Mining Causality Graph For Automatic Web-based Service Diagnosis

Xiaohui Nie†, Youjian Zhao†, Kaixin Sui†, Dan Pei†*, Yu Chen‡, Xianping Qu‡
†Tsinghua University ‡Baidu Inc.
†Tsinghua National Laboratory for Information Science and Technology (TNList)

***Abstract*— It is crucial for Internet company to provide highly reliable web-based services. The web-based services always have many components running in the large-scale infrastructure with complex interactions. As an indispensable part of high reliability, the diagnosis remains to be a thorny problem. With the growth of system scale and complexity, it becomes even more difficult. In this paper, we propose an automatic diagnosis system based on causality graph to help system operators find the root causes. The causality graph is mainly extracted from the historical data of the monitoring system, and the method consists of four steps. 1) It utilizes a data mining method to extract the initial causality graph. 2) Once a failure happens, it lists top-k suspects with a ranking algorithm based on the causality graph. 3) Then system operators check the suspects and label them either right or wrong. 4) A supervised learning algorithm takes the labels as the input to tune the causality graph, in order to improve the diagnosis accuracy on step 2 iteratively. This method requires neither knowledge about the design and implementation details of the web-based service, nor instrumenting the services' source code. Our controlled experiments show that the root causes can be ranked in top 3 with 100% accuracy after countable learning iterations.**

I. Introduction

Web-based services have permeated into people's daily lives, such as search engines, e-commerce, *etc.* Failures or malfunctions in these services would cause great economic losses and even affect people's daily life. For example, Amazon.com went down for 45 minutes in 2013, which cause 5 Million loss in business revenue [1]. Hence, quick and precise diagnosis for web-based service is crucial.

Diagnosing the web-based service in large-scale Internet company is challenging, because the system scale is becoming larger, more complex, and evolving faster over time. Failures are almost happening every week. Here we define the failure as KPI (Key Performance Indicator) becoming anomalous, such as PV (page view) decreasing rapidly, system's response time increasing dramatically, *etc.* Once a failure happens, a large cardinality of suspicious symptoms would be reported by the monitoring system, each suspicious symptom is recorded as a symptom event. Due to the large scale of symptom events, the root cause analysis (RCA) is hard to proceed precisely and completely. Operators of web-based services heavily rely on their domain knowledge and manual inspections to infer the root cause. However, this manual diagnosis is tedious, time-consuming, and error-prone, especially in large web-based services. Operators would dream about an efficient and automatic diagnosis tool.

System diagnosis is quite a common research [2–5]. Under a large, complex and dynamic system environment, the diagnosis for web-based services is still quite challenging. Pinpoint [5] and Shelock [4] use trace technology to acquire system components' dependency graph, but they can not be easily deployed because it needs instrument code in the software. Orion [6] and CauseInfer [7] use TCP latency as a clue to build the network application dependency, however, some symptoms in service level can not be correlated through TCP latency. We argue detailed diagnosis are needed. G-RCA [8] is a diagnosis system based on causality graph, it is designed for ISP. Its causality graph is configured by domain experts, but in web-based services, it is not feasible due to the large scale and dynamic environment. Some systems of web-based services could be developed by many teams, no one can hardly know the entire causality graph in the whole system. We believe the key issue of automatic diagnosis is to figure out a fine-grained causality graph. To the best of our knowledge, there is no generic method which can automatically build a precise causality graph for a dynamic and large scale web-based service with low overhead.

To address the problem above, our approach is from a different angle, aiming to learn the causality graph from domain experts' mind automatically. In fact, the causality graph is in domain experts' mind because they have rich experience in the system diagnosis. In the web-based service, the system operators are the domain experts. In order to translate the domain knowledge to a causality graph, we transform the problem into a supervised machine learning problem. We define the nodes in the causality graph as the symptom events and the edges in the causality graph as the rules which present the causality between two symptom events. Firstly, we employ a data mining algorithm to automatically mine the potential rules from historical data without application knowledge. The potential rules make up the initial causality graph. When a failure happens, our diagnosis system performs RCA automatically based on the causality graph, and it lists top-k suspicious root causes and their inference procedure. Then the system operators examine whether the diagnosis results are right or wrong in reality. What they need to do is simply labeling the results of RCA. The labels make up the training set for machine learning. At last, we select the

* Dan Pei is the corresponding author.

978-1-5090-5252-3/16/$31.00

Random Forest [9] as our machine learning algorithm. It trains an accurate classification model to improve the causality graph. This diagnosis architecture is generic, adaptive, self-learning and iterative. The latter component's results provide valuable feedback for the former component to learn and adjust parameters to achieve better diagnosis accuracy.

Our contributions can be summarized as follows:

- Our system is the first automatic root cause analysis framework for web-based service by learning the causality graph from domain experts. Iteratively applying data mining, root cause analysis, feedback and machine learning techniques to make the whole system a local feedback closed loop.
- We have implemented an effective label tool for system operators, not only by using a ranking algorithm to sort the suspicious root causes, but also making it easier to label the RCA's result for the learning causality graph.
- Our controlled experiment proves that our system can rank the root cause in the top 3 with 100% accuracy after a few times of learning iterations.

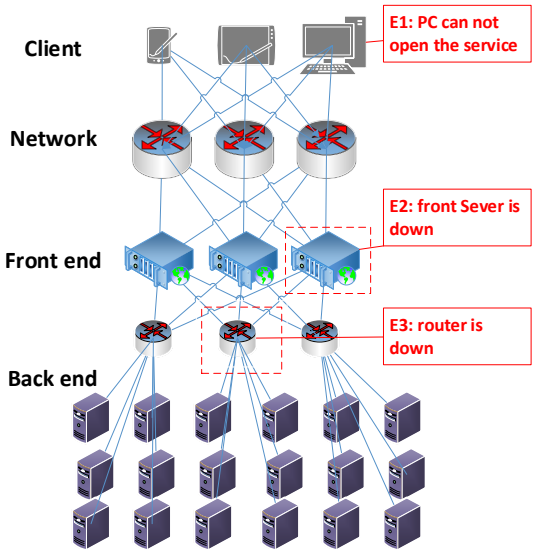


Fig. 1. Web-based service architecture. E1, E2, E3 are the symptom events. E1 means the system failure, the root cause could be E2 or E3. The relationship of E1,E2,E3 (E3 $\rightarrow$ E1, E2 $\rightarrow$ E1) makes up a simple causality graph.

II. System Overview

Web-based services have large-scale infrastructure and software, using commodity hardware and different networks. Figure 1 shows the architecture of the web-based service, which consists of network topology, front-end, back-end, *etc.* Many software components are complicatedly coupled. When the service's KPIs become anomalous, many symptom events would burst out due to complex coupling. According to the operators' domain knowledge, the relationships between these symptom events are the key clues for the system diagnosis, which make up the causality graph in this paper.

Figure 2 shows the overview of our system which can identify the root cause of failure and learn the causality graph

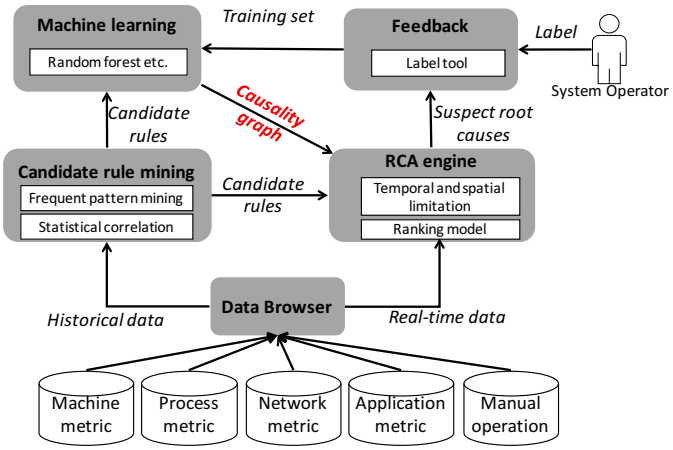


Fig. 2. System architecture

by data mining and machine learning methods. Our system is divided into five important components:

1) **Data Browser**: it collects the data that presents the status of the web-based services. The symptom events are generated by anomaly detection or status checking. They are the basic input of our system.

2) **Candidate rule mining**: it mines a list of candidate rules from the historical data and computes some correlation coefficients for the rules, which are the features for machine learning.

3) **RCA engine**: when a failure happens, it localizes most suspect root causes immediately for system operators based on the causality graph and the candidate rules.

4) **Feedback**: it is hard to label the tremendous rules directly. So here the operators can only label the rules in the RCA engine's result, which make up the training set for machine learning.

5) **Machine learning**: after gathering enough labels, it utilizes random forest algorithm [9] to build a more accurate causality graph based on the rules before. In this way, the accuracy of RCA's result can be improved.

Our system is a self-learning and iterative system. Although the web-based service system would be different after upgrades or other changes, we believe our system could learn to adapt to it, which would minimize operators' effort and company's loss. **Data Browser** is introduced in section III and **Candidate rule mining** is presented in section IV. Section V introduces the detail of **RCA engine**, **Feedback** and **Machine learning**.

III. Data Set

Symptom events are the basic clues for diagnosis in operators' daily life. In order to obtain the causality graph from operators' mind, we qualify the symptom events in operators' mind first. Table I shows the data collected in our system, including machine metric, process metric, network metric, application metric, and manual operation. The data is divided into two types: time series data and event sequence data. By using anomaly detection algorithm [10], we can

TABLE I
DESCRIPTION OF THE DATA METRIC. THE DATA IS DIVIDED INTO TWO TYPES: TIME SERIES AND EVENT SEQUENCE, EVENT SEQUENCE IS EQUAL TO 0 OR 1, 1 MEANS THE SYMPTOM EVENT HAS HAPPENED AND VICE VERSA.

Data metric	Event Description	Location	Type
Machine	CPU usage, memory usage, NIC, disk usage, context switch, *etc.*	Host	Time series
Process	CPU usage, memory usage, port status, file handle number, *etc.*	Process	Time series
Application	function return value, page view number, port status, error log number, *etc.*	Application	Time series, Event sequence
Network	network segment down, bandwidth decrease, *etc.*	Network	Time series, Event sequence
Manual operation	configuration upgrade, software upgrade	Operators' action	Event sequence

detect the abnormal points of time series data, and change it to sequence data. All the sequence data is the symptom events in our system, such as "CPU usage $>$ 80%", "memory usage $>$ 90%", "process port down", and "http port down". The instance's format of each symptom event is (Name, Time, Location, Type). Time represents the timestamp when the symptom event occurred. Location represents where the symptom event happened, such as host, cluster... *etc.* The type of symptom event can be process, network, application, manual operation... *etc.*

Because different data metrics have different properties, there is no general method of anomaly detection. So our strategies are CUSUM[10] and simple threshold detection. Manual operation events is the operation on web-based services, such as software upgrade, hardware upgrade, configuration upgrade, *etc.* Finally, our data browser collects most of the diverse data mentioned above. The data is shown on a graphical interface which includes a graph diagram of the software components as well as a chart of the data sequence of symptom events. It is a useful tool that provides operators with a global view of the service as well as enabling faster troubleshooting. The data is stored in Hbase and HDFS [11]. Hbase is used to store real-time data, and the HDFS is used to store the large scale historical data.

IV. MINING CANDIDATE RULES

It is challenging for operators to find some valuable causality rules by analyzing all the symptom events manually. Screening all the symptom events is a huge burden. Assuming there are n symptom events, the potential rules in causality graph is $A(n, 2)$. Operators can hardly label all the possible potential rules. Here we propose an automatic method to find out the candidate rules by mining the historical data, narrow down the scope of potential rules.

Definition of rule: E is symptom events unit, $A, B \in E, A \rightarrow B$ means A causes B and A happens before B. $\rightarrow$ presents the causality.

Here we transform this problem into a mining association rules problem. We utilize FP-Growth [12] to automatically dig out the valuable candidate rules from historical symptom events' data. Based on the mining result, we can generate a simple causality graph for diagnosis. Section V will introduce how to improve the causality graph. The mining job in this section can be divided into three parts. Section IV-A introduces the basic mining theory and the rules' features for machine learning. Section IV-B introduces how to compute rules' causality, and section IV-C describes how to decrease the redundant rules.

TABLE II
RULES' FEATURES TO EVALUATE THE CORRELATION

Feature($A \rightarrow B$)	**Description**
$Support$ [13]	The frequency of A, B's concurrence
C_1 [13]	Conditional probability:$P(B\|A)$
C_2 [13]	Conditional probability:$P(A\|B)$
$Pearson$ [14]	Novel statistical pearson correlation
$Lift$ [13]	$P(AB)/((P(A) * P(B)))$
$KULC$ [13]	$(P(A\|B) + P(B\|A))/2$
IR [13]	$P(A)/(B)$
Location relation	A, B happened in the same host, cluster, software component or not

A. Compute rule's feature

The symptom events' data is consisted of name, timestamp, location and type. Assuming we have n kinds of symptom events, let $E = \{e_1, e_2.....e_n\}$, e_i represents symptom event i. Here we sort the symptom events by time. Define a transaction t as a subset of E, and the time window is a constant number L. Due to our data properties, we defined L as ten times of sample time, which is 10 minutes in our system. Hence, the symptom events' data in history is a sequence of transactions.

In the mining procedure, symptom events are considered correlated when they occurred in the same transaction frequently. Assuming symptom event e_i, $e_j \subset E$, by using the FP-Growth algorithm [12], we can obtain two-stage results like e_i, e_j, e_i is associated with e_j. The association rule can be defined as $e_i \rightarrow e_j$ or $e_j \rightarrow e_i$. For example, when servers' memory usage is very high, the web-based service alway observes high response time, these two symptom events are supposed to be "high memory usage" and "high response time". The result of FP-Growth [12] could be a associate rule consisted of "high memory usage" and "high response time".

After mining from the historical data, the candidate rules in our system is a list of association rules. In the previous works, a threshold for a correlation coefficient should be chosen to classify whether the rule is right or wrong. However, this is not easy in reality, one correlation coefficient is not enough to explain the correlation of complex symptom events in web-based service. We transform this problem into a machine learning problem. Table II shows the features of $A \rightarrow B$ for machine learning. Hence, we transform how to decide the threshold of each feature into a classification problem. In order

to adapt to dynamic web-based service, the mining procedure starts at a fixed time every day.

The features in table II can be easily computed except for $Pearson$ [13]. Due to imperfect clock synchronization, and recording delays, the temporal information of each symptom event is not 100% reliable. $Pearson$ need to align the symptom events' timestamp. Here we devise a new method to calculate $Pearson$.

Assuming $R(E_1, E_2)$ is the association rule in figure 3, E_1 is the time series presenting symptom event "CPU usage $>$ 80%", E_2 presents symptom event "Software upgrade". We change the point event to range event with a time window, which is ten times of sample time interval. Then we devise a method to align the event. In figure 3, we use a time window and shifting method to align two events. These time series are divided into n time windows. In each time window, we shift E_2 to match E_1. If there are more than one symptom event's instance in a time window, we merge them into a range event, the alignment procedure is matching the middle point of the range event. The forward shifting number and the backward shifting number are recorded in all time windows. After this transformation, E_1, E_2 are converted to $\tilde{E_1}$, $\tilde{E_2}$. The $Pearson$ are defined as equation 1, $Pearson \in [-1, 1]$, the closer to 1 means the two events are more correlated.

$$Pearson = \frac{Cov(\tilde{E_1}, \tilde{E_2})}{\sqrt{Var(\tilde{E_1})Var(\tilde{E_2})}} \quad (1)$$

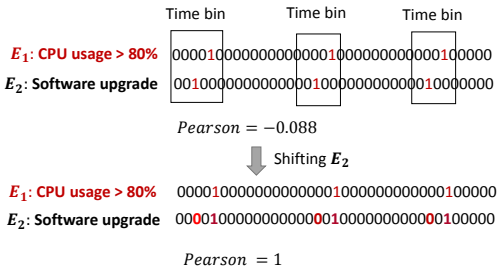


Fig. 3. After alignment, the $Pearson$ value shows E_1 and E_2 are truly correlated.

B. Compute rule's causality

The causality of the rules can hardly be decided by condition probability or other statistics parameters. However, the temporal information of events is not 100% accurate. [15] uses the lag correlation to decide the causality based on imperfect temporal information. Here we also utilize lag correlation as $Pearson$ correlation above. Its basic idea is comparing the forward and backward shifting number. Assuming two events E_1, E_2, if E_1 always happens before E_2 in each time window, we conclude E_1 causes E_2. If E_2 always happens before E_1, we conclude E_2 causes E_1. If the forward and backward shifting number are comparable, we conclude E_1 causes E_2 and E_1 can also causes E_2.

C. Decrease redundant rules

Here we use the symptom events' location information to decrease the redundant rules. Because it is not perfect by using FP-growth [12] to get the causality graph, some candidate rules are coincidence. There is a simple domain knowledge to constrict the rules: if two events' location in one rule has no relation, the rule can be eliminated. For example, A host's "memory usage $>$ 80%" and B host's "memory usage $>$ 80%" make up one association rule, but A and B host do not communicate directly or have no relationship at all, the associate rule is not right. In a real web-based service, the relation of symptom events' location can be configured by operators directly. It is useful for learning causality graph in the next sections.

V. Learning Causality Graph

A accurate causality graph is the key to automatic diagnosis. When a KPI becomes anomalous, the cause inference is triggered. We infer the root causes by searching on the causality graph. The result is a sorted root cause list. Ideally, the first one should be the real root cause. However, it depends the accuracy of causality graph. After mining the historical data, the candidate rule set is a relative coarse causality graph. So our system collects feedback from operators and learns from the feedback to improve the causality graph, in order to perform a accurate diagnosis. Actually, it is a supervised learning problem. Firstly, we use a general RCA engine to infer the root causes and rank the root causes with a ranking algorithm. Secondly, we design a human-computer interaction scheme to collect labels from operators. At last, we utilize machine learning technology to train a classify model for improving the causality graph. In this way, our system can learn the causality graph from operators after a few times of iteration.

A. Root cause analysis

As mentioned in section III, The instance of each symptom event is defined by (Id, Time, Location, Type).

Our system uses the causality graph to decide which instances are related to the others. When the KPI becomes anomalous, our system collects all the instances of the symptom events at accident scene firstly. The time range of accident scene is chosen by operators, the default is 30 minutes. Then a directed acyclic graph of symptoms events is generated based on the causality graph with spatio-temporal restriction. At last, our system lists the suspect root causes to operators. The simple procedure is below: assuming E_1, E_2, E_3, E_4 four events happened in the same time range, E_4 is the symptom event of a failure. If the causality graph contains $E_1 \rightarrow E_2$, $E_2 \rightarrow E_3$, $E_3 \rightarrow E_4$, the final result is $E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow E_4$. The root cause of E_4 is E_1.

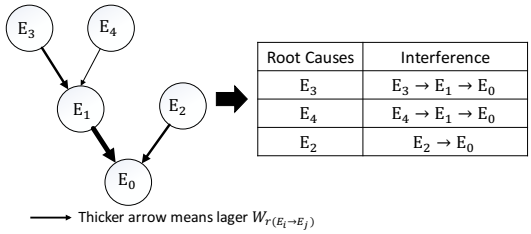
B. Root cause ranking

To be an efficient diagnosis system and accelerate the label procedure, our system uses a ranking mechanism to rank the most likely root causes in the top-k. We define $W_{r(e_1 \rightarrow e_2)}$ in equation 2 as the degree of $e_1 \rightarrow e_2$ being a right rule. $W_{r(e_1 \rightarrow e_2)}$ is equal to a default value in the initial phase, and it will be changed in each learning iteration.

$$W_{r(e_1 \to e_2)} = \begin{cases} 0.5, default \\ F(f_1, f_2, f_3, ...), F \in [0, 1] \end{cases} \quad (2)$$

f_1, f_2, f_3 in $F(f_1, f_2, f_3, ...)$ are the features' values of the rule, function F is the machine learning classify model. Each rule has the probability to be a right or wrong class in machine learning. Here $W_{r(e_i \to e_j)}$ is the probability of rule $e_i \to e_j$ being a right class. If $W_{r(e_1 \to e_2)} \geq 0.5$, class of $e_i \to e_j$ is right and vice versa.

When operator notices the symptom of anomalous KPI, he/she will check the most likely symptom event that directly causes the failure. So we propose a ranking method which simulates the manual diagnosis procedure in figure 4. It is a greedy depth-first search algorithm. The symptom event of anomalous KPI is E_0, our system firstly checks which symptom events would trigger E_0, so they are E_1, E_2. E_1 is ranked before E_2 because $W_{r(E_0 \to E_1)} > W_{r(E_0 \to E_2)}$. Then our system will check whether other symptom events would trigger E_1. In this way, the final ranking result is E_3, E_4, E_2. If the rules' weight W are equal, we randomly choose one symptom event to search its upstream nodes.



Root Causes	Interference
E_3	$E_3 \to E_1 \to E_0$
E_4	$E_4 \to E_1 \to E_0$
E_2	$E_2 \to E_0$

Fig. 4. A simple example of root cause ranking

C. Collecting labels

When a failure happens, a list of ranked suspect root causes will be presented along with the detail inference procedure. The operators check whether the result is right or wrong and give feedbacks. It is easy for them to label whether the root causes are right or wrong. Figure 5 is an example of RCA's result in our system, it is divided into three parts: "Root causes", "Inference" and some buttons. 1) "Root causes" lists the ranked suspect symptom events, and the max number of the root causes is 6 in our system; 2) "Inference" shows the rules that are used in root cause analysis. The arrows in the "Inference" can be clicked, they have three states: uncertain, confirmed, denied. At first, the states of the arrows are all uncertain. If the operators can make sure the rule is right, they can click on it and change the rule's state to the right state. Our system records all the click information which makes up the training set for learning; 3) The button "Refresh" is used to perform RCA again after giving feedbacks, if the operators label some rules wrong, clicking "Refresh" will generate a new result based on the new causality graph without wrong rules. In this way, it can improve the ranked result and accelerate the label procedure.

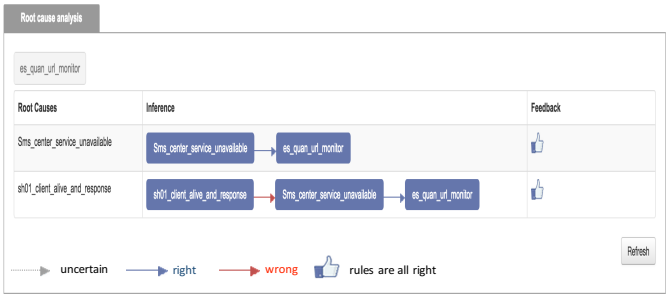


Fig. 5. The web UI of our label tool. It shows the RCA result of symptom event "es_quan_url_monitor". The root causes are symptom events "Sms_center_service_unavailable" and "Sh01_client_alive_and_response". Operator gives feedback by simply clicking the rules to change their state.

D. Machine learning

After collecting operators' feedback, we use a machine learning algorithm to train a classifier, which classifies the rules in the causality graph into right or wrong class. At last, we can get a more accurate causality graph.

Many supervised machine learning algorithm can be used in this paper, such as decision tree [16], NaiveBayes[16], RandomForest [9], RBFNetwork [16], Linear logistic [16]. We utilize Random Forest [9] as our machine learning algorithm. Table II is the rules' features which can be used to describe the rules' correlation. In section VI, we would explain why we choose Random Forest [9]. Actually it is relatively robust and works well for our problem. We implemented it by Weka [17].

VI. Evaluation

Because the ground truth of the web-based service can be hardly obtained in reality, we evaluate our system though a controlled experiment with explicit ground truth in this section. Section VI-A firstly describes the assumption of controlled experiment with a simple ground truth. In the next, the evaluation shows different machine learning algorithm's ability and why we choose Random Forest [9]. At last, our result shows the root cause can be listed in top-k (k = 3) after a countable learning iterations.

A. Controlled experiment

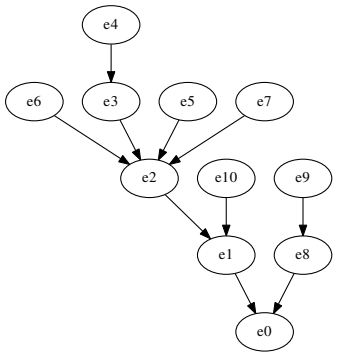


Fig. 6. Ground truth of a simple causality graph.

We firstly choose a simple ground truth of causality graph in figure 6 to evaluate our system. Section VI-B shows that our system works well with more complicate ground truths. Here e_0 is a failure event (anomalous KPI event) and the other symptom events ($e_1 \sim e_{10}$) will trigger the happening of e_0. The leaf nodes $e_4, e_5, e_6, e_7, e_9, e_{10}$ are the root causes. In reality, the noisy symptom events hinder system operators to uncover the actual root cause, which is a key reason why manual diagnosis is inefficiency. So we add some noisy symptom events that randomly co-occurred with failure event e_0 at a close time. Here we add $e_{11} \sim e_{29}$ as the noisy symptom events. The ground truth of causality graph is randomly generated as a L (here $L = 5$) layer directed acyclic graph(DAG). We believe our scheme of ground truth is suitable for most diagnosis cases of the real complicate web-based service. In fact, there are many failure events in a web-based service. Each failure event has a causality graph like figure 6. Our method is generic for each failure event. So here we firstly take one failure event as an example to prove our method works well.

Assumption:

- 1. The ground truth of causality graph is the domain knowledge in system operators' mind.
- 2. Root causes are the leaf nodes in the ground truth.
- 3. The direction of edge in causality graph means the causality.
- 4. When giving feedback, operators label each rule according to the ground truth. The rules in ground truth should be confirmed as right rules and vice versa.

1) Data Simulation: Here we take figure 6 as an example to describe how to simulate the data of the symptom events. e_0 is randomly triggered by root causes $e_4, e_5, e_6, e_7, e_9,$ or e_{10}. These symptom events are all happened in a close time, the minimize time gap between two symptom events is 10 seconds. For example, when e_9 happens, it will cause e_8 happening in 10 seconds later and e_8 will cause e_0 happening 10 seconds later. Besides, the noisy events ($e_{11} \sim e_{29}$) are randomly chosen to co-occur with e_0, e_8, e_9. In every 15 minutes, one root cause symptom event is randomly (equal probability) chosen to happen to cause e_0 happening. In this way, our RCA engine can perform root cause analysis and list the suspect root causes for the operators in every 15 minutes. At last, failure event e_0 are triggered to happen for about 2880 times (one month) in our simulation data. Our goal is to achieve a high accuracy diagnosis in this simulation data set.

2) Mining candidate rules: There are 30 kinds of symptom events in our simulation data. The number of potential rules between them is $A(30, 2) = 820$, but the number of right rules is only 10. Hence, we firstly use FP-growth algorithm to mine the candidate rules in the simulation data. The time window length of a transaction is 15 minutes, and $support > 20$ is the association rules' condition. At last, 91 candidate rules are generated, including 10 right rules in ground truth and 81 wrong rules. The ground truth are all included in candidate rules. In this way, we narrow down the scope of potential rules greatly, but the other wrong rules need to be filtered out from our causality graph, in order to realize a high accuracy diagnosis. In the next, we use machine learning to improve our causality graph.

TABLE III
THE CONFIGURATION OF FIVE MACHINE LEARNING ALGORITHMS IN OUR EXPERIMENT.

Algorithm	**Sampled Parameters**
J48 (Decision tree)	confidenceFactor = 0.25, minNumObj = 2, numFolds = 3, seed = 1
NaiveBayes	useKernelEsimator = false, useSupervisedDiscretization = false
Random Forest	maxDepth = newFeatures = 0, numTrees=100, seed =1
RBFNetwork	clusteringSeed = 1, numClusters = 2, minStdDev = 0.1, ridge = $1.0E-8$, maxIts = -1
Logistic	ridge = $1.0E-8$, maxIts = -1

TABLE IV
FN,TF,NP,NF OF EACH ALGORITHM AFTER LEARNING

Algorithm	**TP**	**FP**	**TN**	**FN**
J48	10	0	81	0
NaiveBayes	7	2	79	3
RandomForest	10	0	81	0
RBFNetwork	8	3	78	2
Logistic	8	1	80	2

3) Evaluation of machine learning algorithms: Different machine learning algorithms show different learning ability in our controlled experiment. Utilizing a proper machine learning algorithm is quite crucial in our system. Here we compared five major machine learning algorithms implemented in WEKA[17], and tables III presents the setting of our algorithm.

Machine learning should be triggered with collecting enough labels from operators. In our simulation, we start machine learning in every one hour to tune the causality graph, which means learning every 4 times of RCA. After machine learning, the causality graph will be changed, and the RCA in the future will be based on the new causality graph. Here we treat this procedure as one learning iteration. One good learning algorithm could increase the accuracy of causality graph after a fewer learning iterations. Table IV shows the true positive (TP), false positive (FP), true negative (TN), false negative (FN) after learning. Accuracy = (TP + TN) / (P + N). The result shows J48 [16] and Random Forest [9] can learn the ground truth with 100% accuracy. What's more, we also consider the learning speed. Figure 7 presents different algorithms' accuracy, Random Forest is the first one that converge with 100% accuracy. That's why we choose Random Forest as our learning algorithm and the result illuminates that our method can learn the ground truth with a high accuracy.

4) Evaluation of causality graph: After mining the simulation data, there are 10 right rules and 81 wrong rules in the causality graph. After each learning iterations, random forest keep training its classification model to improve the causality graph. Figure 8 shows the wrong rule number are dramatically decreased and the right rules set is equal to the ground truth.

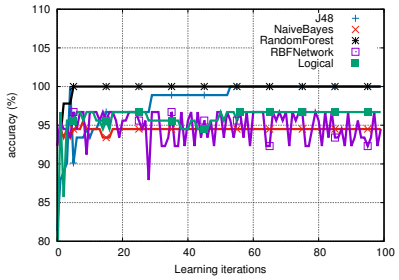


Fig. 7. Different algorithms' accuracy at different learning iteration, the x-axis means the iteration times of learning.

At last, the wrong rules are all eliminated from causality graph. It proves our system works well.

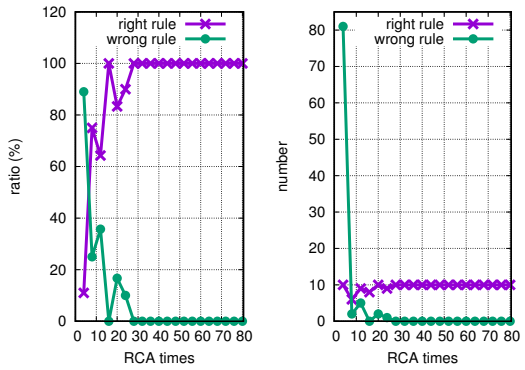


Fig. 8. The rules in causality graph. "RCA times" means the number of each diagnosis for failure event e_0.

5) Whether the root cause can be listed in top-3?: Our system's goal is listing the root cause in top-3. Figure 9 demonstrates that our system can learn to list the root cause in top-3 after a few times of RCA. At last, the ratio of the root cause in top-3 equals to 100%, that means our system can localize the root cause in top-3 with 100% accuracy. Besides, our system converges quickly after only 7 learning iterations (28 times of RCA).

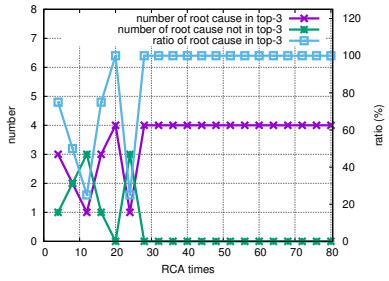


Fig. 9. The ratio of root cause in top-3.

B. Evaluation of different ground truth

All the above results are based on ground truth in figure 6. In this section, our experiment proves that our system can handle more complicate ground truth. We have performed the same experiment with different ground truth in table V. The rules and root causes are all randomly simulated. It is impossible to list all the ground truth in the real web-based services, but we believe our simulation scheme can represent the real ground truth to some degree. Assuming right rule number in causality graph is R_Y and wrong number is R_N, the rule number of ground truth is R_G. The right rule ratio equals to $R_Y/(R_Y + R_N)$ and the right rule coverage equals to R_Y/R_G. Figure 10 shows the right rule ratio and the right rule coverage in causality graph. After operators giving feedback on the results of about 300 times of RCA, the causality graph in our system can contain the entire ground truth. The result shows that our system can improve the ratio of the right rules, which means the wrong rules can be eliminated along with learning. Figure 11 proves that our system can list the actual root cause in top 3 with a certain learning iterations.

TABLE V
CONFIGURATION OF DIFFERENT GROUND TRUTH

Schemes	#Node	#Rule	#Root Cause
1	50	49	34
2	100	99	65
3	200	199	134

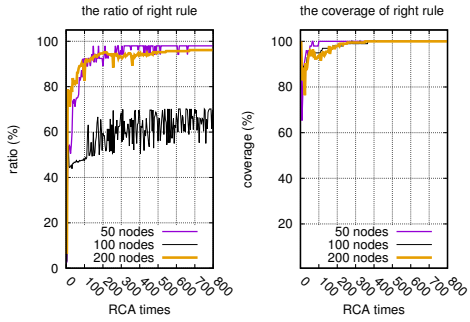


Fig. 10. The learning result of complex ground truth

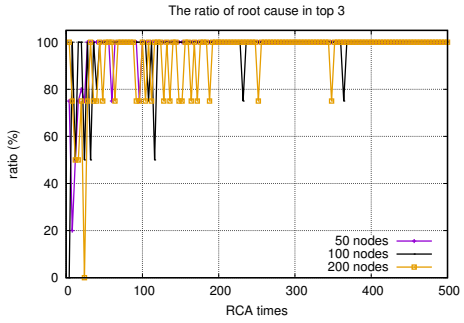


Fig. 11. The ratio of root causes in top 3.

VII. RELATED WORK

Many previous researches work on the large system diagnosis. Some methods are based on mining log [18–20]. These methods use log information to localize the root cause. They train the log's normal patterns in the history and use the pattern to detect the anomalies. Some works depend on the dependency graph. Pinpoint [5] and Shelock [4] use trace

technology to get the dependency graph of the distribution system. These tools record the execution path information and locate the root cause, but they need instrument code in software, the overhead is relative high for the web-based service. RPC technology [21] is also used to find service's dependency graph, but it has a high developing and analysis cost. Besides, dependency graph is different from causality graph, it mainly locates the root cause at service level. Some systems are based on causality graph. G-RCA [8] is designed for ISP, its causality graph is configured by domain expert, but it is quite hard for web-based service because of large scale and dynamic environment. Orion [6] and CauseInfer [7] use TCP latency as clue to build the network application dependency. NICE [14] and [22] use statistical test to figure out whether two events are correlated, the correlation is used to build the causality graph.

VIII. Conclusion

Automatic web-based service diagnosis is important for the Internet companies. It is closely related to the revenue and user experience. In this paper, we propose a generic diagnosis system for web-based services. It utilizes a data mining and machine learning method to build an appropriate causality graph automatically. Based on the causality graph, it realizes an automatic and high accurate diagnosis. It doesn't need detail application information or instrumenting code in the web-based service. What's more, it utilizes a feedback mechanism to learn the causality graph from system operators. After the operators label the RCA result, it can learn to improve the causality graph based on the operators' feedback. The more feedback collected in our system, the more accurate the causality graph would be. At last, we evaluate our system precisely in a controlled experiment. The controlled experiment shows that our system can list the root causes in top-3 with 100% accuracy after a certain learning iterations.

Acknowledgment

This work was partly supported by the State Key Program of National Science of China under grant No. 61233007, the National Natural Science Foundation of China (NSFC) under grant NO. 61472214 & No. 61472210, the National High Technology Development Program of China (863 program) under grant No. 2013AA013302, the National Key Basic Research Program of China (973 program) under grant No. 2013CB329105, the Tsinghua National Laboratory for Information Science and Technology key projects, the Global Talent Recruitment (Youth) Program, and the Cross-disciplinary Collaborative Teams Program for Science & Technology & Innovation of Chinese Academy of Sciences-Network and system technologies for security monitoring and information interaction in smart grid.

References

[1] S. K. Abudheen. Amazon.com goes down for 45 minutes, loses 5m in business. "http://techcircle.vccircle.com/2013/08/20/amazon-website-goes-down-on-monday-company-loses-5m-in-business". August 20, 2013.

[2] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. Adtributor: revenue debugging in advertising systems. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages 43–55. USENIX Association, 2014.

[3] Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl. Detailed diagnosis in enterprise networks. *ACM SIGCOMM Computer Communication Review*, 39(4):243–254, 2009.

[4] Paramvir Bahl, Ranveer Chandra, Albert Greenberg, Srikanth Kandula, David A Maltz, and Ming Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 13–24. ACM, 2007.

[5] Mike Y Chen, Emre Kiciman, Eugene Fratkin, Armando Fox, and Eric Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 595–604. IEEE, 2002.

[6] Xu Chen, Ming Zhang, Zhuoqing Morley Mao, and Paramvir Bahl. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *OSDI*, volume 8, pages 117–130, 2008.

[7] Pengfei Chen, Yong Qi, Pengfei Zheng, and Di Hou. Causeinfer: automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In *INFOCOM, 2014 Proceedings IEEE*, pages 1887–1895. IEEE, 2014.

[8] He Yan, Lee Breslau, Zihui Ge, Daniel Massey, Dan Pei, and Jennifer Yates. G-rca: a generic root cause analysis platform for service quality management in large ip networks. *Networking, IEEE/ACM Transactions on*, 20(6):1734–1747, 2012.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Hiep Nguyen, Yongmin Tan, and Xiaohui Gu. Pal: P ropagation-aware a nomaly l ocalization for cloud hosted distributed applications. In *Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques*, page 1. ACM, 2011.

[11] What is apache hadoop? "http://hadoop.apache.org/".

[12] Le Zhou, Zhiyong Zhong, Jin Chang, Junjie Li, JZ Huang, and Shengzhong Feng. Balanced parallel fp-growth with mapreduce. In *Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on*, pages 243–246. IEEE, 2010.

[13] Bing Liu Wynne Hsu Yiming Ma. Integrating classification and association rule mining. 1998.

[14] Ajay Mahimkar, Jennifer Yates, Yin Zhang, Aman Shaikh, Jia Wang, Zihui Ge, and Cheng Tien Ee. Troubleshooting chronic conditions in large ip networks. In *Proceedings of the 2008 ACM CoNEXT Conference*, page 2. ACM, 2008.

[15] Ajay Anil Mahimkar, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Qi Zhao. Towards automated performance diagnosis in a large iptv network. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 231–242. ACM, 2009.

[16] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York.

[17] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[18] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. Execution anomaly detection in distributed systems through unstructured log analysis. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 149–158. IEEE, 2009.

[19] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132. ACM, 2009.

[20] Jian-Guang Lou, Qiang Fu, Shengqi Yang, Jiang Li, and Bin Wu. Mining program workflow from interleaved traces. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–622. ACM, 2010.

[21] RFC. Rpc: Remote procedure call. http://tools.ietf.org/html/rfc1057. June 1988.

[22] Chen Luo, Jian-Guang Lou, Qingwei Lin, Qiang Fu, Rui Ding, Dongmei Zhang, and Zhe Wang. Correlating events with time series for incident diagnosis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1583–1592. ACM, 2014.