iSwift: Fast and Accurate Impact Identification for Large-scale CDNs

Jiyan Sun[1], Tao Lin[3,*], Yinlong Liu[1,2], Xin Wang[4], Bo Jiang[5], Liru Geng[1,2], Pengkun Jing[1,2], Liang Dai[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]State Key Laboratory of Media Convergence and Communication, Communication University of ChinaBeijing, China
[3]Communication University of China, Beijing, China
[4]Stony Brook University, New York, USA
[5]Shanghai Jiao Tong University, Shanghai, China
Email:sunjiyan@iie.ac.cn, lintao@cuc.edu.cn, liuyinlong@iie.ac.cn, x.wang@stonybrook.edu, bjiang@sjtu.edu.cn, gengliru@iie.ac.cn, jingpengkun@gmail.com, dailiang@iie.ac.cn
*Corresponding Author: lintao@cuc.edu.cn

***Abstract*—One key challenge to maintain a large-scale Content Delivery Network (CDN) is to minimize the service downtime when severe system problems happen (e.g., hardware failures). In this case, a critical step is to quickly and accurately identify the range of users with performance degradation, termed *impact identification*. Successful impact identification not only helps identify impacted users but also provides meaningful information for troubleshooting. However, current practice of impact identification usually takes network engineers several hours to manually identify impacted users, which may lead to a huge business loss. The main challenges for automatic impact identification in large CDNs include the inaccuracy of underlying anomaly detection, huge search space of impact identification and severe long-tail distribution of user traffic. In this paper we propose *iSwift*, a system that is specifically designed for impact identification in large-scale CDNs in order to address aforementioned challenges. We evaluate the performance of iSwift on semi-synthetic datasets and the results show that iSwift can achieve a F1-score greater than 0.85 within ten seconds, which significantly outperforms state-of-the-art solutions. Furthermore, iSwift has been deployed in a production CDN around one year as a pilot project and demonstrated its online performance confirmed by the network operators.**

***Index Terms*—impact identification, failure diagnosis, CDN**

I. Introduction

By intelligently caching popular contents in distributed edge nodes, Content Delivery Network (CDN) [1], [2] greatly improves the bandwidth and delay performance when users fetch data directly from edge servers. However, with the rapid growth of network scale and user traffic, the guarantee of continuous and high-performance services becomes an essential challenge in large-scale CDNs [3], [4].

Many factors may cause the interruption of user service in CDN. Some examples are unexpected hardware failures of servers, potential software bugs for the newly-upgraded user clients, congestion of network links, server overload, etc [5]–[12]. The interruption time of user services is termed the *service downtime*. Since the above factors are unavoidable and difficult to predict, the length of service downtime becomes a critical metric to evaluate the reliability of CDNs.

Current practice to minimize the service downtime mainly relies on the procedure of *impact identification* to identify the range of impacted users that are experiencing the performance degradation [13]–[16]. It helps switch the traffic of impacted users timely to backup servers or roll-back the software changes to recover the service [16]. Based on the log file information in CDN servers, Internet users of CDNs can be associated with some *attributes*, such as client OS, user location, requested web domain, etc.. Each attribute contains a set of elements. For instance, elements of the attribute Client OS include Andriod, Apple iOS, Windows and so on. When system fault alerts are issued or service failures are reported from users, impact identification will be triggered to identify the range of impacted users in terms of attribute combinations as soon as possible [13]–[16]. For example, the identified attribute combination of (Android, video.A.com) indicates that the users of video.A.com using Android App are impacted by a service failure.

However, it is non-trivial to achieve fast and accurate impact identification in large-scale CDNs. First, the log data collected from CDN servers is huge and high dimensional which leads to a large search space of impacted user. Current practice of impact identification usually takes network engineers several hours to manually analyze a large volume of log data, which is time-consuming and error-prone. According to a two-year issue report from a top China's CDN provider which we cooperate with, the average service downtime of more than 50 system failures reaches 3 hours, ranging from 8 minutes to 23 hours. Some typical issues are illustrated in Table I.

Second, it is difficult to accurately identify all the affected users due to the severe long-tail distribution of user traffic in real CDN. Our real-trace analysis shows that above 90% traffic volume comes from top-5 web domains. Existing solutions normally ignore the tail ones which are rather small relative to the total amount [14]. However, they are also very important for the CDN provider. In practice, the CDN provider needs to guarantee the high service stability and transmission performance for paid VIP customers, whose traffic may take a relatively-small percentage of the total traffic (i.e., located

978-1-6654-6824-4/22/$31.00 © 2022 IEEE

in the tail of the traffic distribution).

Great efforts have been made in previous work [13]–[15], [17]–[19] to look for the effective method for impact identification in different network systems other than CDN. However, two limitations render existing solutions inefficient and inaccurate for CDN system. The first limitation is that they mainly focus on the impact analysis while overlooking the data restrictions due to the long tail distribution [13], [14], [17], which however is found to be prevalent in CDN. Existing solutions locate the affected users by identifying the key component that makes major contributions to the largest change of the total traffic. However, when a system failure occurs, although the traffic belonging to the affected users changes dramatically, the total traffic may not change much. This is because the amount of traffic of the affected users usually takes a small percentage of the total traffic (i.e., the long-tail effect in CDN system). On the other hand, in the case of changes in the total traffic, it is mainly due to the normal fluctuation in the elephant traffic of certain top web domains, rather than the system anomalies in the tail ones. Another limitation is that the heuristic metrics used for impact identification in existing work [15], [18], [19] are sensitive to the inevitable errors of anomaly detection, which are possible to mask the real impacted users in practice.

In this paper, we design and prototype *iSwift*, a system that enables automatically perform fast and accurate impact identification in large-scale CDNs. The contributions are summarized as follows.

1) **Handling the Tail**. Rather than focusing on the largest change of total traffic, we derive a key intrinsic property of performance-degraded users, named *confidence lossless* that is independent of the traffic volume to achieve accurate impact identification of system anomaly. It can offer consistently-high performance regardless of the underlying long-tail traffic distribution in CDN systems (Section III-C1).
2) **Real-time Search**. We design novel pruning metrics with robust thresholds that are insensitive to the errors of underlying anomaly detection methods. They are combined with adaptive beam search to effectively reduce the search space. The fast search enables the online impact identification within ten seconds that contributes to essential business values for a CDN system to handle anomaly in real time (Section III-C3).
3) **Real System Validation**. We evaluate the performance of iSwift comprehensively using both synthetic and real data traces collected from one of the largest-scale CDNs in China. The evaluation demonstrates the advantages of iSwift over existing state-of-the-art solutions, and the results identified by iSwift are further confirmed by a CDN operator to validate its effectiveness. More importantly, iSwift has been deployed in a production CDN system around one year as a pilot project and demonstrated its online performance in handling real system anomalies (Section IV and V).

II. Motivation and Problem Statement

A. CDN Infrastructure and Real-world Issues

A typical CDN system usually consists of three types of network components, including edge nodes, center nodes and scheduling nodes. Each network node contains a number of servers. The edge nodes are the first-level caches close to end users. The center nodes are the second-level caches located at a regional center to maintain syncing data from the original sites. The scheduling nodes are responsible for distributing user requests to different edge nodes in a load-balanced way. Specifically, the end users will first query the nearest edge node to fetch data via HTTP connection. Upon receiving an HTTP request with a cache hit, the edge node will directly return the data to user. Otherwise, the edge node will query the center nodes for data. We present a list of real-world CDN accidents in Table I, where each issue is related with the failure category, failure symptom, impacted users in terms of faulty attributes, the root cause and service downtime.

B. An Example

In order to illustrate the problem addressed in this paper, we present a toy example considering only two attributes, i.e, visited web domains and CDN edge cache servers. Triggered by system fault alerts that the traffic volumes of several web domains drop suddenly, a finest-grained multi-attribute record table will be built as shown in Table II. The Key Performance Indicator (KPI) metrics, e.g., traffic volume, of different CDN edge cache servers and domains, are measured every minute. At the same time, there is an anomaly detection method running on these finest-grained KPI metrics to generate a forecast value and compare it with the measured value. A record will be marked as abnormal if the difference between its measured and forecast values is large. We can see records 3-4 show a dramatic difference between the real and forecast value, which indicates that the set of attribute combinations $\{(s2, d1), (s2, d2)\}$ is the abnormal set. A further look at the abnormal set will enable us to identify the root cause as the attribute combination *(s2)*, i.e., the cache server *s2* is experiencing a dramatic traffic drop. We denote the *impacted users* as the users using *s2*, while other users that do not use *s2* are denoted as the *un-impacted users*.

In this case, the operators can switch the traffic of *s2* to other backup CDN edge cache servers immediately to minimize the service downtime. A further investigation may find that the cache server *s2* is suffering from a hardware failure reducing its I/O capacity. Thus, impact identification not only helps identify the impacted users but also provides critical clue for troubleshooting service failures.

C. Graph of Search Space

We organize the total search space of attribute combinations as a directed acyclic graph (DAG) $\mathcal{T}$, where each node corresponds to one attribute combination. Here we take the example described in subsection II-B and illustrate the corresponding DAG as shown in Fig. 1.

TABLE I: Examples of Service Failure in a Real-world Large CDN System

Failure Category	No.	Main Symptoms	Faulty Attributes and Elements	Final Root Cause	Service Downtime
Edge Node	1	Server errors increased	Server: $s1$	Too many plug-ins were used in the cache server configuration, which resulted in a long time to load all plug-ins. It further triggered the server software to kill the time-out user connections by mistake.	2 hours 43 minutes
	2	Some movies of a video website failed to play	Domain: $d1$, Location: $l1$	The maximum size of m3u8 file for domain $d1$ was set too small at edge nodes in location $l1$ by mistake. This led to the rejection for the requests of large videos from $d1$.	10 minutes
User Client	3	Client errors increased	Domain: $d2$, Client Type: $Android$	There was typo error in the newest Android app, which led to an abnormal request to CDN server and made the user request rejected.	1 hour
Network	4	Some HD videos of two video websites were slow to download and cannot be played	Domain: $d3$ $d4$, Location: $l2$	The dedicated back-haul link for domain $d3$ and $d4$ was congested with a high packet loss between the edge nodes in location $l2$ and center nodes.	3 hours 55 minutes
Scheduling Node	5	DNS resolution failed for some domains.	Domain: $d5$ $d6$	Seven servers failed in the scheduling node.	50 minutes
Center Node	6	Video played discontinuous segments	Domain: $d7$ $d8$ $d9$	Data calculating logic in some center nodes was defective when they synchronized data from the original sites, which caused the data it pushed to edges nodes was much smaller than the data it fetched from the original nodes.	41 minutes
Original Sites	7	A live show failed to play	Domain: $d10$	Original server was down because the number of user request for a live show increased suddenly.	1 hour

The root node at layer 0 is the complete set of all the attributes. The nodes at layer 1 are the sets that contain only one attribute element. The nodes at layer 2 are the sets that contain only two attribute elements. Similar rules can be applied to latter layers. In this way, any valid attribute combinations with k specific elements is included in $\mathcal{T}$ and must be located at the k_{th} layer. The edge connection rules are that the node x at layer i points to the node y at layer $i+1$ if the elements in x are contained in y.

Here we give some basic notations about the graph $\mathcal{T}$. We define node a as the *parent* of another node b if a points to b in the graph, e.g., node $(s1)$ is the parent of node $(s1, d1)$. Similarly, b is defined as the *child* of a. Any node without child nodes is a *leaf* node. A node c is said to be *descended* from node a if there exists a path from node a to node c following the edge directions. All the nodes that are descended from node a are called the *subgraph* nodes of a. A *pod* is defined as the set of nodes that have the same type of attributes, e.g., the nodes $(s1)$ and $(s2)$ form one pod with only the server attribute, while the nodes $(s1, d1)$,$(s1, d2)$,$(s2, d1)$,$(s2, d2)$ form another pod with the server and domain attributes.

D. Problem Statement

Following the previous works [13], [15], the definition of impact identification problem is: when an unknown service failure event e happens, find out the most *succinct* set of attribute combinations, termed the *root-cause set* $\mathbf{M}_{opt}$, that should cover all the impacted users affected by the event e and also exclude all the un-impacted users. The attribute combinations in $\mathbf{M}_{opt}$ are called the *root-cause nodes*. Here a set of attribute combinations is said to be more *succinct* than another if it has fewer attribute combinations. For example, the root cause set {(s2)} is more *succinct* than the set {(s2, d1), (s2, d2)}, although both of the two sets can cover all the impacted users and also exclude all the un-impacted users in Table II. Formally, the problem can be further illustrated as: when a service failure event happens, given the graph $\mathcal{T}$ of search space and the binary abnormal labels L for each leaf node in $\mathcal{T}$, find the root-cause node set $\mathbf{M}_{opt}$ in $\mathcal{T}$.

E. Challenges

Although the example shown in Table II is easy to find the root cause, the real-world CDN service failures are much more difficult to analyze due to the following three challenges.

TABLE II: Example of impact identification problem

Index	Server	Domain	Measured	Forecast	Abnormal
1	s1	d1	108	122	0
2	s1	d2	71028	71437	0
3	**s2**	**d1**	**383**	**861**	**1**
4	**s2**	**d2**	**421**	**930**	**1**

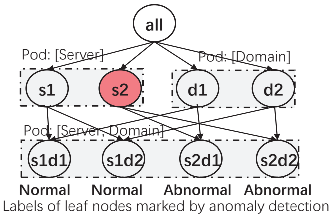


Fig. 1: Graph of search space in Table 1 example

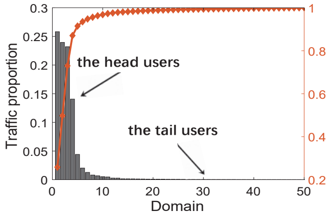


Fig. 2: Traffic distribution over domains

Challenge on anomaly detection errors. If the anomaly detection method f is 100% accurate, it turns to be easy to obtain $\mathbf{M}_{opt}$ in $\mathcal{T}$ by just merging all the leaf nodes that are labeled as *abnormal* in L until the smallest number of parent nodes are found. However, due to the unpredictable traffic variations in real systems, existing anomaly detection algorithms will generate unavoidable and relatively-large errors on marking the abnormal labels L for each leaf node of $\mathcal{T}$, i.e., it is possible to wrongly mark a "normal" leaf node as abnormal, or mark an "abnormal" leaf node as normal. With the inaccurate labeling, simply merging all the leaf nodes that labeled as *abnormal* can not obtain the root-cause set M_{opt}.

Challenge on huge search space. As Fig.1 shows, there are totally 9 valid attribute combinations, which correspond to 9 nodes in graph $\mathcal{T}$. Since the root cause set is possible for holding all valid attribute combinations, it has $2^9 = 512$ potential root cause sets. Generally, suppose the number of attributes is n and each attribute has m elements, it has $(m+1)^n$ valid attribute combinations totally. The number of potential root cause sets can be as large as $2^{(m+1)^n}$. Even for a relatively small value of m and n, the search space is huge and the root cause search is time-consuming.

Challenge on long-tail distribution. The third challenge is due to the long-tail distribution of user traffic. In this paper, when the traffic volume generated by some users takes a small percentage of the total traffic, we say "*the users are in the tail*" or call them "*the tail users*". For example, in Table 1, the users in line 1, 3 and 4 are the tail users because their traffic volumes are two orders of magnitude smaller than the total traffic volume. Fig. 2 shows the long-tail distribution of real user traffic in the CDN system. The users that visit the domains 10-50 take less than 1% of the total traffic, and thus are referred as the tail users. Actually, the traffic difference between the domain 1 and domain 50 is up to six orders of magnitude. All the 50 domains are paid VIP customers and their service performance should be guaranteed.

Existing impact identification solutions either use the absolute metrics, e.g., the absolute difference between the real and forecast values [13], [14], or use the relative metrics, e.g., the relative difference between the real and forecast values [13]–[15]. The long tail effect makes both types of metrics inefficient and inaccurate to work in practice. For the absolute metrics, the search principle that the root cause set should have a large absolute difference between the real and forecast values does not hold anymore. This is because the root cause set may contain users in the tail, who have a small traffic volume, and thus generates a small absolute difference. Similarly, for the relative metrics, the search principle that the root cause should have a large relative difference between the real and forecast values does not hold either. We find that in real CDNs, the tail users with small traffic volume generally have a large relative difference due to their large traffic variation, which however is considered to be normal rather than being included into the root cause set.

III. SYSTEM DESIGN

In this section, we will introduce the detailed design of iSwift to address the above challenges.

A. Overview of the Framework

As shown in Fig. 3, the service fault alerts or user issue reports are the triggers of impact identification. The CDN log files containing user access records are the input of iSwift and each of the record includes the fields of client IP address, server IP address, payload size, cache hit indicator, response time of first byte and so on. Once triggered by service fault alerts or user issue reports, iSwift will find the root cause nodes by the three phases, *anomaly detection and graph building*, *adaptive beam search* and *candidate filtering*. For the phase of anomaly detection and graph building, we first take CDN log files as input and retrieve the time serial data of the finest-grained KPIs, namely, the KPIs of leaf nodes, e.g., the traffic volume for the combination of the attributes represented by a leaf node. Then iSwift continuously applies a light-weight but effective anomaly detection on the KPIs of leaf nodes. Next, iSwift builds the graph $\mathcal{T}$ of search space based on CDN log files and then generates the set of abnormal labels L for each leaf node in $\mathcal{T}$ (see Fig. 1 as an example). $\mathcal{T}$ and L are taken as input of adaptive beam search.

During the second phase, a basic method to search the root-cause set $\mathbf{M}_{opt}$ in the graph $\mathcal{T}$ is to enumerate each node in $\mathcal{T}$ and select proper nodes to form $\mathbf{M}_{opt}$. Since the search space is huge in practice, iSwift develops an adaptive beam search method following the breadth-first search. Although a small beam width can greatly accelerate the search, the accuracy of the search degrades. Therefore, with a reasonable beam width to ensure the search accuracy, it calls for additional methods to further speed up the search. Moreover, the labeling errors of anomaly detection and the long-tail distribution of user traffic make accurate selection of $\mathbf{M}_{opt}$ much more difficult. To address these challenges, iSwift exploits intrinsic properties inside the graph $\mathcal{T}$ to effectively *prune* a large number of non-root-cause nodes and *select* a small number of candidate nodes during the search, which enables a fast and accurate search despite the inaccurate abnormal labels and long-tail distribution. For the third phase, iSwift *filters* a small set of candidate nodes to obtain the final root-cause set $\mathbf{M}_{opt}$.

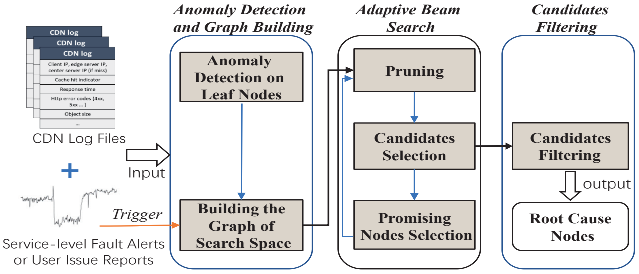

Anomaly Detection and Graph Building
Adaptive Beam Search
Candidates Filtering
CDN log
CDN log
CDN log
Client IP, edge server IP, center server IP (if miss)
Cache hit indicator
Response time
Http error codes (4xx, 5xx ...)
Object size
...
CDN Log Files
+
Input
Trigger
Service-level Fault Alerts or User Issue Reports
Anomaly Detection on Leaf Nodes
Building the Graph of Search Space
Pruning
Candidates Selection
Promising Nodes Selection
Candidates Filtering
output
Root Cause Nodes

Fig. 3: The framework of impact identification

B. Anomaly Detection and Graph Building

Anomaly Detection on KPI Data of Leaf Nodes. Considering the large number of leaf nodes, we apply a light-weight anomaly detection method based on recursive density estimation [20] to the KPI data of leaf nodes. For an observation at each time step of KPI data, the method will generate a forecast value for the KPI metric and compare it with the observed value. It marks an observation as abnormal if the difference between its observed and forecast KPI values is large, e.g., higher than a specific threshold. We note that other complex anomaly detection algorithms [21]–[23] can be further applied to gain better performance without changing our algorithm.

Graph Building. We build the graph as described in subsection II-C. By analyzing the two-year history system failure reports of a commercial CDN, rather than using the two attributes shown in Fig. 1, we apply the five attributes D, L, C, S, N to denote the visited web *domain*, the user *location*, the user *client OS* type, the requested CDN cache *server*, and the access *network* type, respectively. Almost all the root cause nodes for the two-year system failure reports can be represented by the five attributes. And especially, more attributes can be directly supported by our solution without changing the algorithm.

C. Adaptive Beam Search

In this subsection, we will first introduce the design of pruning and candidate selection, followed by promising nodes selection. Finally, we summarize the whole procedure of adaptive beam search.

*1) **Pruning and Candidate Selection**:* The recent schemes Apriori [19] and iDice [14] use the pruning and selection principle based on two metrics, *support* and *confidence*, which are derived from the well-known association rule mining in the area of data mining [24]–[26]. However, our experiments show that both metrics do not work well in real CDN systems, and we identify two key issues.

The first issue is the long-tail distribution of CDN system, which makes the support metric inaccurate and inefficient for pruning. The support metric of a node i is defined as $Sup_i = \frac{Leaf\_Ab_i}{Leaf\_Total}$, where $Leaf\_Ab_i$ denotes the number of abnormal leaf nodes descended from node i and $Leaf\_Total$ denotes the total number of leaf nodes in $\mathcal{T}$. The pruning rule is that a node with the support value smaller than a given threshold δ_s should be pruned. However, the long tail distribution of CDN system may make the root-cause node's support value too small and wrongly pruned, because the denominator $Leaf\_Total$ is very large while the numerator $Leaf\_Ab_i$ may be very small.

Thus potentially there exists no proper threshold of δ_s to distinguish the root-cause nodes from other non-root-cause nodes. Using a high threshold δ_s (e.g., $\delta_s = 0.1\%$ used in iDice [14]) will wrongly prune a large number of root-cause nodes, while a smaller δ_s will make the pruning much less efficient since few nodes will be pruned.

The second issue is the inaccuracy of the anomaly detection methods. In experiments we find that the performance of confidence metric is quite sensitive to the accuracy of the anomaly detection method. For example, for a confidence threshold $0 < \delta_c < 100\%$ (e.g., Apriori [19] sets $\delta_c = 80\%$), if the false negative rate of anomaly detection is higher than $1 - \delta_c$, it will wrongly prune all the root-cause nodes.

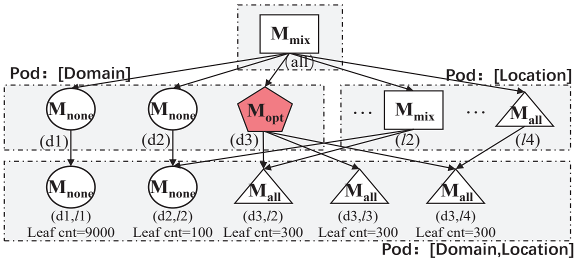
Pruning Design. Motivated by the above observations, we propose a new pruning metric named *latent force*, which uses a small but adaptive denominator associated with the root-cause nodes and adapts to the long-tail distribution of CDN system and different accuracies of anomaly detection. The latent force for node i is defined as

$$\mathcal{L}_i = \frac{Leaf\_Ab_i}{Leaf\_Total\_Ab} \tag{1}$$

where $Leaf\_Total\_Ab$ denotes the total number of abnormal leaf nodes of $\mathcal{T}$.

Suppose $\mathbf{M}_{opt}$ contains a unique root-cause node k. If an ideal anomaly detection method with 100% accuracy is applied, its latent force will be 100%. In case of imperfect anomaly detection, the latent force of node k will still be a large value. For some non-root-cause nodes with a large number of normal leaf nodes descended from them, their latent force will be much smaller than that of root-cause node. Thus the basic pruning principle is that a node with a latent force value smaller than a given threshold δ_l and its subgraph nodes should be pruned, with the threshold δ_l set based on the accuracy of anomaly detection.

Candidate Selection Design. To select the root-cause nodes effectively for long-tail distribution, we develop the key *confidence lossless property* of root-cause nodes. Unlike the metrics in previous works [13]–[15], [17] which are sensitive to the distribution of KPI data, the confidence lossless property is *independent* of the distribution of KPI data and thus the long-tail distribution will not affect its effectiveness to select the root-cause nodes.



Parameters: *Total_Leaf*=10000, R_{fn}=0.1, R_{fp}=0.1

Pod/ Score	Node	Latent force	Confidence	Confidence loss	Action
[All]/-	(all)	100%	17.2%	24.2%	Excluded by confidence
[Domain]/ 47.1%	(d1)	0.6%	10%	0%	Excluded by confidence
	(d2)	52.3%	10%	0%	Excluded by confidence
	(d3)	**47.1%**	**90%**	**0%**	**Selected**
[Location] / 15.7%	(*l*2)	16.3%	70%	20%	Excluded by confidence loss
	(*l*4)	15.7%	90%	0%	Filtered by pod score
	…	…	…	…	…
[Domain, Location] /-	(d1,*l*1)	52.3%	10%	0%	Excluded by confidence
	(d2,*l*2)	0.6%	10%	0%	Excluded by confidence
	(d3,*l*2)	15.7%	90%	0%	Excluded by confidence loss
	(d3,*l*3)	15.7%	90%	0%	Excluded by confidence loss
	(d3,*l*4)	15.7%	90%	0%	Excluded by confidence loss

Fig. 4: Example of impact identification

As shown in Fig. 4, we classify all the nodes in the search space after pruning into three types: (1) the nodes $\mathbf{M}_{all}$ whose leaf nodes are all descended from $\mathbf{M}_{opt}$; (2) the nodes $\mathbf{M}_{none}$, none of their leaf nodes are descended from $\mathbf{M}_{opt}$; (3) the nodes $\mathbf{M}_{mix}$, whose leaf nodes are descended from both $\mathbf{M}_{all}$ and $\mathbf{M}_{none}$. In order to identify the root-cause nodes, we need to identify whether a node belongs to $\mathbf{M}_{all}$ or not during the search. If it belongs to $\mathbf{M}_{all}$, we can safely add it as the candidate root-cause node while pruning its subgraph nodes.

Next, we will introduce how to identify a node in $\mathbf{M}_{all}$ by distinguishing it from the other two types of nodes $\mathbf{M}_{none}$ and $\mathbf{M}_{mix}$. The confidence metric of node i is defined as

$$\mathcal{C}_i = \frac{Leaf\_Ab_i}{Leaf_i} \tag{2}$$

where $Leaf_i$ denotes the number of leaf nodes descended from node i [19], [24], [25]. Similarly, we denote $Leaf_{all}$ and $Leaf_{none}$ as the number of leaf nodes descended from $\mathbf{M}_{all}$ and $\mathbf{M}_{none}$ respectively.

The first step is to distinguish the nodes $\mathbf{M}_{all}$ against $\mathbf{M}_{none}$. Since a normal leaf node may be wrongly labeled as an abnormal one by inaccurate anomaly detection in practice, we denote R_{fp} and R_{fn} as the percentage of false positives and the percentage of false negatives respectively in the total leaf nodes of $\mathcal{T}$. Based on the Equ (2), if R_{fp} and R_{fn} satisfy the uniform-random distribution, the confidence of $\mathbf{M}_{all}$ and their subgraph nodes can be computed as $\mathcal{C}_{all} = \frac{Leaf_{all} \times (1 - R_{fn})}{Leaf_{all}} = 1 - R_{fn}$. The confidence of $\mathbf{M}_{none}$ and their subgraph nodes can be computed as $\mathcal{C}_{none} = \frac{Leaf_{none} \times R_{fp}}{Leaf_{none}} = R_{fp}$. Thus the confidence value satisfies the *confidence lossless* property, i.e., the nodes $\mathbf{M}_{all}$ and $\mathbf{M}_{none}$ have the same confidence value as that of their subgraph nodes. Thus we can easily distinguish these two types of nodes if a proper threshold value δ_c is set as $1 - R_{fn} > \delta_c > R_{fp}$. The pruning principle of confidence value is that a node with a confidence value larger than δ_c should be pruned. The existence of δ_c requires the accuracy of anomaly detection to satisfy the constraint $1 - R_{fn} > R_{fp}$, i.e., $R_{fn} + R_{fp} < 1$. In experiment we show that this constraint can be achieved easily in iSwift system and our approach is robust to the accuracy of the anomaly detection method.

The second step is to distinguish the nodes $\mathbf{M}_{all}$ against $\mathbf{M}_{mix}$. The nodes $\mathbf{M}_{mix}$ have different confidence values and can be calculated as

$$\mathcal{C}_{mix} = \gamma \times (1 - R_{fn}) + (1 - \gamma) \times R_{fp}, \tag{3}$$

where γ is the percentage of leaf nodes descended from $\mathbf{M}_{all}$. Since γ can be any value between 0 and 1, it is difficult and sometimes even impossible to distinguish them if using only the confidence threshold that adopted in previous work. To address the issue, we propose a new metric named *confidence loss* Δ_i of a node i as the difference between the confidence value of node i and the average confidence value of its children nodes. Based on the confidence lossless property, the confidence loss of $\mathbf{M}_{all}$ is zero while that of $\mathbf{M}_{mix}$ is larger than zero. The selection principle of confidence loss is that a node with a confidence loss value smaller than δ_Δ should be selected as a candidate. Otherwise, it should be excluded.

Here we present an example in Fig. 4 as an illustration. The example contains both the long-tail distribution and inaccurate anomaly detection, e.g., the root-cause node $(d3)$ is a set of tail users while the false negative and false positive of anomaly detection are both 10%. The node $(d3)$ is the unique root-cause node in $\mathbf{M}_{opt}$. The confidence of node $(d3)$ and all its child nodes are 90%, which matches the confidence lossless property, i.e., the confidence loss is zero for $\mathbf{M}_{all}$. In contrast, the $\mathbf{M}_{mix}$ node $(l2)$ dose not satisfy the confidence lossless property. In this example, $(l2)$ has 300 leaf nodes descended from $\mathbf{M}_{all}$ node $(d3, l2)$ and 100 leaf nodes descended from $\mathbf{M}_{none}$ node $(d2, l2)$ and thus the γ is 0.75 for $(l2)$. The confidence of $(l2)$ is 70%. The confidence of its children node $(d3, l2)$ and $(d2, l2)$ are 90% and 10% respectively. The average confidence of its children nodes $(d3, l2)$ and $(d2, l2)$ is 50%. The confidence loss for $\mathbf{M}_{mix}$ node $(l2)$ is 20%. Hence the nodes $\mathbf{M}_{all}$ and $\mathbf{M}_{mix}$ can be distinguished by the confidence loss metric.

Since R_{fp} and R_{fn} may not be a perfect uniform-random distribution in practice, the confidence lossless property is not exactly held but is nearly satisfied. Thus we use two robust thresholds δ_c and δ_Δ to address the non-uniform distribution in real CDN system (Section IV-E).

The above two steps help exclude the non-root-cause nodes $\mathbf{M}_{none}$ and $\mathbf{M}_{mix}$, and then the remaining nodes $\mathbf{M}_{all}$ are selected into the set of candidate root-cause nodes.

*2) **Promising Nodes Selection**:* The basic framework of beam search is to explore the graph by expanding the most promising node in a limited-sized set [27]. In our problem, the key is to define which nodes are *promising* at the each search step and how to generate the next-step search set. In iSwift search, we define the *promising probability* metric of node s simply as the sum of its confidence and latent force.

Algorithm 1 Adaptive Beam Search

```
Input: Graph $\mathcal{T}$, Abnormal Labels $L$
Output: Candidate Set $\mathbf{C}$
 1: Calculate latent_force and confidence of each node in $\mathcal{T}$
 2: $\mathcal{S} \leftarrow Child\_Set(all)$
 3: while $\mathcal{S} \neq \emptyset$ do
 4:   for $s \in \mathcal{S}$ do
 5:     //Non-root-cause nodes pruning
 6:     if $s$.latent_force < $\delta_l$ then
 7:       Delete node $s$ and its subgraph in $\mathcal{T}$
 8:     else if $s$.confidence > $\delta_c$ and $|\Delta_s| < \delta_\Delta$ then
 9:       //Root-cause nodes identify & subgraph pruning
10:       Add $s$ to $\mathbf{C}$
11:       Delete the subgraph of node $s$ in $\mathcal{T}$
12:     end if
13:   end for
14:   $\mathcal{S} \leftarrow Top\_K(\mathcal{S})$
15:   $\mathcal{S} \leftarrow Expand\_Set(\mathcal{S})$
16: end while
```

*3) **Put All Together**:* As Algorithm 1 shows, iSwift starts the search from the top layer of the graph $\mathcal{T}$ (line 1-2). During the search, it maintains two special sets, i.e., the *search set* $\mathcal{S}$ and the *candidate set* $\mathbf{C}$. The search set $\mathcal{S}$ is designed to keep track of the most promising nodes in the current search level and then generate the next-level nodes for searching. The candidate set is utilized to simply store the identified candidate root-cause nodes during the entire search process and will be finally taken as the algorithm output. For the current $\mathcal{S}$, iSwift first identifies all the non-root-cause nodes with the metric of latent force (line 3-6), where the identified non-root-cause nodes and their subgraph nodes are then pruned from $\mathcal{T}$ (line 7). Next, iSwift identifies the potential root-cause nodes with the metrics of confidence and confidence loss (line 8). It adds such nodes into the candidate set $\mathbf{C}$ (line 10) and deletes their subgraph nodes in $\mathcal{T}$ (line 11). After the above pruning process, iSwift obtains K nodes in $\mathcal{S}$ that have the top-K promising probability, where K is the beam width (line 14).

By now the most promising nodes at current step are selected, and we need to generate the next-level search set (line 15). iSwift uses a special process named *pairwise combination* to generate the new search set, where any two nodes in the previous search set $\mathcal{S}$ are combined to generate one child node. According to the structure of $\mathcal{T}$, the principle of pairwise combination is that the two nodes will merge the same attribute elements. For example, for two nodes $\mathcal{M}_1 = (d_1, c_1)$ and $\mathcal{M}_2 = (l_1, c_1)$, they are combined to generate their child node $\mathcal{M}_3 = (d_1, l_1, c_1)$. Specially, if the given two nodes have different elements on the same attribute, they have no child node and can not be merged. For example, for two nodes $\mathcal{M}_1 = (d_1, c_1)$ and $\mathcal{M}_2 = (d_2, l_1, c_1)$, they can not be merged based on the principle. We denote the pairwise combination performed on the set $\mathcal{S}$ as a function $Expand\_Set(\mathcal{S})$, which is applied to update $\mathcal{S}$ as the next-level search set.

D. Candidates Filtering

After the beam search process, the output candidate set $\mathbf{C}$ may still contain some redundancy. For example, node $(l4)$

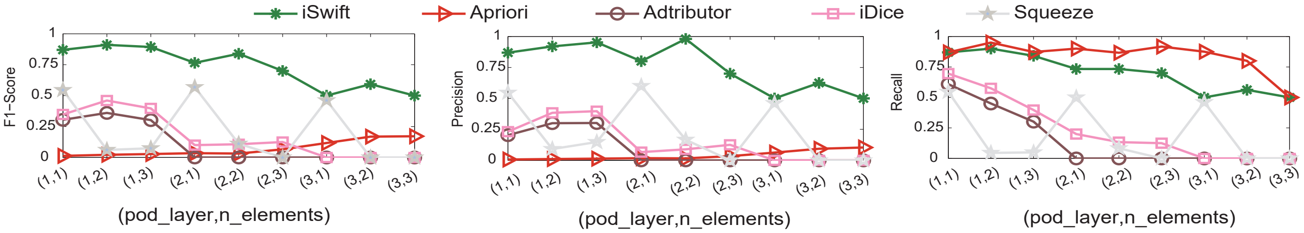


Fig. 5: Performance comparison over different root-cause node numbers and layers

is a redundant candidate in Fig. 4 since all its leaf nodes are covered by the root-cause node's child node $(d3, l4)$. As a result, $(l4)$ has the confidence loss Δ_c as 0% and it is *wrongly* selected in the candidate set. To address this problem, we provide a filter step after the beam search.

The filter step is motivated by the fact that all the root cause nodes in $\mathbf{M}_{opt}$ caused by one system fault are in only one *pod* (see the definition of pod in Section II-C) and it is unlikely to have multiple system faults at the same time in real CDN system. Thus we can classify all the candidate nodes in $\mathbf{C}$ into different pods. We define a *pod score* metric of pod k as $Pod\_Score_k = \sum_{j=0}^{n-1}(\mathcal{L}_j)/n$, where n is the number of candidates in pod k and $\mathcal{L}_j$ is the latent force of j_{th} candidate in pod k. The filter principle is to select the candidate nodes in the pod with maximum pod score (termed the *root-cause pod*) as the final result. As Fig. 4 shows, after the search process, the candidate nodes are the root cause node $(d3)$ in pod $[Domain]$ and the non-root-cause node $(l4)$ in pod $[Location]$. The pod score of $[Domain]$ is 47.1%, while the pod score of $[Location]$ is only 15.7%. So the final root-cause pod is $[Domain]$ with the root-cause node $(d3)$ in it.

IV. Evaluation

A. Evaluation Setup

For comparison, we implement the state-of-the-art impact identification algorithms including Squeeze [15], Apriori [19], [24], Adtributor [13] and iDice [14]. The baseline algorithms use the default parameter setting in the corresponding literature. For iSwift, the confidence pruning threshold is set to 50%, the beam width K is set to 350, the confidence loss threshold and the latent force are 0.08 and 0.001 respectively by default. The running time is evaluated on a laptop with Intel i7-9750H CPU @2.6GHz and 32GB memory.

In this section, we simulate a large range of input cases by injecting synthetic anomalies into the real data traces. The real data traces are collected from one provincial-level edge site of a top CDN in China over one and a half years. Specifically, the edge site is equipped with more than 200 CDN servers and provides services for more than 1 million Internet users. The attributes and their definitions are the same as introduced in Section III-B, i.e., there are five attributes D, L, C, S, N where the number of elements in the attributes are $|D| = 50$, $|L| = 47$, $|C| = 5$, $|S| = 233$, $|N| = 6$.

To make a fair comparison, we use the same setup of synthetic trace generation in previous work [15]. The generation works as follows. First, the injected root-cause nodes are randomly generated and share the same faulty attributes (i.e., located in one pod). We limit the number of root-cause nodes and the layer of root-cause pod to be within 3 based on the setup of [15]. We also confirm that this parameter setting matches the statistics of historical CDN issue reports. Second, we change the KPI value of leaf nodes descended from the injected root cause nodes with random amounts in the same range as that of historical accident statistics from real data traces. Finally, we add Gaussian noises to all leaf nodes with different standard deviations to emulate different traffic forecast errors. Thus we can compare the performance of iSwift with all the other algorithms in a comprehensive way.

To further assess the impact of the anomaly detection accuracy, we directly simulate the anomaly detection results to apply on the leaf nodes of $\mathcal{T}$. This can enable the evaluation of the algorithm performance under different accuracies of anomaly detection without the need to specify any detailed anomaly detection method.

B. Performance over different root causes

In this section, we change the location of root-cause nodes to different pod layers and also vary the number of root-cause nodes to evaluate the algorithm performance in Fig. 5.

We can see that in all the cases, iSwift achieves the best F1-score performance against other algorithms. Moreover, iSwift, Adtributor and iDice share the same decreasing trend of F1-score when meeting deeper location and larger number of root-cause nodes. Specifically, iSwift decreases slightly while Adtributor does not work when the number of root-cause nodes is larger than 1 based on its initial design. iDice can work for multiple root-cause nodes, but its performance decreases quickly when the number of root-cause nodes increases. It achieves relatively good F1-score about 0.4 when the root-cause nodes are located on the first layer and poor F1-score less than 0.15 when located on deeper layers. This is due to the average support value of root-cause nodes becomes smaller when located on higher layers. We can find that the performance of Squeeze is sensitive only to the number of root-cause nodes but not their locations. This demonstrates the inefficiency of the clustering method in Squeeze to deal with multiple root-cause nodes.

C. Performance under long tail effects

We denote the metric *LTM* (Long-Tail-Metric) to evaluate the impact of long tail distribution for a root cause node $\mathbf{M}_{opt}$. We define LTM as the traffic proportion F_r/F_{all}, where F_r is the traffic volume of $\mathbf{M}_{opt}$ and F_{all} is the traffic volume for all leaf nodes. In Fig. 2, the LTM is lower than 10% for above 90% nodes with the attribute of *domain* in the first layer of $\mathcal{T}$.

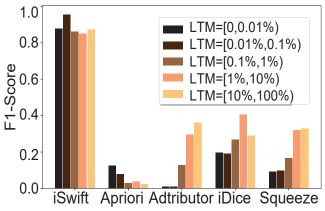


Fig. 6: F1-score over different long tail effects

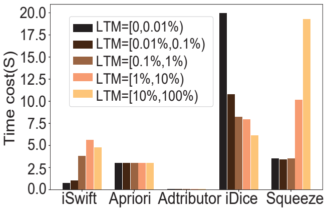


Fig. 7: Search time over different long tail effects

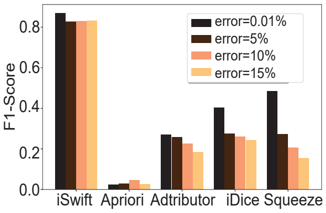


Fig. 8: F1-score over different forecast errors

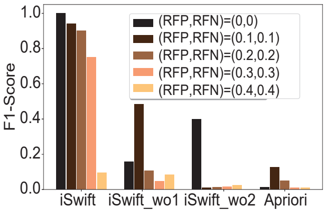
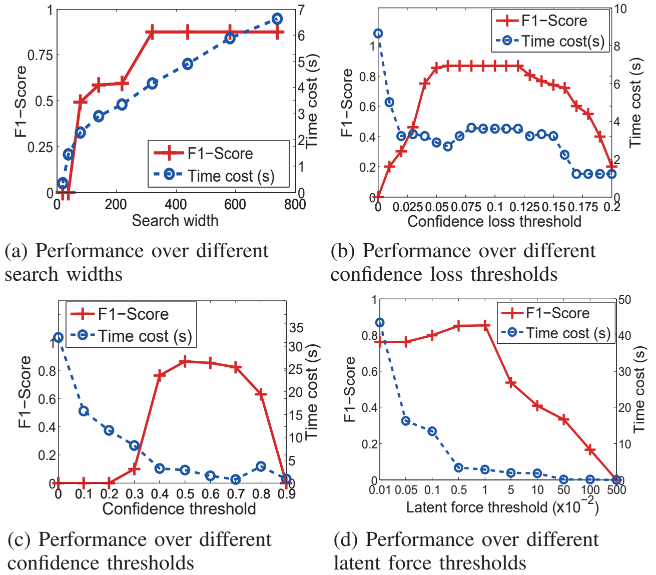


Fig. 9: F1-score at different anomaly detection accuracies



(a) Performance over different search widths

(b) Performance over different confidence loss thresholds

(c) Performance over different confidence thresholds

(d) Performance over different latent force thresholds

Fig. 10: Parameter study of iSwift

The long tail effect becomes more serious for deeper layers, indicated by a smaller LTM value.

As Fig. 6 and Fig. 7 show, iSwift performs best for all the magnitudes of LTM with F1-score larger than 0.85 and it only adds a small overhead on the total search time, which is always kept below 6 seconds in a realistic setting. Although Apriori achieves a short search time due to its simplest search operations, it performs worst on the whole. This is because its fixed threshold of confidence will wrongly put many non-root-cause nodes into the final candidates, which leads to a high rate of false positive and thus low F1-score. Adtributor achieves the lowest search time because it is designed to only work for the case that the root-cause nodes are located on the first layer, which leads to its bad performance when the root-cause nodes are located in deeper layers. The three schemes Adtributor, Squeeze and iDice have similar F1-score performance and are stably superior to Apriori. They share the same trend that the F1-score increases when the LTM value is higher, i.e., the long tail effect is less significant.

D. Imperfect forecast and anomaly detection study

As shown in Fig. 8, we can see that iSwift achieves the highest F1-score and performs robustly over all the forecast errors. As a comparison, the F1-score of Apriori is the lowest, due to its use of a fixed candidate threshold. With the increase of forecast error, the performance of Adtributor, iDice and Squeeze degrades quickly. As all the leaf nodes are added with larger forecast errors, it is difficult to distinguish normal leaf nodes from abnormal leaf nodes.

To dive into the impact of imperfect anomaly detection, we evaluate the performance of different components of iSwift separately in Fig. 9. *iSwift_wo1* denotes the version of iSwift without applying the pruning methods introduced in Section III-C1. *iSwift_wo2* denotes the version of iSwift without applying the candidate filtering method introduced in Section III-D. We can see that iSwift achieves high F1-score above 0.75 when the rate of false positive (RFP) and the rate of false negative (RFN) are less than 0.3, while Apriori gets the worst performance over all the cases. Interestingly, the performance of iSwift drops significantly when removing either the filtering component or the pruning component. This validates the high performance gain and robustness of the combined design of the two components in iSwift.

E. Parameter study of iSwift

In terms of the search width, a small ω may miss the root cause nodes and achieve poor F1-score performance, while a large ω may enlarge the time cost but gain high F1-score performance. As Fig. 10a shows, when ω is less than 350, the F1-score of iSwift increases rapidly. When search width ω is larger than 350, iSwift achieves the max and flat F1-score performance about 0.85, which keeps stable for larger search widths. At the same time, the time cost is close to a linear growth with search width. Specifically, the time cost is less than 7 seconds when the search width ω is within 800. The large flat area of high F1-score demonstrates the robustness of iSwift over the search width parameter.

The confidence loss threshold $\delta_\triangle$ is the essential metric to identify root cause node in iSwift. A larger $\delta_\triangle$ will get a higher rate of false positive with more nodes added in the root-cause candidates. As analyzed in Section III-C1, $\delta_\triangle$ is a number close to zero. However, if $\delta_\triangle$ is too small, it may miss the root cause nodes since the distribution of false negative leaves of root cause nodes can be non-uniform in practice. As Fig. 10b shows, iSwift achieves both the highest and stable performance when $\delta_\triangle$ is within the large middle range of 0.05 and 0.12. This verifies its robustness over confidence loss threshold.

The confidence metric is a basic metric that can distinguish the nodes $\mathcal{M}_{all}$ from $\mathcal{M}_{none}$. Fig. 10c demonstrates that the F1-score increases to a stable high-value region when δ_c is larger than 0.3 and smaller than 0.8. The latent force metric is the basic pruning metric to accelerate the search speed. As Fig. 10d shows, the search time decreases exponentially to be within five seconds when the latent force threshold δ_l increases. The F1-score keeps a stable high value when δ_l is smaller than 1% and decreases linearly when it becomes larger. The large stable region of high F1-score of δ_c and δ_l setting demonstrates the robustness of the confidence threshold and latent force threshold respectively.

V. Case Study in Industrial Practice

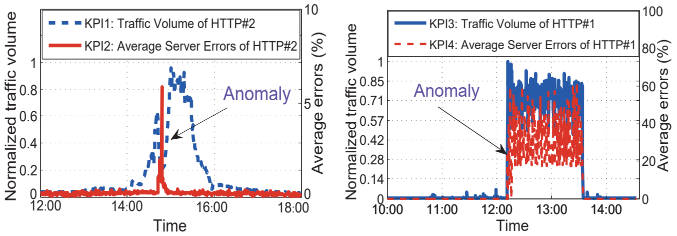
To evaluate the online running performance of iSwift in identifying the impact of anomalies in a practical system, we deploy iSwift at a provincial-level edge site of a top CDN in China. It has been running to analyze the online streaming log data around one year. The system is run over a Hadoop cluster with tens of servers and 250 CPU kernels, which can process log files of about 2 million user requests per minute in real time. In the cluster, the impact identification module is deployed on one server with 24 CPU kernels (Intel Xeon CPU E5-2620 v2@2.10GHz) and 128 GB of RAM. The attribute setting is the same as that of Section IV-A. Here we demonstrate successes of two typical cases.

Since the total number of severe system accidents is small even for the duration of nearly one year, considering the high reliability requirement of running large-scale multimedia content delivery system, the operators currently do not use iSwift directly for commercial use. Instead, it is set up in a two-way testing mode, where the system accidents are still identified and handled manually while the iSwift system is running and validated against manual results at the same time.

The following cases involve the monitoring of KPI metrics on two typical HTTP connections in the CDN system. One is the connection between the users and CDN edge servers (termed HTTP#1), while the other is the connection between the CDN edge servers and center servers (termed HTTP#2). The results of iSwift are sufficient to identify the impacted users and guide a fast troubleshooting. We anonymize some details for confidential reason.

A. Case 1: online video meeting access failure

On March 9th, the network operators found a sudden increase of traffic volume of HTTP#2 (i.e., KPI1 in Fig. 11a). There is a 2-hour online meeting with real-time video streaming during 14:00 and 16:00. Moreover, the operator found that the average number of server errors of HTTP#2 has a sudden increase around 14:24 (i.e., KPI2 in Fig. 11a). It takes about 20 minutes for the operator to identify the impacted user range, namely, the set of users visiting a specific web domain d_x that serves the online meeting. Thus the operator switches partial traffic of the impacted users to other center servers to solve the problem. As confirmed by the operator, our system iSwift shows a real-time identification of the impacted users (within 3 seconds) and the output root-cause set $\mathbf{M}_{opt}$ is $\{(d_x)\}$, which is exactly consistent with that identified by the operator manually. Specifically, the graph $\mathcal{T}$ of the search space involves 210k nodes, i.e., there are totally 2^{210k} potential root-cause sets. In terms of the accuracy of anomaly detection in this case, the rate of false negative is about 18% and the rate of false positive is about 23%. This automated and accurate result shows the deployment of iSwift can benefit real system performance despite the huge search space and inaccurate anomaly detection in practice.



(a) Main abnormal KPIs in case 1 (b) Main abnormal KPIs in case 2

Fig. 11: Real-world system accidents in CDN

B. Case 2: video website access accident for home network at a certain location

The second case happened on April 5th. As Fig. 11b shows, the operator found a sudden increase of traffic volume of HTTP#1 (i.e., KPI3). At the same time, there was a sudden increase of server errors of of HTTP#1 (i.e., KPI4). This anomaly affected a number of users to prevent their access to a top video website d_y and further raised the user reports to the network operator. After a hard manual work about 2 hours, the operators finally confirmed the impacted user range was the users who visited the website d_y via a residential network n_y at a certain location l_y. The case happened because the date April 5 is a Chinese holiday called ”Ching Ming Festival”, where there is a custom of returning home to worship ancestors. Most of the people working in cities went back home in the countryside l_y and watched TV with a home network on that date, and the website d_y happened to have a popular TV show with high ratings during the holiday. The traffic of $\{(d_y, n_y, l_y)\}$ increased by about 30 times, which exceeded the service capacity of the related edge nodes, thus generating the above video website access accident.

Since the service downtime is relatively long in this case, the accident is considered as a serious one. As confirmed by the network operator, our system iSwift outputs the root-cause set $\mathbf{M}_{opt} = \{(d_y, n_y, l_y)\}$ within 4 seconds, i.e., the impacted user range was the users who visited the web domain d_y via a home network n_y at a certain location l_y. The operator confirms that the $\mathbf{M}_{opt}$ is exactly consistent with the manual identification results.

VI. Related Work

With the increasing network scales and high demands of service reliability, the impact identification of system issues has become a popular research topic recently [13]–[15], [17], [28]. Existing works demonstrate the importance and effectiveness of impact identification in different systems. However, they mainly focus on the impact analysis without data restrictions of long tail distribution and anomaly detection accuracy.

Adtributor [13] is deployed to identify the most-likely attribute when the revenue anomalies (i.e., the abrupt reduction of revenue) are detected in advertising systems. But the method is limited to identify only one attribute as the root cause. R-Adtributor [29] further provides a multi-dimensional root cause identification solution by calling Adtributor recursively. iDice [14] is designed to identify the effective combination of multiple attributes for emerging issues in Microsoft service systems. HotSpot [17] identifies the root-cause attribute combinations for online software services. But both iDice and HotSpot are designed only for the additive KPI, while our work shows that both the additive and non-additive KPI metrics are important for impact identification in CDN systems. Apriori [18], [19] detects and localizes the end-to-end performance degradation for cellular services based on the results of anomaly detection. However, Apriori considers only the nodes with the confidence metric larger than a fixed threshold can be a potential root cause. In previous section we show that due to the long-tail distribution of user requests and the inaccurate anomaly detection, setting an appropriate threshold of confidence can be difficult and sometimes infeasible.

As recent work Squeeze [15] shows, all the above schemes suffer in the presence of long tail distribution of user traffic, which however is found prevalent in CDN systems. In Squeeze, Li et al. propose an improved solution based on clustering to address a light long tail issue. However, through extensive experiments we find that it is hard to form an appropriate clustering for the data with much more serious long-tail effect in CDN.

VII. Conclusion

In this paper, we propose a novel system named iSwift to identify the impacted users. Specifically, we propose the key property of confidence lossless to accurately identify the root cause despite the long-tail traffic distribution. Next, we develop effective pruning metrics to reduce the huge search space and achieve the real-time search of root cause candidates. We utilize an adaptive beam search method with candidate filtering to combine all the benefits of pruning and filtering, which achieves both the fast search speed and high accuracy. The evaluation with both the synthetic and real traces demonstrates the high robustness performance of iSwift over a large range of parameter settings and its higher performance on root cause identification over existing state-of-the-art solutions. The real-world system deployment validates the online performance of iSwift to handle real system anomalies in production CDN.

VIII. Acknowledgement

This work is supported by the National Key Research and Development Program of China (No. 2021YFB2910108), and the National Natural Science Foundation of China under Grant 61971382.

References

[1] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Computing 2002*.

[2] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *SIGCOMM 2015*.

[3] M. Calder, R. Gao, M. Schröder, R. Stewart, J. Padhye, R. Mahajan, G. Ananthanarayanan, and E. Katz-Bassett, "Odin: Microsoft's scalable fault-tolerant cdn measurement system," in *NSDI 2018*, pp. 501–517.

[4] M. Z. Shafiq, A. R. Khakpour, and A. X. Liu, "Characterizing caching workload of a large commercial content delivery network," in *INFOCOM 2016*, pp. 1–9.

[5] J. Gao, N. Yaseen, R. MacDavid, F. V. Frujeri, V. Liu, R. Bianchini, R. Aditya, X. Wang, H. Lee, D. Maltz *et al.*, "Scouts: Improving the diagnosis process through domain-customized incident routing," in *SIGCOMM 2020*, pp. 253–269.

[6] C. Fang, H. Liu, M. Miao, J. Ye, L. Wang, W. Zhang, D. Kang, B. Lyv, P. Cheng, and J. Chen, "Vtrace: Automatic diagnostic system for persistent packet loss in cloud-scale overlay network," in *SIGCOMM 2020*, pp. 31–43.

[7] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "Netbouncer: Active device and link failure localization in data center networks," in *NSDI 2019*, pp. 599–614.

[8] A. Khandelwal, R. Agarwal, and I. Stoica, "Confluo: Distributed monitoring and diagnosis stack for high-speed networks," in *NSDI 2019*.

[9] B. Yang, X. Ji, X. Ma, X. Wang, T. Zhang, X. Zhu, N. El-Sayed, H. Lan, Y. Yang, J. Zhai *et al.*, "End-to-end i/o monitoring on a leading supercomputer," in *NSDI 2019*, pp. 379–394.

[10] G. Yu, P. Chen, H. Chen, Z. Guan, Z. Huang, L. Jing, T. Weng, X. Sun, and X. Li, "Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments," in *WWW 2021*.

[11] S. Burnett, L. Chen, D. A. Creager, M. Efimov, I. Grigorik, B. Jones, H. V. Madhyastha, P. Papageorge, B. Rogan, C. Stahl *et al.*, "Network error logging: Client-side measurement of end-to-end web service reliability," in *NSDI 2020*, pp. 985–998.

[12] C. Lou, P. Huang, and S. Smith, "Understanding, detecting and localizing partial failures in large system software," in *NSDI 2020*, pp. 559–574.

[13] R. Bhagwan, R. Kumar, R. Ramjee, G. Varghese, S. Mohapatra, H. Manoharan, and P. Shah, "Adtributor: Revenue debugging in advertising systems," in *NSDI 2014*, pp. 43–55.

[14] Q. Lin, J.-G. Lou, H. Zhang, and D. Zhang, "idice: problem identification for emerging issues," in *Proceedings of the 38th International Conference on Software Engineering (ICSE 2016)*, pp. 214–224.

[15] Z. Li, C. Luo, Y. Zhao, Y. Sun, K. Sui, X. Wang, D. Liu, X. Jin, Q. Wang, and D. Pei, "Generic and robust localization of multi-dimensional root causes," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE 2019)*, pp. 47–57.

[16] S. Zhang, L. Ying, P. Dan, C. Yu, and Z. Zhi, "Rapid and robust impact assessment of software changes in large internet-based services," in *CoNEXT 2015*.

[17] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu *et al.*, "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," *IEEE Access 2018*.

[18] F. Ahmed, J. Erman, Z. Ge, A. X. Liu, J. Wang, and H. Yan, "Detecting and localizing end-to-end performance degradation for cellular data services," in *INFOCOM 2016*, pp. 1–9.

[19] ——, "Detecting and localizing end-to-end performance degradation for cellular data services based on tcp loss ratio and round trip time," *TON 2017*, vol. 25, no. 6, pp. 3709–3722, 2017.

[20] B. S. J. Costa, P. P. Angelov, and L. A. Guedes, "Real-time fault detection using recursive density estimation," *Journal of Control Automation and Electrical Systems*, vol. 25, no. 4, pp. 428–437, 2014.

[21] L. Dai, T. Lin, C. Liu, B. Jiang, and Z. L. Zhang, "Sdfvae: Static and dynamic factorized vae for anomaly detection of multivariate cdn kpis," in *WWW 2021*.

[22] Y. Su, Y. Zhao, M. Sun, S. Zhang, X. Wen, Y. Zhang, X. Liu, X. Liu, J. Tang, W. Wu, and D. Pei, "Detecting outlier machine instances through gaussian mixture variational autoencoder with one dimensional cnn," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[23] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, 2021.

[24] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *European conference on principles of data mining and knowledge discovery*. Springer, 2000.

[25] E. Mitidieri and S. I. Pokhozhaev, "A priori estimates and blow-up of solutions to nonlinear partial differential equations and inequalities," *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 2001.

[26] S. K. Harms and J. S. Deogun, "Sequential association rule mining with time lags," *Journal of Intelligent Information Systems*, 2004.

[27] I. Sabuncuoglu and M. Bayiz, "Job shop scheduling with beam search," *European Journal of Operational Research*, vol. 118, no. 2, pp. 390–412, 1999.

[28] S. Venkataraman and J. Wang, "Towards identifying impacted users in cellular services," in *SIGKDD 2019*, pp. 3029–3039.

[29] M. Persson and L. Rudenius, "Anomaly detection and fault localization an automated process for advertising systems," Master's thesis, 2018.