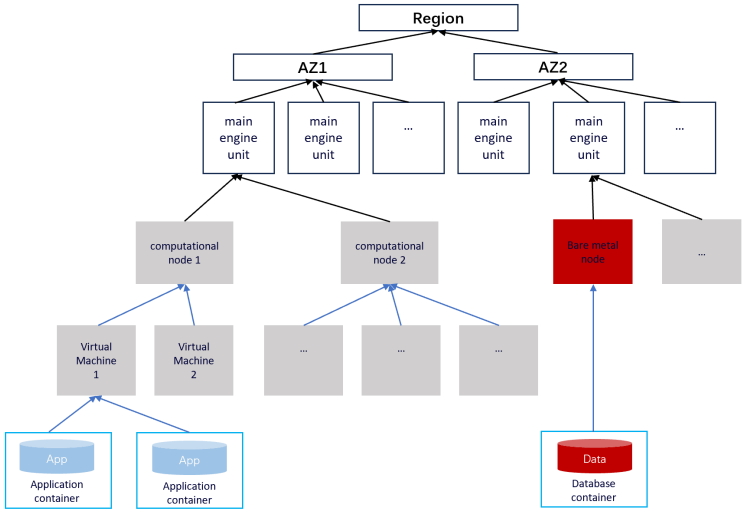

Figure 5. Resource-dependent vertical topology data model.

2. Data access [14,15] mainly includes four types of data, of which configuration management and network management mainly provide static basic device information data, used to build the "skeleton" of the basic environment, while network TCP flow and microservice call logs provide IP-level IT component intercommunication, as well as dynamic information of business calls, used to reflect the actual business deployment situation on top of the "skeleton".
3. Data processing [16,17]: According to the four layers defined by the data model and the constraints between objects, information extraction and entity alignment are completed based on the multi-source data of the access module. In the process of data fusion, potential relationships are mined through intelligent algorithms for some missing relationships. Taking network switches as an example, some old devices cannot support neighbor discovery protocol (LLDP/CDP) data collection, resulting in missing topology connections. However, after introducing a similarity judgment intelligent model, the port connection relationship can be established by calculating the similarity of traffic rate trends on different ports. After introducing network simulation capabilities, cross-network connection relationships between system nodes can also be established through route table entry calculations. With the help of algorithms, the integrity of map data can be further improved, and self-learning capabilities can be built.
4. Data interface access and visualization display [18,19]: Based on graph query, they provide data access interfaces and provide view display.

5. Multi-Source Alarm Screening and Feature Extraction

Providing various professional alarms presents the characteristics of multi-source heterogeneity. The financial industry represented by large banks has realized the aggregation of various professional alarms through centralized monitoring. Centralized monitoring uses complex redundant fields to accommodate various professional alarm information. Taking the centralized monitoring of a bank as an example, there are more than 50 field types, which bring redundant information. In addition, in daily operation, the average daily alarm volume reaches thousands of levels. After an abnormal business, there are still alarms unrelated to the business during the fault period, which brings interference. To address the above issues, the methods in this section aim to screen and extract features [20,21] for multi-source alarms.

(1) Filtering

Filtering refers to the process of identifying and selecting alerts that are relevant to a particular anomalous business event within a fault period, utilizing a sliding window approach, in order to mitigate interference from irrelevant alerts.

Sliding window technology [22] is a widely used technique in spatio-temporal sequence data processing. It can slide a fixed-length window in the data stream and perform uninterrupted real-time processing. Generally speaking, by using sliding windows, we can avoid computing or storing the entire data stream, thereby achieving efficient and useful results.

It includes the following two steps:

a. Window sliding: based on a sliding window W (W = 15 min), scanning centralized monitoring alarm information $(x_1, x_2, ..., x_n)$ at fixed intervals (every minute), mainly selecting abnormal application-related alarms and infrastructure-level alarms related to the application.
b. Data aggregation calculation: For selected alarms, we obtain the IP address cluster where the alarm is located and perform deduplication processing on it.

(2) Feature structure

The goal is to construct key alarm features, which can not only avoid the interference caused by redundant information, improve the performance of subsequent alarm analysis, but also reflect various types of associated information of alarms, and improve the coverage of fault types. Considering that fault localization is mainly used to find the IP information of the faulty component that caused the problem, based on the high availability characteristics of distributed architecture, through isolation, expansion, restart, and other emergency operations, the stable operation of the business can be restored on the first attempt. This scenario essentially defines a fault range (IP set), rather than finding the detailed root cause of a certain operation and maintenance object. Based on this, the most critical thing is the IP information of the alarm object. For the alarm description information of the object itself, due to the vast differences in fault root causes, it is difficult to effectively analyze in a short time, so this semantic information is excluded from the alarm features.

At the same time, in order to better express the correlation between alarm objects and ensure the coverage of fault types, the methods in this section expand various related operation and maintenance object IPs based on read-write I/O streams, database streams, and basic component inter-access streams as features.

For read/write I/O streams, first, we deploy a relationship to investigate whether the application side abnormal container is located on the same host machine. Second, we investigate the connection relationship between the virtual machine and the centralized storage to identify whether the virtual machine failure is caused by a storage node failure. Third, we investigate the connection relationship between the leaf switch and the storage to identify whether the network side switch failure is caused by storage node failure.

For database access flow, firstly, we deploy a relationship to investigate whether the application side exception container is located on the same host machine. Secondly, we

investigate whether the exception container is connected to a database through an access relationship. Thirdly, we investigate whether different database servers are connected to the same network switch through the connection relationship to investigate the failure of the network side switch.

For the basic component intercommunication flow, firstly, we check whether the abnormal container accesses the same basic component (message middleware, cache database) by deploying the relationship. Secondly, we check whether different basic components access the same leaf switch and check the failure of the network side switch.

In summary, the multidimensional characteristics of the designed alarm flow table are shown in Table 2. It can be seen that the information about the associated objects in this flow table is currently empty and needs to be obtained in real-time through dynamic operation and maintenance maps in the future.

Table 2. Multi-dimensional feature table field.

Field	Meaning
Timestamp	Sliding window timestamp
Alarming IP	The IP address where the alarm occurred
Application of	The application to which the alarm IP address belongs
Container vessel IP	Container IP address
virtual machine IP	Virtual machine IP address
host machine IP	IP address of the host
Storage pool ID	Distributed storage information associated with the virtual machine
Information of the leaf switch connected to the application server SW1 IP	Uplink switch of application server
Database accessed by the application server IP	Database information accessed by the application server
Uplink leaf switch information of the accessed database server SW2 IP	The uplink switch of the database being accessed
SW1 is directly connected to spine switch SW3 IP	Spine switchboard SW3
SW2 is directly connected to spine switch SW4 IP	Spine switchboard SW4
Cache database accessed by application server IP	Redis
Message-oriented middleware accessed by application servers IP	Kafka

6. Fault Location Based on Map Addressing and Alarm Commonality Analysis

This section enriches the information of the associated objects in the alarm sequence flow table by querying the dynamic operation and maintenance map in real-time, and performs commonality analysis on them to obtain the final ranking of the fault objects.

Considering that the map is 2D, it primarily comprises nodes and edges. Nodes represent entities such as containers, virtual machines, servers, databases, and switches, uniquely identified by IP addresses. Edges represent various relationships including access, operation, and dependency. Map addressing refers to the process of querying the map based on an alarm IP to obtain associated node IPs, thereby enriching the alarm flow table.

The input for commonality analysis is the enriched alarm flow table. Drawing inspiration from word frequency statistics, a sliding window approach is adopted to conduct statistical analysis based on the frequency of IP occurrences, outputting the top-ranked IPs. These IPs are then checked for alarms, and if any are found, the final localization ranking is the output. See Algorithm 1 for details.

Algorithm 1 Common point analysis of map addressing and alarming

Input: Original alarm flow table: T_{origin};
Dynamic operation and maintenance map: M;
Time window: W;
The number of common value columns to be detected: n
Output: Abnormal IP ranking collection

1. Initialization: Before each round of algorithm iteration, initialize the alarm flow table with some field alarm table sets in $T_{ini} <= T_{origin}$
2. while True do
3. Take the initialized original alarm flow table T_{ini} at the current time
4. Extract the "IP address" field of the alarm in the alarm flow table T
5. Query the dynamic operation and maintenance map based on the "IP address" field of the alarm, and obtain the associated information table.
6. Combine the original alarm flow table T_{ini} and the associated information table $T_{relationd}$ to obtain T_{all}
7. According to the time window W
8. For $i = 1$ to n do
9. Query the common value of each attribute from the merged alarm flow table T_{all}, and obtain the set attribute: attribute1, IP: count, ..., attribute: attribute n, IP: count S.
10. For the set S, for attributes 1 to n, find out whether the corresponding IP address appears in the alarm IP
11. If it exists, it hits an exception
12. For the results with abnormal hits, the ranking of abnormality degree is based on the number of times the IP appears
13. end for

return Return Abnormal IP ranking collection

7. Experiment and Analysis

7.1. Experimental Environment

Three servers are used: Intel(R) Xeon E5-2650v2 mailto: CPU@2.60 GHz, 32 GB memory; operating system: CentOS 7.3; Inspur, Jinan, China. one for deploying the anomaly localization model, which is implemented in Python and divided into four program modules according to algorithm logic: alarm monitoring and scanning, flow table insertion, map addressing and flow table enrichment, and flow table common point search, with anomaly localization result output; one for deploying the graph database Neo4j to save dynamic operation and maintenance map topology relationship data; and one for deploying MySQL to save alarm flow tables. See Table 3 for details.

Table 3. Environment configuration.

Serial Number	Type	Configure	Component Deployment
1	Virtual Machine	16C/32G memory storage/500G	MySQL
2	Virtual Machine	16C/32G memory storage/500G	Neo4j
3	Virtual Machine	16C/32G memory storage/500G	Deploy positioning model

7.2. Data Set

This study focuses on a pivotal payment-related business system of a bank based on the microservice architecture. Utilizing a chaos engineering platform in a test environment, nine infrastructure failures (encompassing network, system, database, and storage) were simulated. The system's components, including applications, operating systems, networks, databases, and storage, are equipped with individual monitoring systems, and all monitoring alerts converge into a centralized monitoring system. The dataset for this research comprises the centralized monitoring alert events from the 15-min period preceding the occurrence of the aforementioned nine types of failures. The test dataset essentially represents

a tabular record of various component alert information during fault periods. The fields within this table encompass critical information such as time, the IP address of the alerting node, the object of the alert, and a descriptive narrative of the alert event itself.

Details are shown in Table 4.

Table 4. Data set.

Fault Type	Data Set Failure Cases	CData Scale
Networking	Core switch hardware failure	1.32 Ten thousand/15 min
	Hardware failure of aggregation switch	1.41 Ten thousand/minutes
	Firewall exception	1.25 Ten thousand/minutes
	Abnormal hardware of access switch	1.18 Ten thousand/minutes
	Load balancing failure	1.25 Ten thousand/15 min
The system	Abnormal computing node	1.51 Ten thousand/minutes
Database	Mutual visit database exception	1.18 Ten thousand/minutes
	Cache exception	1.23 Ten thousand/15 min
Storage area	Distributed storage exception	1.33 Ten thousand/minutes

7.3. Evaluation Indicators

The fault location mainly focuses on the hit rate, which is the ratio of hits to all faults. In addition, the fault location also focuses on timeliness, which is the time consumption of analysis. Therefore, the main indicators are the hit rate and analysis time consumption.

7.4. Experimental Protocol and Analysis

(1) Experiment Triggering

To facilitate the experimentation, the test dataset encompassing the aforementioned nine fault categories was imported into the backend's historical alert table. Concurrently, a real-time alert table with an identical table structure was established. Upon commencing the testing of a specific fault type, a timed script, running every minute, was utilized to sequentially select the minute-by-minute test data for that fault type and insert it into the real-time alert table. Additionally, the actual insertion time was updated in the timestamp field of the real-time alert table.

The proposed model in this paper scans the real-time alert table every minute, retrieves the latest one-minute alert events, and performs fault localization analysis. Through this approach, the simulation of faults is achieved, thus triggering the experimental procedures.

(2) Scene Design

Design the following two experimental scenarios, focusing on comparing hit rates and positioning timeliness.

Scenario 1: Pay attention to the module for filtering and screening multiple source alarms, and compare the timeliness of the fault localization program running under the condition of canceling the module and under normal conditions.

Due to the alarm storm problem during business failures, it will cause performance pressure on the troubleshooting system. Therefore, before the alarm data are accessed by the troubleshooting system, the screening and filtering of alarms is critical. Time T is the trigger time of the business alarm.

For this scenario, we utilized the test dataset pertaining to Fault Case 4, which focuses on the anomaly of an access switch. The experiment was conducted in two phases, with each phase utilizing the fault data to simulate fault recording and playback by inserting an empty real-time alert table for a duration of 15 min. In the first phase, the alert filtering logic was omitted from the model program. Subsequently, in the second phase, the alert filtering logic was incorporated into the model program, enabling a comparative analysis of the results.

The test results are shown in Table 5, Figures 6 and 7.

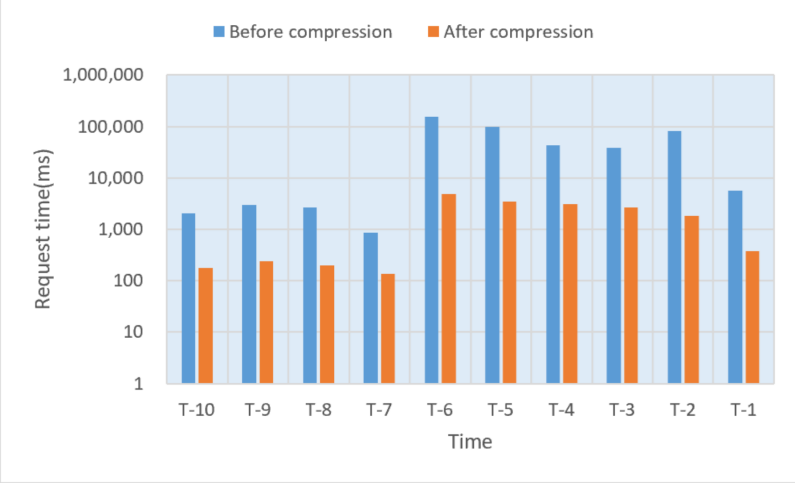

Figure 6. Request time comparison.

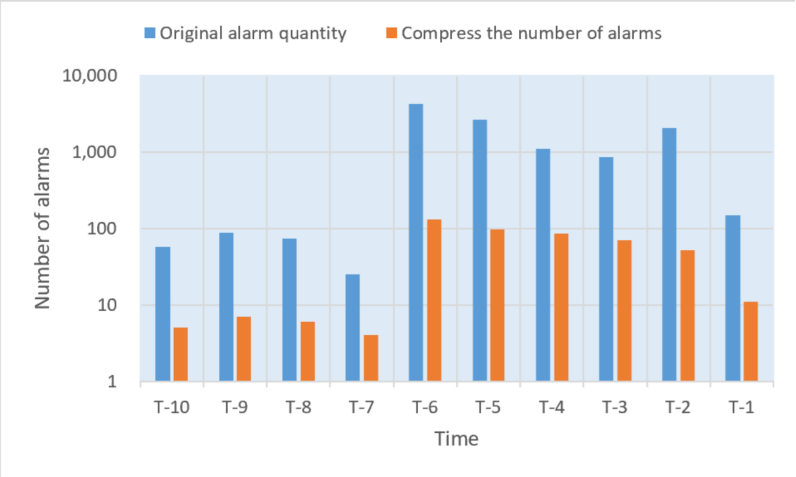

Figure 7. Number of alarms comparison.

Table 5. Data set

Time (Minutes)	Original Alarm Quantity (Canceling the Multi-Source Alarm Screening Step)	Compressed Alarm Volume (Adding Multiple Source Alarm Screening Steps)	Compression Ratio	Cancel the Multi-Source Alarm Screening Step Request Time (ms)	Increase the Request Time for the Multi-Source Alarm Screening Step (ms)
T-10	58	5	8.6%	2030	174
T-9	88	7	7.9%	2992	238
T-8	73	6	8.2%	2701	198
T-7	25	4	16%	873	136
T-6	4248	131	3.1%	152,928	4847
T-5	2658	97	3.6%	98,124	3492
T-4	1118	86	7.7%	42,484	3096
T-3	869	71	8.2%	38,642	2698
T-2	2094	52	2.5%	81,608	1820
T-1	150	11	7.3	5551	374
T	11,381	470	4.1%	427,933	17,073

Test conclusion: At time T, the business-level life and death indicator alarm was triggered. Tracing back 10 time slices from time T, statistical analysis showed that the multi-source alarm screening step was canceled. At time T, the fault localization program

received a total of 11,381 alarms, taking a total of 428 s. If the multi-source alarm screening step was added, the localization program received a total of 470 alarms at time T, taking a total of 17 s. Through comparison and verification, the alarm screening achieved an alarm compression ratio of 3.99%, with the time consumption falling from 428 s to 17 s, a reduction of 96%. This greatly improved the program's performance.

Scenario 2: Remove some topological relationship data from the dynamic operation and maintenance map and run the positioning program.

This experiment analyzed a total of nine failure case data covering four categories: network, system, database, and storage. Among them, the relationship data of aggregation switches were removed from test case 4 to construct test case 4#, the relationship data of core switches were removed from test case 5 to construct test case 5#, and the relationship data of cache were removed from test case 7 to construct test case 7#. The location program was then run.

In this scenario, nine fault case datasets were employed, among which the datasets of Fault Case 4, 5, and 7 required repeated recording and playback, twice each. During the second playback of these fault cases, the topological data related to the dynamic map was excluded. Therefore, a total of 12 rounds of experiments were conducted, each lasting 15 min, to facilitate the analysis and comparison of experimental data.

The test results are shown in Table 6.

Test conclusion: From the 12 test cases, it can be verified that whether the test results are successfully located depends directly on whether the dynamic map provides the configuration topology data corresponding to the fault type. The type of dynamic map topology data directly determines the fault types covered by the location.

(3) Experimental analysis conclusion

In summary, the two key modules of this model—alarm screening and dynamic map addressing are extremely critical. Alarm screening effectively improves the timeliness of hits, while the completeness of the topology data type of the dynamic map directly determines the fault coverage and hit rate. The experimental results demonstrate the effectiveness of this model method.

Table 6. Scenario 2 test results.

Test Case	Computational Node	Distributed Block Storage	Access Switch	Converged Switch	Core Switch	DataBase	Cache Memory	FireWalls	Load Balancing	Failure Scenario	Positioning Success
Testing 1	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	computational node	Yes
Testing 2	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No	Distributed block storage	Yes
Testing 3	No	Yes	Yes	No	No	No	No	No	No	access switch	Yes
Testing 4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Converged switch	Yes
Testing 4#	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Converged switch	No
Testing 5	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Core switch	Yes
Testing 5#	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Core switch	No
Testing 6	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	mutual visit database	Yes
Testing 7	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Cache memory	Yes

Table 6. *Cont.*

Test Case	Computational Node	Distributed Block Storage	Access Switch	Converged Switch	Core Switch	DataBase	Cache Memory	FireWalls	Load Balancing	Failure Scenario	Positioning Success
Testing 7#	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Cache memory	No
Testing 8	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Firewalls	Yes
Testing 9	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Load balancing	No

7.5. Application Effect Analysis

This method has been applied in a bank's key personal settlement business, analyzing real-time alerts related to the business, providing timely positioning results of abnormal objects during malfunctions and sending them to operation and maintenance personnel via email. In addition, a large-screen view has been developed, which supports drilling down to view the running details of suspicious objects. The actual positioning accuracy rate of this method is about 82%, and the analysis time is less than 2 min.

8. Conclusions

In the distributed architecture, due to resource pooling, the vertical mapping relationship between applications and the infrastructure layer is extremely complex, making it difficult to effectively drill down from the abnormal application to locate the abnormal object in the infrastructure layer, thus reducing the efficiency of the emergency response. This paper studies a model method of vertical infrastructure-level localization and provides a systematic solution. In terms of positioning the longitudinal foundation layer, current research focuses on methods such as knowledge graph and indicator correlation analysis. Knowledge graph technology has high complexity and is only effective in local scenarios such as networks, leaving a gap between practical use. Indicator correlation analysis lacks configuration correlation information between operation and maintenance objects, limiting accuracy. This paper proposes a foundation layer fault positioning method based on dynamic operation and maintenance maps and alarm commonality analysis. During the fault period, various alarm entities are associated with the dynamic operation and maintenance map, and the commonality of alarms is obtained based on graph search addressing, covering deployment relationship commonality points, connection commonality points (physical, logical), access flow commonality points, etc., thus positively locating the possible root cause of the fault. This method is more concise and efficient than the knowledge graph method and is more accurate than the indicator correlation analysis method due to the addition of configuration correlation information. It has been implemented in a key business scenario of payment type in a bank's actual production environment, with an alarm positioning accuracy rate of over 82% and an analysis time of less than 2 min.

Author Contributions: Conceptualization, S.W.; Methodology, S.W.; Software, S.W.; Resources, J.G.; Writing—original draft, S.W.; Writing—review & editing, J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets presented in this article are not readily available because the data are from a 3rd party business.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, S.; Wang, J.; Rong, J. Design-Oriented product fault knowledge graph with frequency weight based on maintenance text. *Adv. Eng. Inform.* **2023**, *58*, 102229. [CrossRef]
2. Subagdja, B.; Shanthoshigaa, D.; Wang, Z.; Tan, A.-H. Machine Learning for Refining Knowledge Graphs: A Survey. *Acm Comput. Surv.* **2024**, *56*, 1–38. [CrossRef]
3. Li, Z.; Li, Y.; Sun, Q.; Qi, B. Bearing Fault Diagnosis Method Based on Convolutional Neural Network and Knowledge Graph. *Entropy* **2022**, *24*, 1589. [CrossRef]
4. Zhou, Y.; Lin, Z.; Tu, L.; Song, Y.; Wu, Z. Big Data and Knowledge Graph Based Fault Diagnosis for Electric Power Systems. *EAI Endorsed Trans. Ind. Netw. Intell. Syst.* **2022**, *9*, e1. [CrossRef]
5. Li, M.; Li, Z.; Yin, K.; Nie, X.; Zhang, W.; Sui, K.; Pei, D. Causal Inference-Based Root Cause Analysis for Online Service Systems with Intervention Recognition. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Washington, DC, USA, 14–18 August 2022; pp. 3230–3240. [CrossRef]
6. Sun, S.; Chai, Z.; Wu, R.; Jin, J.; Wang, Y.; Xu, W.; Qi, G. Customer Complaint Guided Fault Localization Based on Domain Knowledge Graph. In Proceedings of the Database Systems for Advanced Applications: 28th International Conference, DASFAA 2023, Tianjin, China, 17–20 April 2023; pp. 568–579. [CrossRef]
7. Zhuang, B.; Shen, C.; Reid, I. Training Compact Neural Networks with Binary Weights and Low Precision Activations. *arXiv* **2018**, arXiv:1808.02631.
8. Su, Y.; Zhao, Y.; Xia, W.; Liu, R.; Bu, J.; Zhu, J.; Cao, Y.; Li, H.; Niu, C.; Zhang, Y.; et al. CoFlux: Robustly Correlating KPIs by Fluctuations for Service Troubleshooting. In Proceedings of the 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), Phoenix, AZ, USA, 24–25 June 2019; pp. 1–10. [CrossRef]
9. Liu, P.; Zhang, S.; Sun, Y.; Meng, Y.; Yang, J.; Pei, D. FluxInfer: Automatic Diagnosis of Performance Anomaly for Online Database System. In Proceedings of the 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), Austin, TX, USA, 6–8 November 2020; pp. 1–8. [CrossRef]
10. Brenner, M.; Gillmeister, M. Designing CMDB data models with good utility and limited complexity. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; pp. 1–15. [CrossRef]
11. Saenz-Core, J.; Vicente, E.J.F.; de la Cámara, M. A CMDB Meta Model Based on Services. In *Integrated Spatial Databases*; Springer: Berlin/Heidelberg, Germany, 2011.
12. Bonifati, A. The Quest for Schemas in Graph Databases (keynote). In Proceedings of the International Workshop on Data Warehousing and OLAP, Santiago, Chile, 22–26 May 2023.
13. Crowe, M.K.; Laux, F. Graph Data Models and Relational Database Technology. *arXiv* **2023**, arXiv:abs/2303.12376.
14. Luaces, D.; Viqueira, J.R.R.; Cotos, J.M.; Flores, J.C. Efficient access methods for very large distributed graph databases. *Inf. Sci.* **2021**, *573*, 65–81. [CrossRef]
15. Vela, B.; Barca, J.M.C.; Cáceres, P.; Sierra-Alonso, A.; Cuesta, C.E. Using a NoSQL Graph Oriented Database to Store Accessible Transport Routes. In Proceedings of the EDBT/ICDT Workshops, Vienna, Austria, 26 March 2018.
16. Eldin, A.N.; Assy, N.; Kobeissi, M.; Baudot, J.; Gaaloul, W. Enabling Multi-process Discovery on Graph Databases. In Proceedings of the International Conference on Cooperative Information Systems, Bozen-Bolzano, Italy, 4–7 October 2022.
17. Adoni, H.W.Y.; Nahhal, T.; Krichen, M.; Aghezzaf, B.; Elbyed, A. A survey of current challenges in partitioning and processing of graph-structured data in parallel and distributed systems. *Distrib. Parallel Databases* **2019**, *38*, 495–530. [CrossRef]
18. Klein, K.; Sequeda, J.; Wu, H.Y.; Yan, D. Bringing Graph Databases and Network Visualization Together (Dagstuhl Seminar 22031). *Dagstuhl Rep.* **2022**, *12*, 67–82.
19. Wu, H.Y.; Klein, K.; Yan, D. Effective Network Analytics: Network Visualization and Graph Data Management. *IEEE Comput. Graph. Appl.* **2023**, *43*, 10–11. [CrossRef]
20. Kuraku, N.V.P.; He, Y.; Ali, M. Comparative Analysis of the Fault Diagnosis in CHMLI Using k-NN Classifier Based on Different Feature Extractions. In *Machine Learning Paradigms*; Springer: Berlin/Heidelberg, Germany, 2018.
21. Kaplan, K.; Kaya, Y.; Kuncan, M.; Mínaz, M.R.; Ertunc, H.M. An improved feature extraction method using texture analysis with LBP for bearing fault diagnosis. *Appl. Soft Comput.* **2020**, *87*, 106019. [CrossRef]
22. Zeng, Z.; Cui, L.; Qian, M.; Zhang, Z.; Wei, K. A survey on sliding window sketch for network measurement. *Comput. Netw.* **2023**, *226*, 109696. [CrossRef]