ϵ-Diagnosis: Unsupervised and Real-time Diagnosis of Small-window Long-tail Latency in Large-scale Microservice Platforms

Huasong Shan[1], Yuan Chen[1], Haifeng Liu[2,3], Yunpeng Zhang[2], Xiao Xiao[2], Xiaofeng He[2], Min Li[1], Wei Ding[1]
[1]JD.com Silicon Valley R&D Center, Mountain View, CA, USA
[2]JD.com, Beijing, China
[3]University of Science and Technology of China, Hefei, China

ABSTRACT

Microservice architectures and container technologies are broadly adopted by giant internet companies to support their web services, which typically have a strict service-level objective (SLO), tail latency, rather than average latency. However, diagnosing SLO violations, e.g., long tail latency problem, is non-trivial for large-scale web applications in shared microservice platforms due to million-level operational data and complex operational environments.

We identify a new type of tail latency problem for web services, *small-window long-tail latency* (SWLT), which is typically aggregated during a small statistical window (e.g., 1-minute or 1-second). We observe SWLT usually occurs in a small number of containers in microservice clusters and sharply shifts among different containers at different time points. To diagnose root-causes of SWLT, we propose an unsupervised and low-cost diagnosis algorithm–ϵ-Diagnosis, using two-sample test algorithm and ϵ-statistics for measuring similarity of time series to identify root-cause metrics from millions of metrics. We implement and deploy a real-time diagnosis system in our real-production microservice platforms. The evaluation using real web application datasets demonstrates that ϵ-Diagnosis can identify all the actual root-causes at runtime and significantly reduce the candidate problem space, outperforming other time-series distance based root-cause analysis algorithms.

CCS CONCEPTS

• **General and reference** → *Performance*; *Measurement*; • **Information systems** → *Online analytical processing*;

KEYWORDS

Root-cause analysis; tail latency; time series similarity

ACM Reference Format:
Huasong Shan[1], Yuan Chen[1], Haifeng Liu[2,3], Yunpeng Zhang[2], Xiao Xiao[2], Xiaofeng He[2], Min Li[1], Wei Ding[1] [1]JD.com Silicon Valley R&D Center, Mountain View, CA, USA [2]JD.com, Beijing, China [3]University of Science and Technology of China, Hefei, China . 2019. ϵ-Diagnosis: Unsupervised and Real-time Diagnosis of Small- window Long-tail Latency in Large-scale Microservice Platforms . In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313653

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313653

1 INTRODUCTION

Tail latency [3, 7–9, 18, 19, 21, 24, 28, 28, 37, 43, 46, 47] becomes the main consideration as Service Level Objective (SLO) on performance [1, 29] for today's most of systems and applications, such as search engine [7, 24], key-value store [9, 20], datacenter network system [21, 46], shared networked storage [47], online data-intensive applications [43], big data system [8] etc. Strict SLO usually requires tail latency, rather than average latency. For example, Google requires the 99th percentile service time (written *TP*99) for its webSearch cluster within tens of milliseconds [24].

Some e-commerce companies try their best to provide the fastest user experience, they expect their web services can serve the requests within several milliseconds. Using a large statistical window to calculate the tail latency can not accurately identify the long-tail latency problem, thus they calculate the tail latency for their online shopping web services during 1-minute period, even during 1-second period. We call this type of tail latency as *small-window tail latency*. We study the small-window tail latency from the real-production web services in container-based microservice platforms (§2), finding that a new type of long-tail latency problem–*small-window long-tail latency* (SWLT), which usually occurs in a small number of containers in the microservice clusters, sharply shifts among different containers at different time points, and the tail latency of each container varies significantly.

Diagnosing root-causes of SWLT quickly is a challenging task. Firstly, as the business requirements increase, the number of web services dramatically increases. Figure 1 records the weekly number of web services from an internet company since July 2018, consequently the monitored performance metrics increase in million level [42]. Once SWLT occurs, identifying root-causes of SWLT seems like finding a needle from haystack. Secondly, current machine learn and deep learning based root-cause analysis approaches, heavily depend on training of massive data with highly computing cost, which is suitable for diagnosing daily [22] or seasonal anomaly [45], not fit for diagnosing the long-tail latency at extremely small timescales which have a heavy-tail, frequent-change and high-variance feature. Diagnosing root-causes of SWLT requires the algorithm with low computation cost and high recall, and real-time delivery of analytical results.

In this paper, we present an unsupervised and low-cost root-cause analysis algorithm to diagnose root-causes of SWLT at runtime for web services in large-scale microservice platforms. In particular, we make the following contributions.

- We identify a new type of tail latency problem, *small-window long-tail latency* (e.g., in an 1-minute or 1-second period), which has a heavy-tail and high-variance characterization

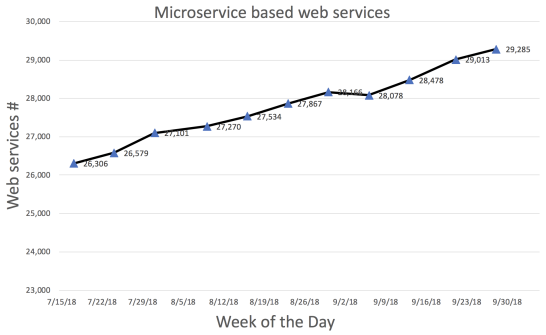


Figure 1: Number of web services in an internet company.

(§2). To the best of our knowledge, no current root-cause analysis algorithm can scale to such granularity.

- We propose an unsupervised and low-cost root-cause analysis algorithm–ϵ-Diagnosis (§3), using two-sample test algorithm and ϵ-statistics for measuring the similarity of time series, to identify root-causes of SWLT from millions of metrics for on-line web services at runtime. The ϵ-statistics is specially well-suited for the root-cause analysis for the heavy-tailed and highly-variant web applications.
- We implement a real-time diagnosis system in our real-production microservice platforms (§3). The evaluation using real-production datasets demonstrates the effectiveness and efficiency of the proposed root-cause analysis algorithm (§4). Our results show that ϵ-Diagnosis can identify all the root-cause metrics of SWLT with highest confidence level (lowest confidence threshold) compared to other time series distance based analysis algorithms, and reduce the candidate problem metrics to about 10%.

2 SMALL-WINDOW TAIL LATENCY

Long-tail phenomenons have been broadly observed in data-center [1, 7–10, 21, 24, 36, 37, 43, 46, 47], we focus on studying the tail latency at extremely small timescales (e.g., 1 minute, even 1 second) for web services deployed in container based microservice platforms.

Datasets. We have running tens of thousands of web services in our microservice platforms as shown in Figure 1, 90% applications are deployed in a cluster of less than 100 containers. So we select 4 types of representative applications with different cluster sizes, from small, medium, big, to super as shown in Table 1. We identify SWLT, and manually verify the problematic containers and root-cause metrics. We record 13 types of metrics for each container, such as CPU utilization, memory usage, Disk I/O utilization etc., all the metrics are observed with 1-minute resolution.

Observations. Figure 2 shows the 99th percentile response time (called *TP*99, which are calculated during 1 minute. Here, we use *TP*99 as an example of tail latency) and its variability of each container for the four application datasets.

From Figure 2a, we can see *only a very small number of containers are problematic when SWLT occurs.* For application Small, Medium, Big, there is only 1 container with problems most of the time. Application Super can have up to 5 problematic containers, which can have the problem at the same time. For all cases, the number of problematic containers is very small compared to the total number of containers. Figure 2a also shows that *the small-window tail latency of each container changes very sharply.* For example, the tail latency increases from several milliseconds to approximately 2.5 seconds during the 25-minute monitoring period at the time of 18 for application Small.

Table 1: Datasets from real-production web services.

DataSet	Small	Medium	Big	Super
Problematic Containers (#)	2	2	7	13
Total Containers (#)	15	55	99	260
Root-cause Metrics (#)	2	3	12	17
Total Metrics (#)	13*15	13*55	13*99	13*260
Alarm Windows (minutes)	15	30	10	5

To analyze the variability of TP99 for each container, We calculate the coefficient of variation (COV) [44] for each container in the container clusters for the four applications. *COV* is the ratio of the standard deviation and the mean of the dataset, if $COV > 1$, it means high-variance distribution, otherwise, it's low-variance. In Figure 2b, y-axis is *COV* for each container. We can see *the small-window tail latency for all the applications is highly variant.* For application Super, there are 24 high-variance containers, Big is 12, Medium is 18, Small is 7.

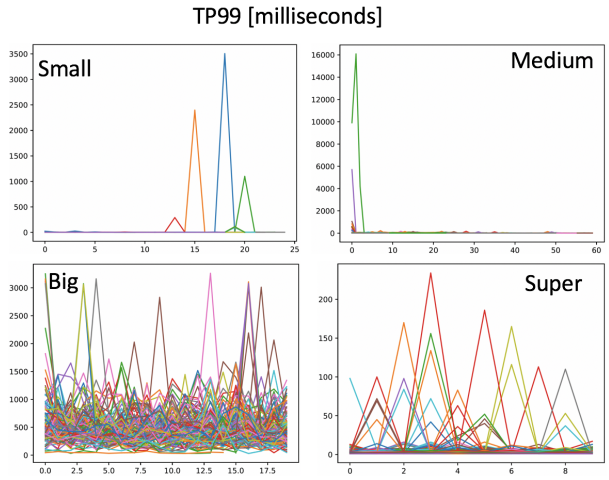
Due to heavy-tail and high-variance characterization, identifying root-causes of SWLT seems like finding a needle from haystack. Developing an intelligent and real-time analysis system to automatically diagnose root-cause metrics of SWLT from the application cluster for large-scale web services has an important implication for application administrators to identify SLO violations.

Goals. To diagnose root-causes of SWLT with high-variance and frequent-shift in large-scale microservice platforms, the objectives of designing the diagnosis algorithm and system are two-fold: (1) the algorithm and the system can quickly diagnose root-causes at runtime with low computation cost, (2) the algorithm can significantly reduce the problem space (metrics) while guaranteeing not to miss any actual root-cause.

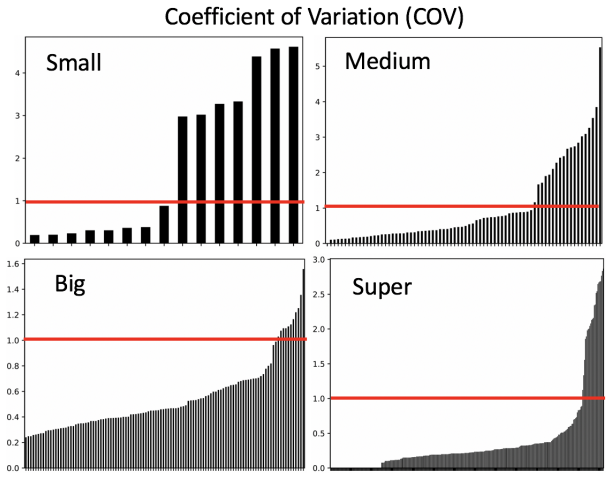
3 ϵ-DIAGNOSIS SWLT

We propose an unsupervised and low-cost root-cause analysis algorithm, ϵ-Diagnosis, which can diagnose root-causes of SWLT for large-scale web services at runtime.

We assume that once the long response time occurs, root-cause metrics in the problematic container might significantly change between the abnormal and normal periods. Such that, root-cause analysis is to identify the significantly-changed metrics. Therefore, in order to identify the significantly-changed metrics as the candidate root-cause metrics, we can use two-sample null hypothesis test (abnormal and normal samples) [25], which can use various time series similarity measurement algorithms [17]. Here, we adopt ϵ-statistics test (energy distance correlation) algorithms.



(a) TP99 of 4 APPs. Each line represents the $TP99$ for a container.



(b) COV for 4 APPs. Each bar represents the COV for a container.

Figure 2: $TP99$ and COV for 4 application clusters.

Detecting SWLT. To diagnose the root-cause of long-tail latency, the first task is to detect the long-tail latency. Threshold-based detection [12] is the simplest and widely-used anomaly detection approach. Our diagnosis system provides the alarm threshold interface (e.g., *TP99 threshold*) for the application administrators to detect SWLT for their web services. For example, the administrators can define a rule: if the 99th percentile response time during 1 minute for one service is bigger than 2000ms, it will trigger an alarm. Furthermore, the system provides *alarm window* interface to aggregate the alarm number for the same type of alarms during the alarm window. For example, if the alarm window is 15 minutes, the alarm system only reports the first alarm during the 15-minute time window. Once we detect a long-tail latency, it triggers ϵ-Diagnosis algorithm to analyze root-causes.

Selecting two samples from the snapshot. To identify the significantly-changed metrics as the root-cause, we store the context of web applications in the snapshot for comparison analysis when long-tail latency occurs. The snapshot includes various time-series metrics data, extracted from both the application layer and infrastructure layer cross millions of containers.

For application layer, we aggregate various performance metrics from the log files of various servers (e.g., Apache, Tomcat, MySQL), the metrics include throughput, QPS, concurrent loads, response time, number of error log, number of log, number of database connections etc. For infrastructure layer, we record all the metrics about CPU, memory, disk, network of each container. We formulate these time-series metrics as a time-series vector by

$$S_{(t)} = [x_1, x_2, ..., x_n]$$

where x_i is the aggregation value during the statistical/sampling period for each metric.

The system provides the alarm window interface to aggregate the alarm number for the same type of alarms during the alarm window. Thus we can guarantee there must exist some anomaly metrics during the alarm window leading to long tail latency. We use time-series metrics data during the alarm window as the *abnormal sample* (S_A). We choose time-series metrics data during the normal period from the snapshot as the *normal sample* (S_N).

Two-sample null hypothesis test. We would like to find the significantly-changed time series metrics as root-cause metrics. The two-sample test [35] is one of the most commonly used hypothesis tests when you want to compare two independent datasets to see if they are statistically similar or not. So we use two-sample test as the algorithm flow of ϵ-Diagnosis. The hypothesis of the two-sample test can be expressed by:

$$\begin{cases} H_0: & S_A = S_N \ . \\ H_a: & S_A \neq S_N \ . \end{cases} \tag{1}$$

if H_0 is true, it means that abnormal samples (S_A) and normal samples (S_N) are statistically equal. Otherwise, they are statistically different.

Further, we can use permutation test [30] or bootstrapping [39] to do hypothesis test, and calculate the p-value (P) using the sampling distribution of the test statistic under the null hypothesis. For different confidence level, we can get a confidence threshold to accept or reject the hypothesis. If the distribution of the test statistic of Hypothesis(1) is symmetric about 0, the test statistic is a two-sided test; otherwise it is a one-side test. For example, if the confidence level is 99% and it is a two-sided test, the confidence threshold (α) is 0.05,

$$\begin{cases} P < \alpha, S_A \neq S_N \ . \\ P \geqslant \alpha, S_A = S_N \ . \end{cases} \tag{2}$$

Here, if $P < \alpha$, we reject the hypothesis, which means that the anomaly sample and the normal sample are significantly different, so the corresponding metrics of the samples are potential root-causes. The overall ϵ-Diagnosis algorithm can be described in Algorithm 1.

The confidence threshold (α) plays a critical role in root-cause accuracy for our root-cause service. If α is too low, we might miss some true root-cause metrics, leading to higher false negative. If α is too high, we might consider more metrics as the potential root-cause, leading to higher false positive. So we have to make a trade-off between recall and precision. In the evaluation section,

Algorithm 1 Pseudo-code for the ϵ-Diagnosis algorithm

Input: small-window long-tail latency, M time-series metrics of N containers, confidence threshold α, alarm window
Output: candidate root-cause metrics, problematic containers

procedure ϵ-DIAGNOSIS
  for $ContainerN \leftarrow 1$ to N **do**
    S_A = **getAnomalySample**
    S_N = **getNormalSample**
    for $MetricM \leftarrow 1$ to M **do**
      (ρ (S_A, S_N), P) = Calculate Energy distance correlation coefficient of S_A and S_N using Equation (3) with P-value
      if $P < \alpha$ **then**
        /* Reject Hypothesis S_A != S_N*/
        Add $MetricM$ as a candidate root-cause metric
        Add $ContainerN$ as a candidate problematic container
      else
        /* Accept Hypothesis $S_A = S_N$ */
      end if
    end for
  end for
end procedure

we empirically get the optimal confidence threshold α. We leave the auto-tuning of α as our future work.

ϵ-Statistics (Energy distance correlation). There are a lot of literature work on time series similarity measurement [17]. From observations in Figure 2) in Section 2, we note that the tail latency at extremely small timescale for web services is heavy-tailed and highly-variant. Energy distance based ϵ-statistics is specially well-suited for heavy-tailed and highly-variant datasets with a low computation cost [41]. Thus, we adopt ϵ-statistics test (energy distance correlation) to measure the similarity of two samples.

Energy distance is a variation of squared pairwise distance. The Energy correlation coefficient ($\rho(S_A, S_N)$) between anomaly samples (S_A) and normal samples (S_N) is defined as the square root of,

$$\rho^2\,(S_A, S_N) = \begin{cases} \frac{cov^2(S_A,S_N)}{\sqrt{\sigma^2(S_A)\sigma^2(S_N)}}, & \sigma^2\,(S_A)\,\sigma^2\,(S_N) > 0\,. \\ 0, & \sigma^2\,(S_A)\,\sigma^2\,(S_N) = 0\,. \end{cases} \quad (3)$$

where cov is the covariance of the two samples, σ is the standard deviation of each sample. The benefit and speciality of ϵ-statistics test is distribution-free, scale-equivariant and rotation-invariant. Therefore, it is suitable for diagnosing root-causes of long-tail latency for response time sensitive user-facing web services, in which case the tail latency is aggregated at small timescales, e.g., during 1 minute or 1 second.

Real-time diagnosis system. We implement an automatic and intelligent diagnosis system at runtime using ϵ-Diagnosis algorithm to analyze root-causes of the long-tail latency at extremely small timescales (e.g., 1 minute or 1 second) for tens of thousands web applications deployed in our microservice platforms managed by

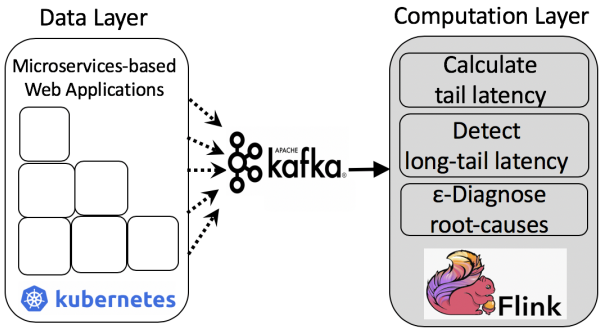


Figure 3: System architecture.

Kubernetes [1] as shown in Figure 3. It consists of two main components: data layer and computing layer.

To support population-scale applications, enabling their SLO violation monitoring and delivering results as fast as possible (seconds) are non-trivial. One challenge is to transfer the large volume operation data from the distributed containers. To reduce the amount of data to transfer, we adopt Apache thrift [2] in the data collection agents. Thrift can work with plug in serialization protocols and data compression (e.g. gzip). We observe the compression ratio is around 1/23. The data is pipelined by kafka [3]. Kafka is able to be scaled quickly and easily without incurring any downtime, and handle many terabytes of data with consistent performance.

In computation layer, we calculate the small-window tail latency, detect SWLT by comparing the tail latency with the predefined alarm thresholds, and use ϵ-Diagnosis algorithm to identify the significantly-changed metrics as root-cause metrics. We adopt Apche flink [4] to implement these functions in computation layer, which can process the stream data with high performance and low latency. All the long-tail alarms are stored in event database (e.g., MySQL), all the time-series metrics are stored in time-series database (e.g., ClickHouse [5]), time-series database [32] can provide scalable performance for analytics and aggregation of time-series.

4 EVALUATIONS

4.1 Operational Data in Real-production

Our monitoring cluster consists of approximately 300 containers, which can monitor the small-window tail latency of approximately 30000 web services in our microservice platforms as shown in Figure 1. We deployed the root-cause analysis system in the monitoring cluster. Figure 4 shows the hourly amount of long-tail latency alarms in three days for our web services. We detected approximately 30,000 long-tail latency with a peak of 50,000 in every hour. The request rate of root-cause diagnosis in total is 8.3 per second with a peak of 13.9 per second.

Execution time. One goal of designing the algorithm and the system is to provide an diagnosis service to identify root-causes of

[1] https://kubernetes.io/
[2] https://thrift.apache.org/
[3] https://kafka.apache.org/
[4] https://flink.apache.org/
[5] https://clickhouse.yandex/

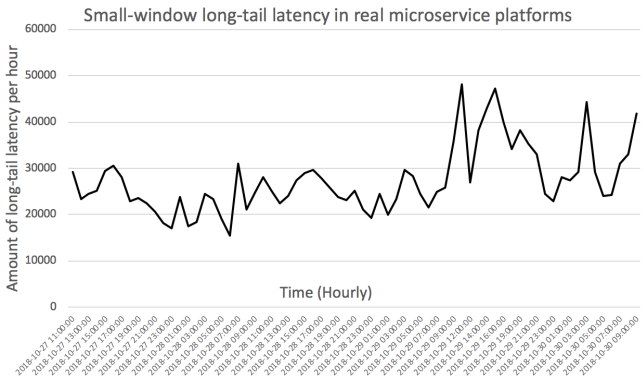


Figure 4: Service rate of ϵ-Diagnosis in a real-life production.

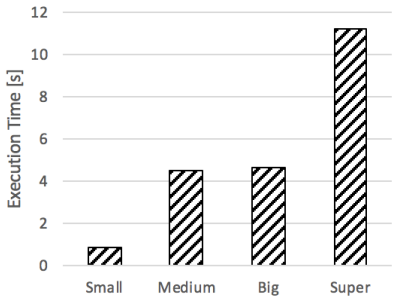


Figure 5: Execution time of ϵ-Diagnosis, in seconds.

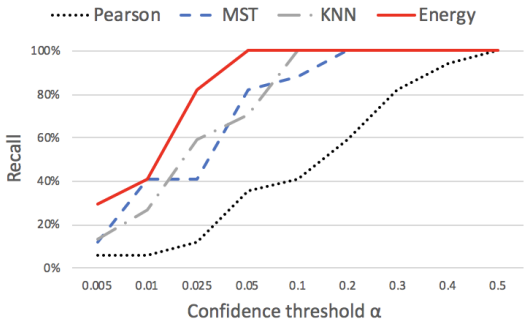


Figure 6: Energy can reach 100% recall quickly as α increases.

SWLT in mocroservice platforms at runtime. So we evaluate the computation cost for ϵ-Diagnosis.

We run the ϵ-Diagnosis system in a container equipped with a quad-core Intel Core i7 2.8 GHz CPU and 16 GB 2133 MHz LPDDR3 memory. Figure 5 shows the execution time for ϵ-Diagnosis algorithm for the datasets in Table 1. For Small application, the execution time of ϵ-Diagnosis is less than 1 second. As the number of the containers increases, the running time increases. Medium and Big are in the same level, since the sample size (alarm window) reduces from 30 to 10 as shown in Table 1, although the container amount increases from Medium to Big. In our microservice platforms, 80% applications are small web applications. ϵ-Diagnosis can finish the analysis within 1 second, so we use 10 containers to serve the requests of root-cause analysis as shown in Figure 4.

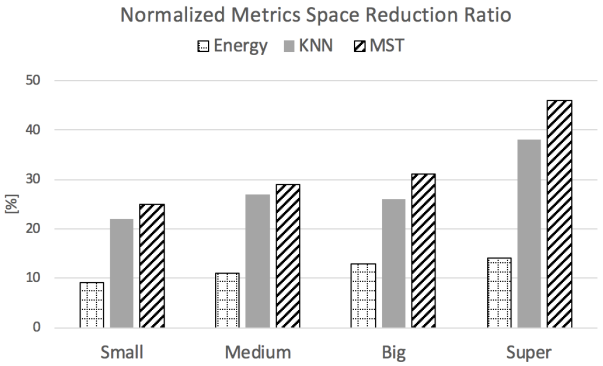


Figure 7: Energy can reduce metrics to approximately 10%.

4.2 Performance of Algorithms

Next, we compare the performance of ϵ-Diagnosis with other time-series distance based root-cause analysis algorithms.

Experimental Methodology. We use an empirical approach in our experiments to define α. For example, we define α as a set of values [0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5], which are corresponding to the confidence level of two-sample test [99%, 98%, 96%, 90%, 80%, 60%, 40%, 20%, 0%] (two-sides test). In our experiments, we gradually increase α and evaluate the performance of the algorithms.

We compare the performance of the ϵ-Diagnosis algorithm (based on energy distance) with other three time-series similarity measurements. Pearson's distance is the simplest one, defined by,

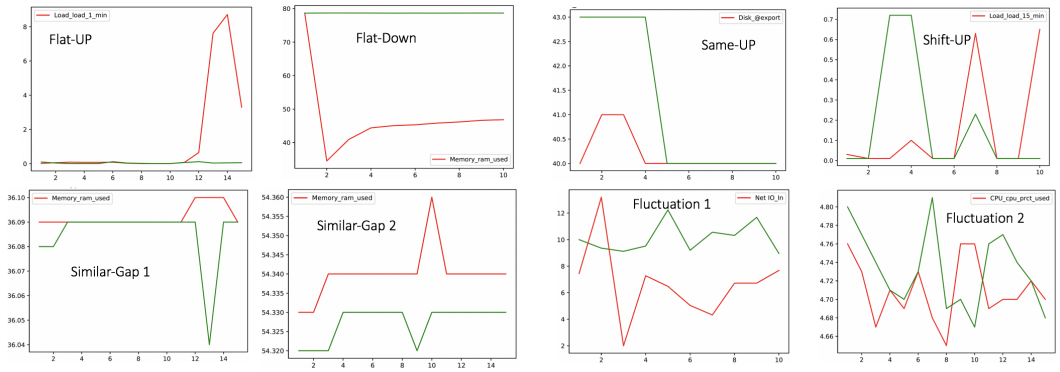
$$\rho\,(S_A, S_N) = \frac{cov(S_A, S_N)}{\sigma(S_A)\sigma(S_N)} \tag{4}$$

where cov is the covariance of the two samples, σ is the standard deviation of each sample. K-NN [16, 25] and MST [15] algorithms both discretize the time-series samples, transform the distance of two time-series samples into inter-point distance, and use graph-theoretic algorithm to measure their similarity. KNN algorithm uses nearest neighbours to model the inter-point distance of two time series, meanwhile MST algorithm uses minimum spanning tree to represent their inter-point distance.

Results. We require that the algorithm can significantly reduce the problem space from million-level metrics while guaranteeing not missing any actual root-cause, namely guaranteeing 100% recall.

Figure 6 shows the recall of the four time-series measurement algorithms on all four datasets. As the confidence threshold α increases, the recall increases. We note that Energy can achieves 100% recall when α equals 0.05. The two graphic-theory based algorithm (MST and KNN) are 0.2 and 0.1, respectively. As α increases, Energy can identify all the root-causes at the fastest speed. Pearson can not find all the actual root-causes until α equals 0.5. So Pearson can not fit our case and operational environments, and we will omit the evaluation of Pearson.

Figure 7 shows that how much the time-series similarity measurement algorithms can narrow down the candidate problematic metrics from the total metrics under the condition of 100% recall (For Energy, KNN, MST, α is 0.05, 0.1, and 0.2). Energy can reduce metrics to approximately 10%, outperforming the other two, since



Algorithm	*Flat-UP*	*Flat-Down*	*Same-UP*	*Shift-UP*	*Similar-Gap 1*	*Similar-Gap 2*	*Fluctuation 1*	*Fluctuation 2*
Pearson	0.903	0.041	0.060	0.924	0.072	0.073	0.867	0.901
MST	0.568	0	0.774	0.582	0.430	0	0	0.737
KNN	0.063	0.011	0.033	0.296	0.103	0.106	0	0.712
Energy	0.043	0	0.061	0.047	0.031	0.001	0	0.211

Figure 8: Various two-sample patterns. Each line represents a sample. Smaller P, more significantly different time series.

Table 2: Precision of the algorithms for the four datasets.

DataSet	Small	Medium	Big	Super
MST (%)	62	59	67	66
KNN (%)	76	67	65	62
Energy (%)	79	72	76	72

it uses the smallest α, counting smaller metrics as the potential metrics. For the same reason, Energy can also get the best precision (e.g., 79% for the small application) as shown in Table 2.

To explain why Energy can outperform the other three algorithms, Figure 8 lists several representative time-series patterns from the infrastructure layer metrics and their corresponding p-value (P). Smaller P, more significantly different the two samples. We can note Energy is more sensitive than the other algorithms for all of the cases, specifically, it can identify more patterns as different samples, so it can achieve 100% recall with the smallest α. Meanwhile, there also exists false positive for Energy, since it might falsely identify some more metrics patterns as root-causes (e.g., "Fluctuation 2"). Pearson can not identify the the difference of "Flat-UP", which is disappointing, because several root-cause metrics have this type of peak pattern from our observations in our microservice platforms, it will miss this type of root-cause metrics.

5 RELATED WORK

Traditional rule (or signature) based performance diagnosis approaches for distributed systems, heavily depend on the operational fault knowledge base (such as Fingerprint [2, 6, 11]), and detect the anomaly and performance bottlenecks through tracing the request process flowing in large-scale distributed system (such as Google's Dapper [38], Mace [23], Pinpoint [5], Pip [33], Stardust [34], X-Trace [14], Whodunit [4], Retro [26], Pivot Tracing [27], Quanto [13]). However, it is hard to build both operational fault knowledge base and call relationship in complex micro-service based applications with tens of thousands of invocation graphs [31]. Compared to rule based root-cause analysis, we do not need any operational knowledges and the training process.

Models and techniques for root-cause analysis are extensively summarized in the survey paper [40]. Luo et al. [25] model incident diagnosis as a two-sample problem and apply k-NN statistic based method to correlate event and time-series. Daily chronic problems [22] are statistically diagnosed, and seasonal KPIs alarms [45] are detected by an unsupervised anomaly detection algorithm based on VAE (Variational Auto-Encoder). Compared to statistical learning based root-cause analysis, our approach is an unsupervised approach with low computation cost. More importantly, our diagnosis target is more fine-grained long-tail latency problem at extremely small timescale (e.g., 1-minute or 1-second) for the large-scale on-line web services. To the best of our knowledge, no current root-cause analysis algorithm can scale to such granularity.

6 CONCLUSIONS

We identify a new type of SLO violation for web services, SWLT in large-scale microservice platforms, which has a heavy-tail and high-variance characterization. To diagnose root-causes of SWLT, we propose an unsupervised and low-cost algorithm–ϵ-Diagnosis, using two samples test algorithm based on ϵ-statistics. We develop and deploy the root-cause diagnosis system in the microservice platforms to help identify the potential root-causes of SWLT from millions of metrics for tens of thousands of web services. Through evaluating with real-production datasets, we show that ϵ-Diagnosis can outperform other time-series distance based root-cause analysis algorithms in finding all the actual root-cause metrics of SWLT with highest confidence level.

REFERENCES

[1] Salman A Baset. 2012. Cloud SLAs: present and future. *ACM Operating Systems Review* (2012).

[2] Peter Bodik, Moises Goldszmidt, Armando Fox, Dawn B Woodard, and Hans Andersen. 2010. Fingerprinting the datacenter: automated classification of performance crises. In *Proceedings of the 5th European conference on Computer systems*. ACM, 111–124.

[3] Valeria Cardellini, Emiliano Casalicchio, Francesco Lo Presti, and Luca Silvestri. 2011. Sla-aware resource management for application service providers in the cloud. In *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*. IEEE, 20–27.

[4] Anupam Chanda, Alan L Cox, and Willy Zwaenepoel. 2007. Whodunit: Transactional profiling for multi-tier applications. In *ACM SIGOPS Operating Systems Review*, Vol. 41. ACM, 17–30.

[5] Yen-Yang Michael Chen, Anthony J Accardi, Emre Kiciman, David A Patterson, Armando Fox, and Eric A Brewer. 2004. Path-based failure and evolution management. (2004).

[6] Ira Cohen, Steve Zhang, Moises Goldszmidt, Julie Symons, Terence Kelly, and Armando Fox. 2005. Capturing, indexing, clustering, and retrieving system history. In *ACM SIGOPS Operating Systems Review*, Vol. 39. ACM, 105–118.

[7] Kristal Curtis, Peter Bodík, Michael Armbrust, Armando Fox, Mike Franklin, Michael Jordan, and David Patterson. 2010. *Determining SLO Violations at Compile Time.*

[8] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* (2013).

[9] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: amazon's highly available key-value store. In *ACM SIGOPS operating systems review*, Vol. 41. ACM, 205–220.

[10] Pamela Delgado, Diego Didona, Florin Dinu, and Willy Zwaenepoel. 2018. Kairos: Preemptive Data Center Scheduling Without Runtime Estimates. In *ACM SoCC*.

[11] Songyun Duan and Shivnath Babu. 2008. Guided problem diagnosis through active learning. In *Autonomic Computing, 2008. ICAC'08. International Conference on*. IEEE, 45–54.

[12] Hector Fernandez, Corina Stratan, and Guillaume Pierre. 2014. Robust performance control for web applications in the cloud. In *4th International Conference on Cloud Computing and Services Science*.

[13] Rodrigo Fonseca, Prabal Dutta, Philip Levis, and Ion Stoica. 2008. Quanto: Tracking Energy in Networked Embedded Systems.. In *OSDI*, Vol. 8. 323–338.

[14] Rodrigo Fonseca, Michael J Freedman, and George Porter. 2010. Experiences with Tracing Causality in Networked Services. *INM/WREN* 10 (2010), 10–10.

[15] Jerome H Friedman and Lawrence C Rafsky. 1979. Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *The Annals of Statistics* (1979), 697–717.

[16] Jerome H Friedman, Lawrence C Rafsky, et al. 1983. Graph-theoretic measures of multivariate association and prediction. *The Annals of Statistics* 11, 2 (1983), 377–391.

[17] Tak-chung Fu. 2011. A review on time series data mining. *Engineering Applications of Artificial Intelligence* 24, 1 (2011), 164–181.

[18] Anshul Gandhi, Yuan Chen, Daniel Gmach, Martin Arlitt, and Manish Marwah. 2011. Minimizing data center SLA violations and power consumption via hybrid resource provisioning. In *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 1–8.

[19] Anshul Gandhi, Yuan Chen, Daniel Gmach, Martin Arlitt, and Manish Marwah. 2012. Hybrid resource provisioning for minimizing data center SLA violations and power consumption. *Sustainable Computing: Informatics and Systems* 2, 2 (2012), 91–104.

[20] Vikas Jaiman, Sonia Ben Mokhtar, Vivien Quéma, Lydia Chen, and Etienne Riviere. 2018. Héron: Taming Tail Latencies in Key-Value Stores under Heterogeneous Workloads. In *International Symposium on Reliable Distributed Systems (SRDS) 2018*.

[21] Keon Jang, Justine Sherry, Hitesh Ballani, and Toby Moncaster. 2015. Silo: Predictable message latency in the cloud. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 435–448.

[22] Soila P Kavulya, Scott Daniels, Kaustubh Joshi, Matti Hiltunen, Rajeev Gandhi, and Priya Narasimhan. 2012. Draco: Statistical diagnosis of chronic problems in large distributed systems. In *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*. IEEE, 1–12.

[23] Charles Edwin Killian, James W Anderson, Ryan Braud, Ranjit Jhala, and Amin M Vahdat. 2007. Mace: language support for building distributed systems. In *ACM SIGPLAN Notices*, Vol. 42. ACM, 179–188.

[24] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. 2015. Heracles: Improving resource efficiency at scale. In *ACM SIGARCH Computer Architecture News*, Vol. 43. ACM, 450–462.

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

[26] Jonathan Mace, Peter Bodik, Rodrigo Fonseca, and Madanlal Musuvathi. 2015. Retro: Targeted Resource Management in Multi-tenant Distributed Systems.. In *NSDI*. 589–603.

[27] Jonathan Mace, Ryan Roelke, and Rodrigo Fonseca. 2015. Pivot tracing: Dynamic causal monitoring for distributed systems. In *Proceedings of the 25th Symposium on Operating Systems Principles*. ACM, 378–393.

[28] Aniket Mahanti, Niklas Carlsson, Anirban Mahanti, Martin Arlitt, and Carey Williamson. 2013. A tale of the tails: Power-laws in internet measurements. *IEEE Network* 27, 1 (2013), 59–64.

[29] Anton Michlmayr, Florian Rosenberg, Philipp Leitner, and Schahram Dustdar. 2009. Comprehensive qos monitoring of web services and event-based sla violation detection. In *Proceedings of the 4th international workshop on middleware for service oriented computing*. ACM, 1–6.

[30] Anders Odén, Hans Wedel, et al. 1975. Arguments for Fisher's permutation test. *The Annals of Statistics* 3, 2 (1975), 518–520.

[31] Fábio Oliveira, Sahil Suneja, Shripad Nadgowda, Priya Nagpurkar, and Canturk Isci. 2017. Opvis: extensible, cross-platform operational visibility and analytics for cloud. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track*. ACM, 43–49.

[32] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. 2015. Gorilla: A fast, scalable, in-memory time series database. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1816–1827.

[33] Patrick Reynolds, Charles Edwin Killian, Janet L Wiener, Jeffrey C Mogul, Mehul A Shah, and Amin Vahdat. 2006. Pip: Detecting the Unexpected in Distributed Systems.. In *NSDI*, Vol. 6. 9–9.

[34] Raja R Sambasivan, Alice X Zheng, Michael De Rosa, Elie Krevat, Spencer Whitman, Michael Stroucken, William Wang, Lianghong Xu, and Gregory R Ganger. 2011. Diagnosing Performance Changes by Comparing Request Flows.. In *NSDI*, Vol. 5. 1–1.

[35] Howard J Seltman. 2012. Experimental design and analysis. *Online at: http://www.stat. cmu. edu/, hseltman/309/Book/Book. pdf* (2012).

[36] Huasong Shan, Qingyang Wang, and Calton Pu. 2017. Tail attacks on web applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1725–1739.

[37] Huasong Shan, Qingyang Wang, and Qiben Yan. 2017. Very Short Intermittent DDoS Attacks in an Unsaturated System. In *International Conference on Security and Privacy in Communication Systems*. Springer, 45–66.

[38] Benjamin H Sigelman, Luiz Andre Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, and Chandan Shanbhag. 2010. *Dapper, a large-scale distributed systems tracing infrastructure.* Technical Report. Technical report, Google, Inc.

[39] Kesar Singh and Minge Xie. 2008. Bootstrap: a statistical method. *Unpublished manuscript, Rutgers University, USA. Retrieved from http://www. stat. rutgers. edu/home/mxie/RCPapers/bootstrap. pdf* (2008).

[40] Marc Solé, Victor Muntés-Mulero, Annie Ibrahim Rana, and Giovani Estrada. 2017. Survey on models and techniques for root-cause analysis. *arXiv preprint arXiv:1701.08546* (2017).

[41] GJ Székely. 2003. E-Statistics: The energy of statistical samples. *Bowling Green State University, Department of Mathematics and Statistics Technical Report* 3, 05 (2003), 1–18.

[42] Jörg Thalheim, Antonio Rodrigues, Istemi Ekin Akkus, Pramod Bhatotia, Ruichuan Chen, Bimal Viswanath, Lei Jiao, and Christof Fetzer. 2017. Sieve: actionable insights from monitored metrics in distributed systems. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*. ACM, 14–27.

[43] Balajee Vamanan, Jahangir Hasan, and TN Vijaykumar. 2012. Deadline-aware datacenter tcp (d2tcp). *ACM SIGCOMM Computer Communication Review* 42, 4 (2012), 115–126.

[44] Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari. 2009. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 conference on USENIX Annual technical conference*. USENIX Association, 28–28.

[45] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 187–196.

[46] Timothy Zhu, Daniel S Berger, and Mor Harchol-Balter. 2016. SNC-Meister: Admitting more tenants with tail latency SLOs. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*. ACM, 374–387.

[47] Timothy Zhu, Alexey Tumanov, Michael A Kozuch, Mor Harchol-Balter, and Gregory R Ganger. 2014. Prioritymeister: Tail latency qos for shared networked storage. In *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 1–14.