CoFlux: Robustly Correlating KPIs by Fluctuations for Service Troubleshooting

Ya Su[16], Youjian Zhao[16], Wentao Xia[16], Rong Liu[2], Jiahao Bu[16], Jing Zhu[16], Yuanpu Cao[3], Haibin Li[146], Chenhao Niu[16], Yiyin Zhang[5], Zhaogang Wang[5], Dan Pei[16*]
[1]Tsinghua University, [2]Stevens Institute of Technology, [3]Nankai University, [4]National University of Defense Technology, [5]Alibaba Group, [6]Beijing National Research Center for Information Science and Technology (BNRist)

ABSTRACT

Internet-based service companies monitor a large number of KPIs (Key Performance Indicators) to ensure their service quality and reliability. Correlating KPIs by fluctuations reveals interactions between KPIs under anomalous situations and can be extremely useful for service troubleshooting. However, such a *KPI flux-correlation* has been little studied so far in the domain of Internet service operations management. A major challenge is how to automatically and accurately separate fluctuations from normal variations in KPIs with different structural characteristics (such as seasonal, trend and stationary) for a large number of KPIs. In this paper, we propose CoFlux, an unsupervised approach, to automatically (without manual selection of algorithm fitting and parameter tuning) determine whether two KPIs are correlated by fluctuations, in what temporal order they fluctuate, and whether they fluctuate in the same direction. CoFlux's robust feature engineering and robust correlation score computation enable it to work well against the diverse KPI characteristics. Our extensive experiments have demonstrated that CoFlux achieves the best F1-Scores of 0.84 (0.90), 0.92 (0.95), 0.95 (0.99), in answering these three questions, in the two real datasets from a top global Internet company, respectively. Moreover, we showed that CoFlux is effective in assisting service troubleshooting through the applications of alert compression, recommending Top N causes, and constructing fluctuation propagation chains.

CCS CONCEPTS

• **Networks** → Network monitoring; • **Social and professional topics** → Software maintenance.

KEYWORDS

key performance indicator, service operation and management, time series, fluctuation correlation, service troubleshooting

*Dan Pei is the corresponding author.

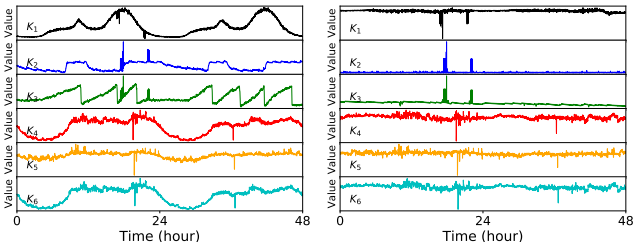
IWQoS '19, June 24–25, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6778-3/19/06…$15.00
https://doi.org/10.1145/3326285.3329048

1 INTRODUCTION

Large-scale Internet-based service companies provide a large number of services and applications through thousands of servers [1, 2]. However, service interruptions are inevitable due to many reasons, such as network outages, server breakdown and malicious attacks [3, 4]. To stay competitive, operators in these companies take great efforts to keep their services reliable. They constantly monitor *KPIs* (*Key Performance Indicators*), time series that contain service quality measures (such as success ratio and number of requests [5]). An incident can trigger fluctuations in some KPIs, and the fluctuations may propagate to more related KPIs to form interweaved fluctuations, which complicate incident resolution and root cause analysis [6]. Without understanding their relationships, it is challenging for operators to prioritize alerts triggered by interweaved fluctuations and identify possible incident impact.

In this paper, we focus on analyzing the correlation between KPI fluctuations. Fluctuations are deviations from expected values in a KPI. Large fluctuations likely are anomalies and often presented as sudden spikes or dips [3]. We define that two KPIs are ***flux-correlated*** if fluctuations in one KPI are correlated with those in the other over a period of time. Note that *flux-correlation is different from KPI correlation which is often calculated using the raw values of KPI.* Two flux-correlated KPIs may be uncorrelated when their raw values are considered. For example, as shown in Fig. 1(a), the raw values of K_1 and K_2 (or K_1 and K_3) are not correlated by algorithms, for example, Pearson Correlation [7]. However, the fluctuations of these two KPIs, as shown in Fig. 1(b), are highly correlated since they often happen synchronously or in sequence. As a result, flux-correlation should be analyzed based on fluctuations rather than raw KPIs.



(a) Six KPIs

(b) Fluctuations (Flux-feature) of the KPIs in (a), obtained by methods in Section 4.1.1

Figure 1: Examples of Flux-correlation.

Flux-correlation can *assist* operators in service troubleshooting in many aspects. An Internet-based service often consists of a stack of software modules deployed on multiple machines [1]. The service is monitored by a number of KPIs at different levels, such

as machine-level and module-level [2]. When a failure happens, excessive and often correlated alerts are generated from these KPIs. These redundant alerts compete for operators' attention and distract them from diagnosis. With flux-correlation, first, we can cluster KPIs based on their flux-correlation results for alert compression [8]. If two or more KPIs are flux-correlated, they can be grouped into a cluster, alerts are raised based on the cluster as a whole as opposed to individual KPIs, greatly reducing operators' workload. For example, in Fig. 1(a), K_1, K_2 and K_3 can be grouped into one cluster, while K_4, K_5 and K_6 form another cluster. Second, when anomalies occur at a KPI, operators can check the fluctuations of its Top N flux-correlated KPIs to narrow down their analysis scope. Third, in order to infer root causes, we can construct a fluctuation propagation chain by the temporal order in which KPIs fluctuate (*e.g.*, first K_4, then K_5 and finally K_6, as shown in Fig. 1(a)). As suggested by [6], within a service, KPIs with the earliest anomalous changes can better reflect actual root causes.

The idea of flux-correlation has also attracted interest from other fields such as system modeling [9] and stock market [10]. [9] aims at characterizing the anomalous interactions among KPIs in a system and constructing Structure-of-Influence Graphs (SIG). The anomaly signals of a KPI are computed by KL divergence between the distribution of data in a sliding window and the *entire* distribution of the KPI. As a result, [9] focuses on the outliers [11] (or novelty anomalies) and cannot identify anomalies that are mainly judged by *partial* history data. The statistical models ARCH [10], VARMA [12], and CoIntegration [13] have been developed to analyze correlations in stock price changes, or study volatility across stock markets, *etc.* However, these models cannot solve the flux-correlation problem well in our scenario because they cannot accurately identify fluctuations of KPIs. Another related area is the correlation analysis between raw time series. Pearson [7] and Spearman Correlation focus on linear and rank correlation between raw KPIs respectively. They cannot work well with fluctuations, especially when two KPIs have different patterns, such as seasonality and phase shift. Granger causality [14] identifies the correlation between two KPIs by whether one KPI is useful for forecasting another. However, in our context, fluctuations are usually caused by unexpected accidents and may not be predicted by regression. Therefore, these methods cannot solve our flux-correlation problem well.

This paper fills the gap by focusing on KPI flux-correlation analysis. The major challenges are as follows.

- *There is no generic mechanism for fluctuation extraction.* Different KPIs often have different time series characteristics (such as seasonal, stationary and trend) [15] and need specific models to capture their fluctuations [3]. Typically 10k to 1 million KPIs exist in Internet companies, it would be time-consuming and unrealistic to manually search for a suitable fluctuation extraction model for each of them. Thus, it is challenging to design a *generic* fluctuation (*i.e.*, flux-feature) extraction mechanism that is robust against diverse KPI characteristics.
- *Flux-correlation should not be based on anomaly detection.* One might be tempted to apply J-measure [2] to the binary results of anomaly detection. However, selecting anomaly detection models and determining their thresholds often highly depend on specific characteristics of KPIs and users' sensitivity preferences [16]. Thus, flux-correlation should not be based on anomaly detection.
- *Two flux-correlated KPIs may present different interaction patterns.* In practice, KPIs may fluctuate synchronously or in sequence with a delay. Also, sometimes, when one KPI plunges, some KPIs follow with a big dip, while other KPIs respond with a spike. These patterns complicate flux-correlation analysis.

We make the following contributions in this paper when addressing the above challenges:

- To the best of our knowledge, this paper is the first attempt to formulate flux-correlation and study it in detail in the domain of Internet service operations management. We propose a novel unsupervised algorithm, CoFlux, to automatically and accurately measure the flux-correlation between KPIs based on their flux-features, without the need for anomaly detection. CoFlux can answer three questions. (1) Are two KPIs flux-correlated? If flux-correlated, (2) in what temporal order they fluctuate? (3) are they positively or negatively correlated?
- CoFlux includes a robust set of flux-features extracted from forecast errors of well-accepted time series models. Our intuition is that, at least one of these flux-features is capable of capturing a given KPI's fluctuations in a sufficiently accurate way, although we do not know which one beforehand. This makes our feature extraction robust against diverse KPI characteristics, and can work with a large number of KPIs without algorithm fitting and parameter tuning.
- CoFlux computes two KPIs' flux-correlation score by using the best Cross-correlation [17] score among different flux-features of two KPIs. Intuitively, if two KPIs X and Y are flux-correlated, then the best flux-features from X and Y are also correlated. There is no need to identify this best flux-feature pair beforehand or manually, making CoFlux robust against diverse KPI characteristics.
- Our extensive experiments have demonstrated that CoFlux achieves the best F1-Scores of 0.84 (0.90), 0.92 (0.95), 0.95 (0.99), in answering the three questions in flux-correlation analysis, in the two real datasets from a top global Internet company, respectively, significantly outperforming the baseline algorithms and their variants. Moreover, we showed that CoFlux is effective in assisting service troubleshooting through the applications of alert compression, recommending Top N causes, and constructing fluctuation propagation chains.

2 PROBLEM STATEMENT

In this section, we first define several key terms, including KPIs, flux-feature, flux-correlation, and two flux-Correlation patterns, and then describe our objective in this study.

KPIs and flux-features. A KPI is a series of successive observations collected at equal-spaced timestamps [2], denoted as $S = [s_1, s_2, \ldots, s_m]$, where s_i is the observation corresponding to

time index i for $i \in 1, 2, \ldots, m$, and m is the length of the KPI. If two KPIs have different sampling intervals, when considering their flux-correlation, we resample them using the Lowest Common Multiple of their intervals as the final interval. We define a *predicted KPI* as a sequence of predicted values produced by a time series forecast model, denoted as $P = [p_1, p_2, \ldots, p_m]$, where p_i is the predicted value of s_i. The prediction errors are denoted as $F = [f_1, f_2, \ldots, f_m]$, where $f_i = s_i - p_i$. While normal observations can be well predicted, fluctuations often generate prediction errors. Thus, prediction errors can be very useful in analyzing fluctuations. In this work, the prediction errors are treated as fluctuation features, referred to as *flux-features*. Accordingly, the time series model with specific parameters used for prediction is a *feature detector*. We can create many flux-features for a KPI through different time series forecast models.

Fig. 2 shows an example of a real KPI, its predicted values by Historical Average [18], and the corresponding flux-feature. Since flux-features are generated by time series forecast models, the model selection is critical to the flux-feature extraction. If a model can predict normal values precisely, then prediction errors can capture fluctuations well. We will discuss the model selection in detail later.

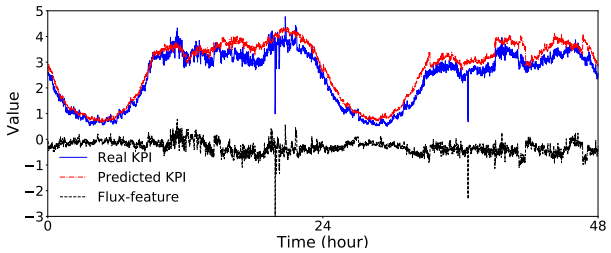


Figure 2: A real KPI, predicted KPI and flux-feature.

Flux-correlation. In this study, for a KPI pair, X and Y, we first determine whether their fluctuations are correlated, *i.e.*, *flux-correlated*. We define that X and Y are flux-correlated, denoted as $X \backsim Y$, if their flux-features are correlated. $X \nsim Y$ if X and Y are not flux-correlated. For example, Fig. 1 shows six KPIs and their flux-features. The flux-features of K_1 and K_2 look highly correlated, so they are flux-correlated, *i.e.*, $K_1 \backsim K_2$. However, K_1 and K_4 are not flux-correlated, *i.e.*, $K_1 \nsim K_4$.

When two KPIs are flux-correlated, we continue to understand their temporal order and whether they fluctuate in the same direction, as described below.

Temporal order of flux-correlation. The fluctuations from two KPIs may be synchronized or shifted by some intervals. Specifically, we use $X \rightarrow Y$ to denote the case that X fluctuates before Y. If their fluctuations happen simultaneously, we denote it as $X \leftrightarrow Y$. As is shown in Fig. 1(a), the fluctuations of K_2 and K_3 happen at the same time, *i.e.*, $K_2 \leftrightarrow K_3$, while K_1 fluctuates before K_2 and K_3, *i.e.*, $K_1 \rightarrow K_2$, $K_1 \rightarrow K_3$.

Direction of flux-correlation. When the fluctuations of X and Y are correlated, the correlation can be positive or negative. If the fluctuation of X is an increase but the corresponding fluctuation of Y is a decrease, their flux-correlation is negative, denoted as $X \overset{-}{\leftrightarrow} Y$ (or $X \overset{-}{\rightarrow} Y$). On the other hand, if the fluctuation of X is an increase (or decrease) and the corresponding fluctuation of Y is also an increase (or decrease), their flux-correlation is positive, denoted as $X \overset{+}{\leftrightarrow} Y$ (or $X \overset{+}{\rightarrow} Y$). In Fig. 1(a), K_2 and K_3 have positive flux-Correlation, but K_1 is negatively flux-correlated with K_2 and K_3.

With these definitions, we state that our objective is to solve the following problems, denoted as questions Q1~3:

(1) determining the existence of flux-correlation between two KPIs (Q1),
(2) for flux-correlated KPIs, understanding the temporal order of their fluctuations (Q2), and,
(3) determining the direction of flux-correlation(Q3).

3 CORE IDEA

To solve the problems mentioned above, we designed an unsupervised approach, called CoFlux, as shown in Fig. 3. The input to CoFlux is two KPIs. We first extract their flux-features through feature engineering, and then measure the correlation of these flux-features. In the end, we provide answers to questions Q1~3. Note that, CoFlux aims at determining the flux-correlation of KPIs over a long period, so it is an *offline* model and the timeliness is not our goal. Moreover, the flux-correlation results can be updated routinely (*e.g.*, once per week or month) in consideration of the current data.

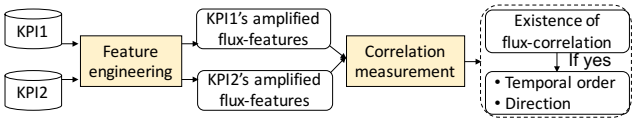


Figure 3: CoFlux architecture.

Feature engineering, as a key component in this architecture, is to find suitable time series models as our flux-feature detectors. Although many models have been proposed to forecast time series, for example, MA (Moving average), TSD (Time Series Decomposition) [19], each model only fits well with some types of characteristics of time series. For example, for the two KPIs shown in Fig. 1(a), K_4 has strong seasonality while K_5 is stable. For seasonal KPIs, models like TSD and Historical Average may be appropriate. Stable KPIs can be better predicted by MA or weighted MA because their prediction mainly relies on recent values [20]. Certainly, there are no generic models that can predict any type of KPIs accurately. As mentioned in Section 1, the challenge is that we have a large number of KPIs with different characteristics. It would be very time-consuming and unrealistic to manually search for a suitable forecast model for each of them.

Therefore, we cannot rely on a single time series model to extract flux-features. Instead, to make CoFlux as versatile as possible, we adopt several well-accepted models with corresponding parameters as flux-feature detectors. This design is based on the following two intuitions:

- For any given KPI, if we survey a wide range of models, there will be one or more of them that are capable of predicting its normal observations precisely enough and producing a close-to-truth flux-feature.
- If two KPIs, X and Y, are flux-correlated, then at least one flux-feature of X and one flux-feature of Y are correlated.

We will use extensive experiments to validate these two intuitions in Section 5.4. With flux-features extracted, we continue to improve them through denoising and amplification.

To determine flux-correlation, for each pair of KPIs, we calculate the pairwise correlation over their flux-features by Cross-correlation [17], a widely-accepted time series similarity measurement. Then, the *maximum* one is used to determine whether the two KPIs are flux-correlated. Since we have included many flux-feature detectors in CoFlux, some may be able to extract close-to-truth flux-features, while others may generate misleading flux-features due to the inaccurate forecast. Averaging over all flux-features or decision by majority vote would produce false negatives. Certainly, our approach of only considering the maximum one may cause potential false positives. We will use experiments in Section 5.4.1 to demonstrate that with good flux-feature detectors, false positives are very few.

4 DESIGN

In this section, we describe the details of the two components of CoFlux: feature engineering and correlation measurement. We start with feature engineering.

4.1 Feature engineering

Table 1: Prediction models and detectors.

Prediction models / # of detectors	Parameter Configurations
Diff / 2	last-day, last-week
Holt-Winters / 64	α, β, γ = {0.2, 0.4, 0.6, 0.8}
Historical average / 4	window = 1, 2, 3, 4 week(s)
Historical median / 4	
TSD / 4	
TSD median / 4	
Wavelet / 4	window = 1, 3, 5 ,7 day(s)
In total: 7 prediction models / 86 detectors	

4.1.1 Feature extraction. As discussed before, KPIs often have different shapes and time series characteristics. To understand their flux-correlation, we need to use different suitable time series models to generate flux-features. We carefully selected 7 widely-used models, as listed in Table 1. *Diff* [3] simply uses the value of last day or last week separately to predict the current one. *Holt-Winters* [21] calculates forecast values using three smoothing equations (level, trend and seasonal components) with three parameters ranged from 0 to 1. *Historical Average/Median* [18] calculates the average/median of the historical data within a window as a prediction. *TSD (Time Series Decomposition)* [19] extracts four components from a KPI: level, trend, seasonality, noise, and then makes predictions using the sum of the first three components. *TSD Median* is similar to TSD, but it uses median value instead of mean when calculating that three components. *Wavelet decomposition* [22] can cover the entire frequency domain of a KPI and we set the high-frequency part as predictions.

Most time series models have one or more parameters, and their parameters have to be tuned to fit a KPI the best. However, parameter tuning can be time-consuming and often requires domain knowledge. In our study, our ultimate goal is to determine flux-correlation. As long as a model can make sufficiently accurate forecast such that the obtained flux-features capture real fluctuation patterns, it is not necessary to strive for the best fitting parameters. Thus, in CoFlux, for each model parameter, we enumerate a list of possible values empirically. For example, the parameter of Wavelet can take several possible values. With each parameter configuration (*e.g.*, *win*=1 day), Wavelet model can produce one predicted KPI, and then one flux-feature. We view *a time series model with a specific parameter configuration as a flux-feature detector*. In total, We get 86 detectors from the models and parameter configurations shown in Table 1. Accordingly, for each KPI, they produce 86 flux-features.

Certainly, our flux-feature extraction module in CoFlux can be configured with other models or parameter values if needed. Note that we have carefully evaluated most widely-used time series models. Through our experiments, we found some of them, such as MA (Moving average) [20], WMA (weighted MA) and Exponentially WMA, did not perform well in extracting flux-features from our testing KPIs, because the prediction of these models mainly rely on recent data and cannot handle seasonal KPIs. Fig. 4 shows an example that MA mistakenly captures the seasonality in KPI's main patterns as flux-features, causing false positives.

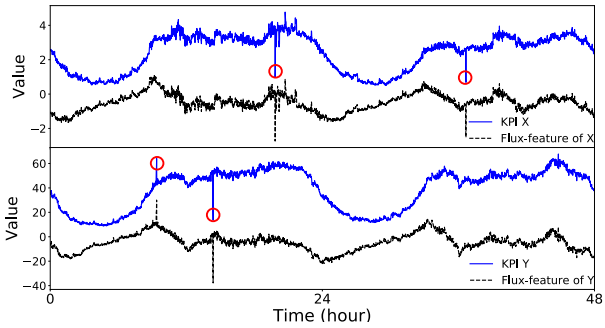


Figure 4: Two KPIs which are not flux-correlated and their flux-features extracted by Moving average; The circles mark the large fluctuations.

4.1.2 Feature amplification. After feature extraction, we get 86 raw flux-features for each KPI. The raw flux-features of different KPIs usually have values in different scales and units. We apply *z-score* to normalize each flux-feature. The closer to zero a value is in the normalized flux-feature, the less the KPI fluctuates.

In general, most values in a KPI look quite normal and they slightly deviate from forecast values mainly due to noises. To reduce the influence of noises, we use *modified exponential activation* (Eq. 1) to strengthen large fluctuations. Eq. 1 amplifies values that significantly deviate from zero, while having little impact on those close to zero. This amplification can make flux-feature more distinguishable and help flux-correlation identification. In CoFlux, we set $\alpha = 0.5$ (the degree of growth, the larger the value, the faster the growth) and $\beta = 10$ (if $|x| > \beta$, f value will not grow). The effectiveness of modified exponential activation will be evaluated by experiments and discussed later in Section 5.4.3.

$$f(\alpha, \beta, x) = \begin{cases} e^{min(x,\beta)\times\alpha} - 1, for\ x \geq 0 \\ -e^{min(|x|,\beta)\times\alpha} + 1, for\ x < 0 \end{cases} \quad (1)$$

4.2 Correlation measurement

Cross-correlation [17], a similarity measurement for time-lagged time series, has been widely used in signal and image processing. Cross-correlation can determine the shape similarity of two time series well with considering their distortions in amplitude and

Algorithm 1: Correlation measurement

Input: $afxSet$: Set of amplified flux-features of KPI X
$afySet$: Set of amplified flux-features of KPI Y
$coTHR$: Threshold of existence of flux-correlation

$resultSet \leftarrow [\,]$
// Set of candidate flux-correlation results
for *afx in afxSet* **do**
  for *afy in afySet* **do**
    $resultSet \leftarrow FCC(afx, afy)$ // **Eq. 4**
if *abs(max(resultSet[:,0])) > abs(min(resultSet[:,0]))* **then**
  [ccV, shiftV] = max(resultSet) /* ccV: correlation value about the existence of flux-correlation; shiftV: shifted value of X when get ccV */
else [ccV, shiftV] = min(resultSet);
if *abs(ccV) ≥ coTHR* **then**
  if *shiftV = 0* **then**
    if *ccV ≥ 0* **then** $X \overset{+}{\leftrightarrow} Y$;
    else $X \overset{-}{\leftrightarrow} Y$;
  if *shiftV < 0* **then**
    if *ccV ≥ 0* **then** $X \overset{+}{\rightarrow} Y$;
    else $X \overset{-}{\rightarrow} Y$;
  if *shiftV > 0* **then**
    if *ccV ≥ 0* **then** $Y \overset{+}{\rightarrow} X$;
    else $Y \overset{-}{\rightarrow} X$;
else $X \nsim Y$;

phase, which is suitable for judging the three questions in CoFlux. Thus, we choose Cross-correlation to measure the correlation between flux-features in CoFlux. For two amplified flux-features $G = [g_1, \cdots, g_l]$ and $H = [h_1, \cdots, h_l]$, considering the shift between G and H along the time axis, Cross-correlation always keeps one vector (*e.g.*, H) static and makes another one (*e.g.*, G) slide over H to calculate the inner product for each shift s of G (where $s \in (-l, l)$, l is the length of amplified flux-feature). G_s, the vector with shifted value s (especially $G_0 = G$), can be denoted as:

$$G_s = \begin{cases} [\overbrace{0, \cdots, 0}^{|s|}, g_1, \cdots, g_{l-s}], for\ s \geq 0 \\ [g_{1-s}, \cdots, g_l, \underbrace{0, \cdots, 0}_{|s|}], for\ s < 0 \end{cases} \tag{2}$$

The inner product of G_s and H, and their Cross-correlation value can be calculated as follows:

$$R(G_s, H) = \sum_{i=-l+1}^{l-1} G_s[i] \times H[i] \tag{3a}$$

$$CC(G_s, H) = \frac{R(G_s, H)}{\sqrt{R(G, G) \times R(H, H)}} \tag{3b}$$

Enumerating all possible values of s, we can obtain a Cross-correlation value vector of length $2l - 1$. Let the minimum and maximum of the vector be $minCC$ and $maxCC$, corresponding to shifted values $s1$ and $s2$ respectively. Then the final Cross-correlation value of G and H can be denoted as FCC:

$$minCC = \min_s \left(CC(G_s, H)\right), s1 = \arg\min_s \left(CC(G_s, H)\right) \tag{4a}$$

$$maxCC = \max_s \left(CC(G_s, H)\right), s2 = \arg\max_s \left(CC(G_s, H)\right) \tag{4b}$$

$$FCC(G, H) = \begin{cases} [minCC, s1], for \left|maxCC\right| < \left|minCC\right| \\ [maxCC, s2], for \left|maxCC\right| \geq \left|minCC\right| \end{cases} \tag{4c}$$

From Eq. 3, we know that $FCC \in [-1, 1]$. The closer to 1 or -1 FCC lies, the stronger the correlation between G and H. Moreover, positive FCC means they move in the same direction, while negative FCC indicates that one flux-feature increases while the other decreases and vice versa. The effectiveness of Cross-correlation can be seen in Section 5.4.4.

As shown in Algorithm 1, with amplified flux-features, we can then analyze flux-correlation for two KPIs easily. We enumerate all amplified flux-feature pairs to calculate their Cross-correlation values. Among these values, the one with the maximum absolute value is selected as the final score and compared with a threshold to determine the existence of flux-correlation between the KPIs. This threshold can be decided by practical requirements, or by precision-recall curves, as discussed in Section 5.4.1. If the absolute value of the final score is below the threshold, $X \nsim Y$; otherwise $X \backsim Y$. If $X \backsim Y$, we continue to find out the temporal order and direction. The temporal order can be determined by the shifted value of X. If it is zero (*i.e.*, no shift) then $X \leftrightarrow Y$; if it is negative (*i.e.*, X shift to left) then $X \rightarrow Y$; otherwise, $Y \rightarrow X$. This flux-correlation can be positive or negative, as indicated by the final score.

5 EVALUATION

In this section, we evaluate the performance of CoFlux using two real-world datasets.

5.1 Data sets

We collected two datasets, named Dataset I and II, from a top global Internet company. Each dataset contains a number of KPIs, and their flux-correlations have been labeled by domain experts to provide us the ground truth for evaluation.

Table 2: Dataset I: "Ground truth" column lists the number of intra-category flux-correlated KPI pairs.

Category	KPI ID, name, characteristics	# KPIs	Ground truth
1	1.1 Success rate of requests per cluster (stable)	21	33
	1.2 Count of HTTP error code per cluster (seasonal)	21	
2	2.1 Error count of request per data center (seasonal)	14	38
	2.2 Error rate (Error count/ Total count) per data center (stable)	14	
3	3.1 Java Garbage Collection overhead per machine (no clear pattern)	11	11
	3.2 JVM CPU usage per machine (no clear pattern)	11	
4	4.1 CPU Idle per machine (no clear pattern)	26	26
	4.2 Memory usage per machine (no clear pattern)	26	
5	5.1 Success rate of requests per product (stable)	16	20
	5.2 CPU Idle per product (seasonal)	16	

Dataset I: Flux-correlated KPIs with different time series characteristics. As shown in Table 2, Dataset I contains 5 categories and each category has two types of KPIs collected from different server clusters, machines or products. The two flux-correlated KPIs often have different time series characteristics (*e.g.*, seasonal, stable). For example, in Category 1, a fluctuation in "success rate of request" is often accompanied by a sudden change in "count of HTTP error code", *i.e.*, "success rate of request"$\overset{-}{\leftrightarrow}$"count of HTTP error code".

Dataset II: Flux-correlated KPIs with homogeneous time series characteristics. Dataset II contains KPIs that are collected from services used by 10 software modules, from which customers can request services from providers on demand. The details are shown in Table 3. All KPIs are heavily influenced by the count of customer requests, thus they are all seasonal. In each module, the services are called in a chain, and therefore the corresponding KPIs are flux-correlated with each other with temporal orders. For example, in Module 1, there are 9 of 20 KPIs are flux-correlated. When a fluctuation happens in "#customers request for providers", after a while, the fluctuation propagates to "#providers accept and manage the orders" and then to "#customers pay for the orders", *etc.*

Table 3: Dataset II: KPIs collected in 10 software modules. All KPIs are seasonal. "Ground truth" column lists the number of intra-module flux-correlated KPI pairs.

Module ID	Module Name	# KPIs	Ground truth
1	*T* services	20	36
2	Order creation services	26	66
3	*H* services	19	21
4	Payment actions services	15	6
5	Customer support services	36	171
6	Quality monitoring services	16	15
7	Provider support services	16	10
8	Location-based services	18	21
9	H5 support services	22	45
10	Public platform services	18	21

Note that only intra-category and intra-module flux-correlations are measured in our experiments. For example, in Dataset II, pairwise flux-correlations are done among 20 KPIs within Module 1.

5.2 Baseline algorithms

There exist no special methods designed to calculate the flux-correlations of KPIs. In order to show the effectiveness of CoFlux, we carefully choose seven baseline algorithms.

5.2.1 J-measure. a widely used event correlation method [2] which calculates the correlation as the average information content of a probabilistic classification rule. J-measure is designed for event correlation and cannot be used here directly. Thus, we need to detect anomalies first. Recall that flux-features are forecast errors. We define anomalies as the values that differ from the mean by more than 2 standard deviations [18] in the flux-features, and the selection of flux-features will be described in a short while. As a result, a KPI is transformed to a binary sequence. For two binary variables X and Y, if the correlation of J-Measure is larger than some threshold, then $X \backsim Y$. Otherwise, $X \approx Y$. Neither the temporal order nor direction of correlation can be determined by this approach.

For fair comparison, for the anomaly detection in J-measure, we enumerated all detectors in Table 1 and selected one detector for each dataset which can provide the best F1-score (see Section 5.3). The same method has been used to choose the best performing detectors for the other two algorithms, Pearson② and Granger②, too.

5.2.2 SIG [9]. [9] computes the anomaly signal of time series using KL divergence and then get the correlation results using Cross-correlation. However, its anomaly signal is relative entropy and non-negative so SIG cannot judge the direction of correlation.

5.2.3 Pearson Correlation. a popular technique for investigating the correlation between two variables, say X and Y, [2]. The threshold is TH, If $\rho > TH$, then X and Y are correlated with positive direction; and if $\rho < -1 * TH$, two KPIs are correlated and the direction is negative. If not, $X \approx Y$. This method does not consider the temporal order of correlation. Here we calculate Pearson Correlation for each pair of raw KPIs (denoted as Pearson①), and for each pair of amplified flux-features (denoted as Pearson②), for fair comparison.

5.2.4 Granger causality. Granger causality [14] captures the linear interdependencies among time series using VAR (Vector autoregressive). Specifically, Y Granger-causes X if and only if at least one of the corresponding coefficients of Y is non-zero. If Y Granger-causes X and X Granger-causes Y, X and Y occur simultaneously. Note that, Granger causality cannot indicate whether the correlation is positive or negative. Similar to Pearson Correlation, we calculate Granger causality of raw KPIs (denoted as Granger①) and Granger causality of flux-features (denoted as Granger②), for fair comparison.

5.2.5 Cross-correlation. We apply Cross-correlation, which is also part of our CoFlux, to calculate the correlations between two KPIs using its raw values instead of the flux-features.

5.3 Evaluation Metric

Table 4: Descriptions for FP and FN about three questions of flux-correlation.

	FP/FN	Ground Truth	Output
Existence	FP	$X \approx Y$	$X \backsim Y$
	FN	$X \backsim Y$	$X \approx Y$
Temporal order	FP	$X \rightarrow Y$ or $Y \rightarrow X$	$X \leftrightarrow Y$
	FN	$Y \rightarrow X$ or $X \leftrightarrow Y$	$X \rightarrow Y$
Direction	FP	negative	positive
	FN	positive	negative

We use F1-Score to evaluate the effectiveness of CoFlux and baseline algorithms, which is defined as: F1-Score = $\frac{2 \times Precision \times Recall}{Precision + Recall}$, where Precision = $\frac{TP}{TP+FP}$, Recall = $\frac{TP}{TP+FN}$. Table 4 describes the FP (false positive) and FN (false negative) about the three questions of flux-correlation.

Because the thresholds for J-measure, SIG, Pearson Correlation, Cross-correlation and CoFlux are usually determined based on actual requirements for precision and recall, we apply PRC (Precision Recall Curve) to show their performance by varying the threshold value from 0 to 1. A PRC close to the upper right has better performance than one close to the bottom left. Moreover, the best F1-Score can be selected using PRC.

5.4 Result and Observations

5.4.1 CoFlux vs baseline algorithms. We compare CoFlux with seven algorithms using the two datasets described before.

Table 5 lists the best F1-Scores of eight algorithms according to PRC. CoFlux outperforms the other seven algorithms significantly for both two datasets. The overall best F1-Scores of CoFlux for the two datasets are over 0.84, 0.92, 0.95 for Q1~3, respectively. Moreover, PRCs shown in Fig. 5 indicates CoFlux has overall better performance in terms of all precision and recall combinations.

Table 5: Best F1-Scores of eight algorithms; for J-measure, the best performing detector for Dataset I is Wavelet(win=1 day), for Dataset II is TSD(win=2 weeks); for Pearson②, the best performing detector for Dataset I is Wavelet(win=1 day), for Dataset II is diff(win=1 day); for Granger②, the best performing detector for Dataset I is Wavelet(win=1 day), for Dataset II is TSD(win=2 weeks). ('N/A' denotes no results).

Data set	Algorithms	Best F1-Score		
		Existence	Temporal order	Direction
I	CoFlux	**0.8412**	**0.9608**	**0.9579**
	J-measure	**0.7213**	N/A	N/A
	SIG	0.5381	**1.0**	N/A
	Pearson①	0.3106	N/A	0.6127
	Pearson②	0.5909	N/A	0.6945
	Granger①	0.2864	0.9009	N/A
	Granger②	0.4128	0.8952	N/A
	Cross-correlation	0.3613	0.9320	**0.9814**
II	CoFlux	**0.9026**	**0.9206**	0.9987
	J-measure	**0.8462**	N/A	N/A
	SIG	0.7706	0.8012	N/A
	Pearson①	0.7193	N/A	0.9845
	Pearson②	0.7828	N/A	**1.0**
	Granger①	0.4533	0.9025	N/A
	Granger②	0.6732	**0.9141**	N/A
	Cross-correlation	0.7494	0.7781	**1.0**

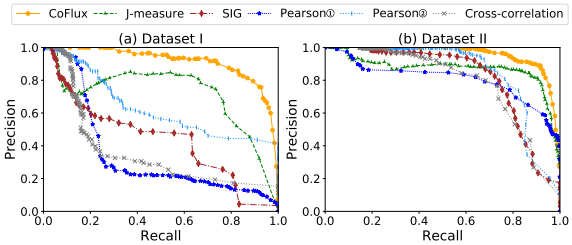


Figure 5: PRCs about the existence of flux-correlation among CoFlux and five baseline algorithms. Granger causality is not shown here because it has no thresholds thus no PRC.

Another observation is that these algorithms all perform better in Dataset II than in Dataset I, because the KPIs in Dataset II are homogeneous in nature (they are all part of service call graph, heavily influenced by the patterns of customer requests) and fluctuations follow the same pattern, while those in Dataset I vary by their time series characteristics. As a result, it is relatively easier to extract fluctuations from Dataset II. Next, we analyze each algorithm's performance in answering Q1~3.

With respect to Q1, our CoFlux excels the others in many aspects. Particularly, its performance in Dataset I only drops slightly compared to Dataset II, while other algorithms have more significant changes. This indicates CoFlux is very robust and can handle heterogeneous KPIs due to its rich detector library. The precision and recall with best F1-score shown in Fig. 6 provides more details about the performance difference among these algorithms. Particularly, the high precision of CoFlux indicates a small false positive rate in identifying flux-correlation. This demonstrates that it is appropriate to determine the final flux-correlation by the *maximum* absolute Cross-correlation value, as described in Section 3.

J-measure achieves the second best performance in our experiments. The best performing detector by F1-Score for Dataset I is Wavelet(win=1 day). This detector decomposes a KPI into a set of hierarchical structure series [22]. However, some KPIs have no strong structural characteristics and are hard to be predicted. Thus J-measure performs moderately. For Dataset II, TSD(win=2 weeks) is the best detector and it fits well with the seasonal structure of KPIs in this dataset. For SIG, its anomaly signal is computed by KL divergence, so it works well with the outliers and is not suitable for the anomalies that are mainly judged by partial history data. As a result, SIG does not perform well on Dataset I, because the various types of KPIs make its anomalies also complex.

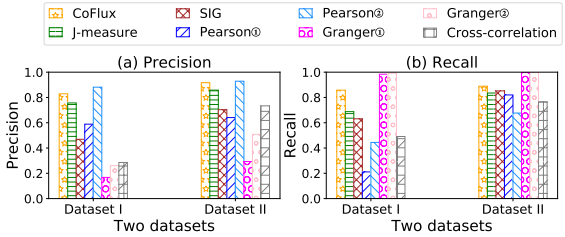


Figure 6: Precision and recall with best F1-Score about the existence of flux-correlation among eight algorithms.

For Pearson①, the recall of Dataset I drops dramatically compared to Dataset II, because two flux-correlated KPIs in different shapes often have weak linear association. Pearson② performs better than Pearson① because it uses flux-features which remove most inherent characteristics of KPI. Granger① tries to predict one KPI using another one, and its prediction performance mostly depends on the raw KPI instead of the fluctuation part, thus it performs not well. Granger② works better than Granger①, but it cannot achieve satisfactory performance either, because in our scenario, flux-features are usually created by unexpected accidents and it is not promising to predict them using regression. Cross-correlation with raw KPIs does not work well for flux-correlation, because flux-correlation mainly cares about the fluctuations instead of raw KPIs. It performs poorer on Dataset I than Dataset II because the KPIs of Dataset I are with different characteristics, which also demonstrates the effectiveness of flux-features.

Now we turn to Q2. Note that, only after an algorithm correctly recognizes Q1 of flux-correlation for a KPI pair, the test on Q2 and Q3 will continue for this pair. Although Granger causality has about 0.90 F1-Score in recognizing the temporal order, it has difficulties in identifying Q1 of flux-correlation. As a result, its overall performance is unsatisfactory. The temporal order of flux-correlation cannot be calculated by Pearson Correlation and J-measure. On the other hand, CoFlux and SIG perform very well in identifying the temporal order due to the use of Cross-correlation.

Finally, regarding Q3, Pearson Correlation can indicate positive or negative flux-correlation. The direction cannot be obtained by SIG, J-measure and Granger causality. Again, CoFlux has excellent performance in this regard.

Summary. CoFlux performs better than J-measure, an algorithm based on anomaly detection, because CoFlux does not need to define anomalies and the flux-features have more information than the binary results of anomaly detection. For SIG, it computes anomaly signal of KPIs using KL divergence, thus it focuses on the outliers

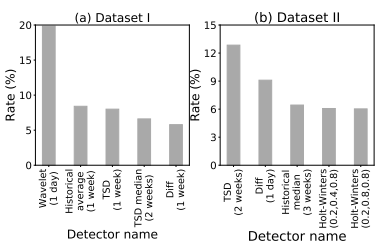


Figure 7: Top 5 detectors which give the flux-correlation results.

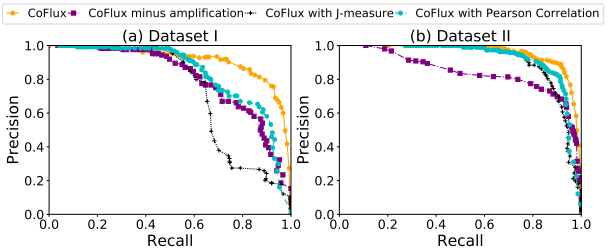


Figure 8: PRCs about the existence of flux-correlation among CoFlux and its variants.

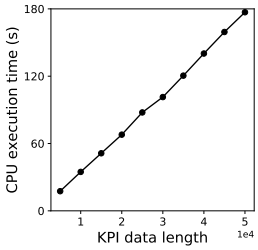


Figure 9: Efficiency by varying data length.

and cannot identify anomalies that are mainly judged by *partial* history data. Compared with SIG, we includes a robust set of flux-features, so CoFlux is robust against diverse anomalies and KPI characteristics. CoFlux performs better than Pearson①, Granger① and Cross-correlation, because raw KPIs can not reflect the flux-correlation well. Unlike Pearson② and Granger②, CoFlux uses Cross-correlation which can work well with the shape similarity of flux-features in consideration of the distortions in amplitude and phase. Moreover, in practice, for J-measure, Pearson② and Granger②, the anomaly detector selection for different KPIs is also very challenging, while the detector selection for different KPIs in CoFlux is determined automatically.

5.4.2 Analysis of detectors in CoFlux. Since CoFlux uses a number of time series models with different parameters as its flux-feature detectors, a natural question is whether so many detectors are needed and what detectors are the most useful. We analyzed the detectors that are used to produce the final flux-correlation for all the KPIs pairs in each dataset. Fig. 7 shows Top 5 detectors used in the two datasets. Wavelet is the most popular one in Dataset I, but TSD takes the lead in Dataset II. This can be explained by the fact that many KPIs in Dataset I have no fixed structural characteristics, and Wavelet fits better with this kind of KPIs. Most KPIs in Dataset II present seasonal characteristics, which can be handled by TSD very well. In either case, the proportion of Top 1 detector is no more than 20%, and the total proportion of Top 5 detectors in each dataset is less than 50%. Therefore, we can conclude that it is necessary to include a variety of detectors in Table 1, to boost the robustness of CoFlux.

5.4.3 The effectiveness of feature amplification in CoFlux. In this section, we evaluate whether feature amplification can indeed improve the performance of CoFlux. Fig. 8 shows the PRCs obtained by two versions of CoFlux: with and without amplification. Without amplification, the performance of CoFlux degrades significantly. Feature amplification can boost the performance, because it makes large fluctuation larger and then reduces the impact of noises on flux-features.

5.4.4 The effectiveness of Cross-correlation in CoFlux. In this section, we evaluate the benefits of Cross-correlation in terms of measuring the existence of flux-correlation in CoFlux. Because Granger causality works worse than the other algorithms, we use Pearson Correlation and J-measure to replace Cross-correlation separately in CoFlux, denoted as CoFlux with Pearson Correlation and CoFlux with J-measure. Fig. 8 shows their PRCs, which indicates that CoFlux achieves a much better performance with Cross-correlation. The main reason is that both Pearson Correlation and J-measure cannot appropriately handle fluctuation similarity and the situation where there is a delay between the fluctuations in two KPIs. We can also learn that Pearson Correlation performs better than J-measure because of the difficulty of anomaly definition for J-measure. Therefore it is a good choice to apply the Cross-correlation for the flux-correlation measurement.

5.4.5 The efficiency study of CoFlux. The experiments about the efficiency of CoFlux were run on a server with 24-core Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz with 64GB RAM. The flux-features of one KPI are calculated only once and used repeatedly in the Cross-correlation measurement with other KPIs. Execution time of Dataset I and II are shown in Table 6 and 7. Data length is the number of data points in a KPI. The execution time of all KPI categories and modules are less than 25 minutes. Certainly, longer or more KPIs can increase execution time. Also, execution time can be significantly reduced by using more computation resources.

Table 6: Execution time (seconds) on Dataset I.

Category	# KPI	Data length	Time
1	42	6720	1014
2	28	3024	462
3	22	5040	304
4	52	5040	1478
5	32	5040	584

Table 7: Execution time (seconds) on Dataset II (Length: 6720).

Module	# KPI	Time	Module	# KPI	Time
1	20	174	6	16	176
2	26	334	7	16	181
3	19	242	8	18	234
4	15	153	9	22	314
5	36	770	10	18	232

Furthermore, we test the impact of KPI length on execution time. In order to obtain KPIs in different lengths, we use two synthetic KPIs and vary their lengths. As shown in Fig. 9, the execution time increases with the KPI length almost in a linear fashion. It ranges from 15 to 180 seconds as the data length increases from 5k to 50k, while it is less than 20 seconds for baseline algorithms. Considering CoFlux is offline, the time is acceptable. The time of feature engineering is sensitive to data length, and the computational cost is $O(n)$, where n is the KPI length. When the number of detectors is fixed, the time complexity of Cross-correlation is affected by data length. We use Fast Fourier Transform to accelerate Cross-correlation and its computation complexity is $O(nlog(n))$.

6 APPLICATIONS OF COFLUX

In this section, we demonstrate how CoFlux can be used to assist operators in assisting service troubleshooting in a top online shopping service company.

6.1 Clustering KPIs for alert compression

We evaluated CoFlux's effectiveness in alert compression on 118 KPIs provided by the company. Operators manually analyzed the 699 alerts generated during a month by anomaly detection on individual KPIs. They concluded that only 184 alerts should be raised (served as the ground truth in our evaluation), and all other alerts are correlated with the ones in the ground truth.

To demonstrate how CoFlux can be used to compress alerts [8] and reduce operators' workload, we group these KPIs into 23 clusters by K-Means based on the absolute flux-Correlation values of KPIs. K is determined by silhouette coefficient [23] which has shown good performance in experiments, and the largest silhouette coefficient over different clustering number indicates the best K for clustering. Fig. 10 shows 3 of the clusters composed of 24 KPIs. Apparently, KPI pairs within a cluster have stronger flux-Correlation than those outside it.

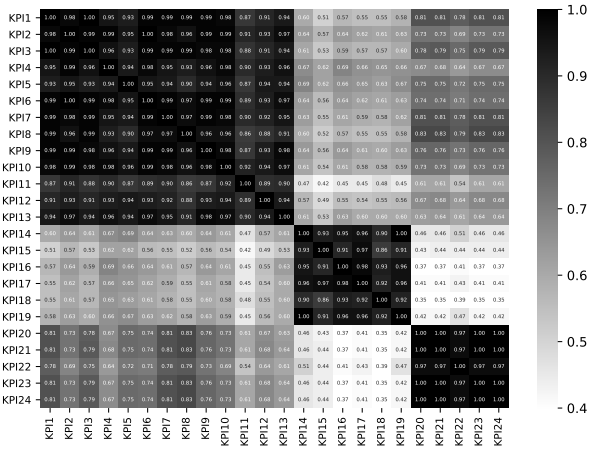


Figure 10: Heat map visualization for clustering results of 24 KPIs. The value at (x, y) means the absolute flux-correlation value between KPI x and y. Cluster1: KPI1 ~ KPI13; Cluster2: KPI14 ~ KPI19; Cluster3: KPI20 ~ KPI24.

Then we can monitor all KPIs within a cluster as a whole, and raise only one alert when one or more anomalies occurred at these KPIs, resulting in only 208 alerts, achieving 70.24% (1-208/699) compression ratio. We calculate compression accuracy by comparing the compressed alerts with the alerts in the ground truth, and CoFlux obtained 0.9748 compression F1-Score. For comparison, we conduct another experiment where raw KPIs were clustered based on their Cross-correlation, and achieved 70.67% compression ratio and 0.8322 compression F1-Score. In practice, it is critical for alert clustering to maintain a high compression F1-Score (*i.e.*, close to 1). Although these two experiments gave similar compression ratios, clustering by CoFlux leads to far fewer compression errors than by Cross-correlation of raw KPIs, demonstrating its excellent capability in alert compression.

6.2 Recommending Top N flux-correlated KPIs

Fig. 11 shows a given KPI, say X, and its Top 5 flux-correlated KPIs out of the remaining 117 KPIs ranked by flux-correlation. Clearly, these KPIs are highly flux-correlated with KPI X. Thus, when KPI X suffers from an anomaly, operators can narrow down their analysis scope to the Top N flux-correlated KPIs for more efficient diagnosis.

Motivated by the idea of HitRate@K for recommender systems [24], we calculate the recommendation accuracy using the KPIs in Section 6.1, and the result is 0.8051 for HitRate@5. In comparison, recommendation based on Cross-correlation of raw KPIs achieved only 0.5594 for HitRate@5.

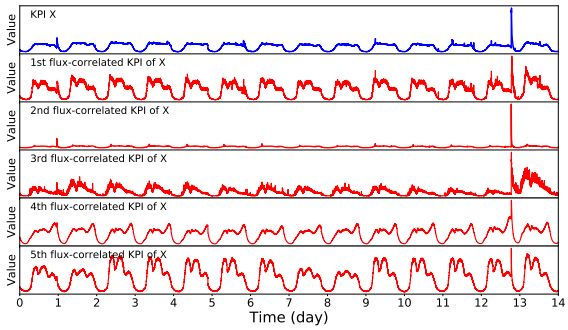
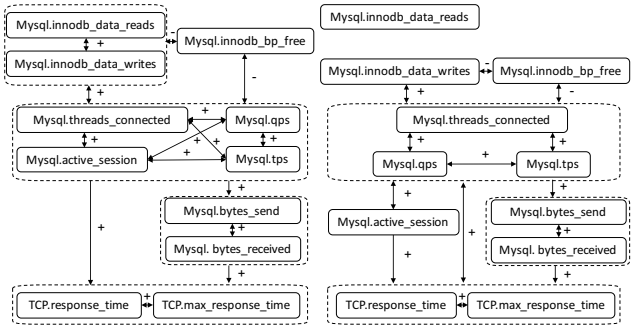


Figure 11: Top 5 flux-correlated KPIs for a given KPI X.

6.3 Constructing fluctuation propagation chains



(a) Fluctuation propagation chain of a database service manually constructed by the operators.

(b) Fluctuation propagation chain of database service generated by CoFlux with threshold = 0.65.

Figure 12: Fluctuation propagation chains of a database service constructed by the operators and CoFlux. → and ↔ denote the temporal order of flux-correlation, + and - denote the direction.

Understanding how fluctuations are propagated through KPIs is extremely useful in root cause analysis, but this requires deep domain knowledge of the KPIs. Intuitively, fluctuation propagation chains can be constructed based on flux-correlation. Fig. 12 shows such chains of a database service in the Internet company, where each node denotes a KPI and each edge denotes the flux-correlation between KPIs. To make the figures less crowded, an edge ended at a dashed box means the originating KPI of this edge is flux-correlated with every KPI inside this box. Fig. 12(a) presents the fluctuation propagation chain manually constructed by operators based on their deep domain knowledge, which serves as the ground truth in our comparison. When there happens a fluctuation for SQL Execution in a storage engine, it first propagates to the connection/thread module and then to the network interface card which provides the hosting service for database. Fig. 12(b) shows the fluctuation propagation chain generated by CoFlux. Apparently,

Fig. 12(b) can almost replicate Fig. 12(a), with high F1-scores, 0.8684, 0.9524, 1.0 for Q1~3 respectively. Thus, operators can apply CoFlux to automatically construct fluctuation propagation chains even without deep domain knowledge. In comparison, Cross-correlation of raw KPIs achieves inferior F1-scores: 0.5455, 0.9412, 1.0 for Q1~3 respectively.

7 RELATED WORK

Over the years, a substantial body of research has been conducted to analyze different types of correlations which can be divided into three categories: correlation between KPIs [14, 17, 25], correlation between events [6], correlation between event and KPI [2]. Those correlating algorithms mainly analyze the correlation of raw KPIs, or turn the KPIs into events and analyze the correlation using events. According to the analysis in Section 5.4.1, these algorithms cannot accurately analyze flux-correlation.

Besides our baseline algorithms, there are other generic methods that aim to find relationships among KPIs. CoIntegration [13] transforms a collection of non-stationary KPIs to be stationary by a linear combination. However, flux-correlated KPIs may have similar fluctuations but different seasonal and trend components. As a result, they cannot be made stationary easily through CoIntegration. VARMA [12] is a model for producing linear forecasts of many KPI variables, and provides an intuitive explanation for the correlations of KPIs. However, through the experiments with Granger causality in Section 5.4.1, we found that it is difficult to model the correlation between flux-features through regression. [10] uses ARCH models to predict the volatility of stock market and learn their correlations, but volatility, measured by the standard deviation of a time series, is conceptually different from fluctuations in our context.

In summary, all these methods cannot work well in identifying flux-correlation. Our work differs from them in that we extract features capturing fluctuations by time series models, and focus on flux-features instead of raw KPIs.

8 CONCLUSION

Correlating KPIs by fluctuations is extremely useful for service troubleshooting in many respects such as alert compression, recommending Top N flux-correlated KPIs and constructing fluctuation propagation chains. However, such a KPI flux-correlation has been little studied in the domain of Internet service operations management. To the best of our knowledge, this paper is the first attempt to formulate flux-correlation and study it in detail. For the large number and diverse characteristics of KPIs, with CoFlux, we can measure their flux-correlations automatically and robustly. Our experiments have demonstrated that CoFlux significantly outperforms baseline algorithms by a large margin, and the applications of CoFlux in a top global Internet company also demonstrated its effectiveness in assisting service troubleshooting.

As our future work, we will explore deep learning as flux-correlation algorithm and possible models to fit flux-features to further understand the nature of flux-correlation.

9 ACKNOWLEDGMENT

We thank Weibin Meng, Ping Liu, Nengwen Zhao, Shenglin Zhang, Yongqian Sun and Yu Sun for their helpful discussions on this work. We thank Juexing Liao for proofreading this paper. This work has been supported by the Beijing National Research Center for Information Science and Technology (BNRist) key projects, and the Okawa Research Grant.

REFERENCES

[1] Mike P Papazoglou and Willem-Jan Van Den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, 16(3):389–415, 2007.
[2] Chen Luo, Jian-Guang Lou, Qingwei Lin, Qiang Fu, Rui Ding, Dongmei Zhang, and Zhe Wang. Correlating events with time series for incident diagnosis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1583–1592. ACM, 2014.
[3] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. Opprentice: Towards practical and automatic anomaly detection through machine learning. In *Proceedings of the 2015 Internet Measurement Conference*, pages 211–224. ACM, 2015.
[4] Shenglin Zhang, Ying Liu, Dan Pei, Yu Chen, Xianping Qu, Shimin Tao, Zhi Zang, Xiaowei Jing, and Mei Feng. Funnel: Assessing software changes in web-based services. *IEEE Transactions on Service Computing*, 2016.
[5] Yongqian Sun, Youjian Zhao, Ya Su, Dapeng Liu, Xiaohui Nie, Yuan Meng, Shiwen Cheng, Dan Pei, Shenglin Zhang, Xianping Qu, and Xuanyou Guo. Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes. *IEEE Access*, 6:10909–10923, 2018.
[6] Xiaohui Nie, Youjian Zhao, Kaixin Sui, Dan Pei, Yu Chen, and Xianping Qu. Mining causality graph for automatic web-based service diagnosis. In *Performance Computing and Communications Conference (IPCCC), 2016 IEEE 35th International*, pages 1–8. IEEE, 2016.
[7] Jacob Cohen. Statistical power analysis for the behavioural sciences, 1988.
[8] Jingmin Xu, Yuan Wang, Pengfei Chen, and Ping Wang. Lightweight and adaptive service api performance monitoring in highly dynamic cloud environment. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 35–43. IEEE, 2017.
[9] Adam J Oliner, Ashutosh V Kulkarni, and Alex Aiken. Using correlated surprise to infer shared influence. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 191–200. IEEE, 2010.
[10] Yasushi Hamao, Ronald W Masulis, and Victor Ng. Correlations in price changes and volatility across international stock markets. *The review of financial studies*, 3(2):281–307, 1990.
[11] Maurizio Filippone and Guido Sanguinetti. Information theoretic novelty detection. *Pattern Recognition*, 43(3):805–814, 2010.
[12] Helmut Lütkepohl. Forecasting with varma models. *Handbook of economic forecasting*, 1:287–325, 2006.
[13] Richard ID Harris. Using cointegration analysis in econometric modelling. 1995.
[14] Huida Qiu, Yan Liu, Niranjan A Subrahmanya, and Weichang Li. Granger causality for time-series anomaly detection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1074–1079. IEEE, 2012.
[15] Jianqing Fan and Qiwei Yao. *Nonlinear time series: nonparametric and parametric methods*. Springer Science & Business Media, 2008.
[16] Shashank Shanbhag and Tilman Wolf. Accurate anomaly detection through parallelism. *IEEE network*, 23(1):22–28, 2009.
[17] Yasushi Sakurai, Spiros Papadimitriou, and Christos Faloutsos. Braid: Stream mining through group lag correlations. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 599–610. ACM, 2005.
[18] Suk-Bok Lee, Dan Pei, MohammadTaghi Hajiaghayi, Ioannis Pefkianakis, Songwu Lu, He Yan, Zihui Ge, Jennifer Yates, and Mario Kosseifi. Threshold compression for 3g scalable monitoring. In *INFOCOM, 2012 Proceedings IEEE*, pages 1350–1358. IEEE, 2012.
[19] Yingying Chen, Ratul Mahajan, Baskar Sridharan, and Zhi-Li Zhang. A provider-side view of web search response time. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 243–254. ACM, 2013.
[20] David R Choffnes, Fabián E Bustamante, and Zihui Ge. Crowdsourcing service-level network event monitoring. *ACM SIGCOMM Computer Communication Review*, 41(4):387–398, 2011.
[21] He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Dan Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. Argus: End-to-end service anomaly detection and localization from an isp's point of view. In *INFOCOM, 2012 Proceedings IEEE*, pages 2756–2760. IEEE, 2012.
[22] Keyi Zhang, Ramazan Gençay, and M Ege Yazgan. Application of wavelet decomposition in time-series forecasting. *Economics Letters*, 158:41–46, 2017.
[23] Katherine S Pollard and Mark J Van Der Laan. A method to identify significant clusters in gene expression data. 2002.
[24] Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. On top-k recommendation using social networks. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 67–74. ACM, 2012.
[25] William A Gardner, Antonio Napolitano, and Luigi Paura. Cyclostationarity: Half a century of research. *Signal processing*, 86(4):639–697, 2006.