

# FC-ADL: Efficient Microservice Anomaly Detection and Localisation Through Functional Connectivity

Giles Winchester  
 G.Winchester@sussex.ac.uk  
 University of Sussex  
 Brighton, UK

George Parisis  
 G.Parisis@sussex.ac.uk  
 University of Sussex  
 Brighton, UK

Luc Berthouze  
 L.Berthouze@sussex.ac.uk  
 University of Sussex  
 Brighton, UK

## Abstract

Microservices have transformed software architecture through the creation of modular and independent services. However, they introduce operational complexities in service integration and system management that makes swift and accurate anomaly detection and localisation challenging. Despite the complex, dynamic, and interconnected nature of microservice architectures, prior works that investigate metrics for anomaly detection rarely include explicit information about time-varying interdependencies. And whilst prior works on fault localisation typically do incorporate information about dependencies between microservices, they scale poorly to real world large-scale deployments due to their reliance on computationally expensive causal inference. To address these challenges we propose *FC-ADL*, an end-to-end scalable approach for detecting and localising anomalous changes from microservice metrics based on the neuroscientific concept of *functional connectivity*. We show that by efficiently characterising time-varying changes in dependencies between microservice metrics we can both detect anomalies and provide root cause candidates without incurring the significant overheads of causal and multivariate approaches. We demonstrate that our approach can achieve top detection and localisation performance across a wide degree of different fault scenarios when compared to state-of-the-art approaches. Furthermore, we illustrate the scalability of our approach by applying it to Alibaba's extremely large real-world microservice deployment.

## CCS Concepts

- Software and its engineering → Software reliability; Software maintenance tools.

## Keywords

Functional Connectivity, Anomaly Detection, Root Cause Analysis, Microservices

## 1 Introduction

Microservices (MSs) have fast become a cornerstone of contemporary software architecture. Under this paradigm, an application is built as a suite of small, independent services that communicate with each other over well-defined APIs to provide application functionality. Owing to distinct benefits such as developmental agility and fine-grained scaling, MSs are becoming the leading method for orchestrating online services and have already been integrated into many large service providers such as Netflix, eBay, Uber, and Alibaba.

While MS systems outclass traditional monolithic approaches in many aspects, they can also introduce significant operational challenges [10, 15], particularly in regards to meeting acceptable

up-time guarantees. With downtimes associated with high costs [7], accurate anomaly detection and localisation is paramount. However, as the systems grow, it becomes increasingly difficult to track the full set of inter-dependencies between components, particularly, as, by design, these relationships are constantly in flux during runtime operation, through many factors such as changing user behaviour, service orchestration, load balancing, and horizontal scaling.

To address these challenges, a number of promising anomaly detection and localisation approaches, with a specific interest on faults, have been developed that seek to leverage the vast amounts of operational data collected from microservice architectures (MSAs) [31]. These approaches can be broadly categorised based on the primary observability data that they leverage into trace-based and metric-based approaches. However, despite the promising performance of many of these approaches, they face significant limitations in practice. Trace-based methods rely on comprehensive MSA instrumentation and primarily detect anomalies that manifest in direct invocation paths, not considering indirect methods for fault propagation such as resource contention [35]. Metric-based approaches, whilst less intrusive, often treat metrics from MSs independently, overlooking the contribution of complex inter-MS interactions or rely on precise anomaly detection times which are often impractical to obtain in a real-world system [44]. Recent work has begun to develop metric-based approaches that leverage the inter-dependent nature of MSAs such as through causal structural or multivariate modelling [9, 27, 28, 33–36, 44, 46, 60, 65, 67, 69], however, many of these approaches lack end-to-end anomaly detection and localisation or are computationally expensive, limiting their application to MSAs larger than smaller-scale research testbeds [49].

Motivated by the above challenges, in this work we introduce FC-ADL, Functional Connectivity-based Anomaly Detection and Localisation. Inspired by work leveraging the concepts of *functional connectivity* (FC) [41] or *functional dependencies* [18], we develop a framework to construct a time-varying graph representation of the statistical relationships between MSs at runtime. These time-varying FC dependencies are inferred directly from usage metrics using light-weight correlation measures without requiring distributed tracing data (3.1). By efficiently quantifying the structural distance between these graphs, we demonstrate that anomalies can be detected directly from changes in these FC-based dependencies (3.2, 3.3). By focusing on changes in MS relationships rather than individual metric values, FC-ADL is more robust to spurious detection from time-varying metric data. Furthermore, we demonstrate that FC-ADL can accurately locate the candidate root cause of anomalies without needing causality by identifying the MS component with the largest change in FC dependencies before and after a change-point. We validate FC-ADL by comparing

its fault detection and localisation performance against state-of-the-art baselines on a new MS fault testbed by integrating chaos engineering tools [5] and a time-varying workload generator [30] into the widely used DeathStarBench testbed [17] which allows for more realistic fault-based detection and localisation testing. We also demonstrate the scalability of FC-ADL by comparing its computational complexity to other detection and root cause analysis (RCA) approaches when applied to a large-scale MS system maintained by Alibaba [31]. An open-source implementation of FC-ADL, the developed fault testbed, and all code and data associated with evaluation will be released alongside this paper [1].

## 2 Related Work

### 2.1 Trace-Based Anomaly Detection

Trace-based detection techniques leverage distributed tracing data to infer inter-MS dependencies and detect anomalies, often through structure-similarity methods or machine learning models trained to represent the normal operational patterns of a system's invocation paths [8, 20, 29, 38, 39, 42, 43, 50, 61, 73]. These methods can identify anomalous propagation patterns through call graphs and can then be paired with graph-based ranking techniques [71] to localise root causes. However, distributed tracing data is generally limited to anomalies that manifest through changes in invocation chains or invocation slowdown, which, whilst providing valuable insights into how services interact and which dependencies are involved in an anomaly, is limited to the service or endpoint level. Additionally, anomalies within MSAs can manifest due to indirect interactions that are not limited to direct invocation-based propagation. For example an anomaly that propagates through co-located MSs on the same node due to resource contention may be misattributed by tracing-based approaches [35]. Furthermore, instrumenting and maintaining distributed tracing at scale is difficult and prone to incomplete data collection [52, 58] if possible at all due to ownership and privacy concerns [72]. What's more, tracing data can be associated with runtime overheads [14] and costs in processing and storing large volumes of data [31, 62].

### 2.2 Metric-Based Anomaly Detection

Metric-based approaches rely on component or microservice-level measurements such as resource usage rates, network traffic, and response latency. Unlike trace-based approaches, they can capture anomalies that occur due to indirect dependencies and often do not require the same extensive instrumentation. One set of statistical approaches for anomaly detection from metric data focuses on time-series outlier detection, based on the assumption that anomalies present as unusual metric values [24, 25, 27, 40, 47, 48, 59, 65, 67]. However, such approaches equally apply the same assumptions to all metrics, when in reality metrics can show very different behaviours due to the heterogeneity of MSAs. Another approach, involving either statistical [6, 26, 70] or ML techniques [19, 22, 64, 65, 68], is to model the normal behaviour of individual metrics during anomaly-free runs of the system and use deviations in individual metrics from this normal behaviour to infer anomalies. A key shortcoming of this approach is that as MSAs are highly inter-connected, focusing on metrics from MSs individually can lead to false inference. For example, two microservices might only

show sub-threshold changes in their usage metrics despite having stopped interacting due to a fault where the computational overhead of the functionality is low. Conversely, metrics might show spikes in activity due to increases in front-end demand or changing user behaviour without any change in the strength of dependencies between microservices.

### 2.3 Metric-Based Root Cause Analysis

Once an anomaly has been detected, the next step is localise what caused the anomaly through root-cause analysis (RCA). One set of more simple approaches rely on comparing the statistics of metrics before and after detection, ranking the most likely candidate root causes based on the largest deviations [24, 25, 27, 40, 65, 67]. However, such approaches do not consider the propagation of faults between MSs and confounding factors such as resource contention on a co-habited node. A popular set of approaches for RCA from MS metric data that consider these factors are causal graph-based analysis approaches [9, 27, 28, 33–36, 46, 60, 65, 67, 69]. The common principle behind these approaches is the construction of a causal graph where vertices represent individual MSs or MS metrics, and edges represent the causal relationship between these MSs or metrics. These graphs are commonly constructed using causal structural modelling methods such as PC and LiNGAM [9, 28, 33–36, 46, 60, 65, 67]. Motivated by the observation that anomalies within MSAs propagate from the original root cause to dependent metrics several different approaches are then taken to locate candidate root causes from these causal graphs, such as applying centrality graph measures [36, 65, 67], random walks [33–35, 60], or depth or breadth-first searches [9, 27, 46]. However, the time complexity of such causal modelling approaches can be prohibitive. For example, the PC algorithm is worst-case exponential on the number variables as each variable must be tested for conditional dependencies on all subsets of other variables [12, 57]. Variants of LiNGAM do not incur the same combinatorial overhead as the PC algorithm, they still often operate in polynomial time on the number of variables due to matrix operations and independent component analysis steps [53, 54]. Furthermore, many of these localisation approaches assume the accurate detection of anomaly onset, which can not be guaranteed in complex operating environments [44]. Moreover, many of these approaches only concern themselves with RCA and do not implement end-to-end anomaly detection and localisation. Recently, BARO [44] introduced an end-to-end approach that models time-varying changes in multivariate metric time series for end-to-end detection and localisation. However, to update run-length distributions, online determinant and inverse operations, that do not scale well to the number of variables, must be repeatedly carried out.

## 3 Framework

The design of FC-ADL is motivated by shortcomings of prior work identified above, namely:

- The use of tracing data which limits the scope of anomalies that can be identified and incurs significant costs for the operator;

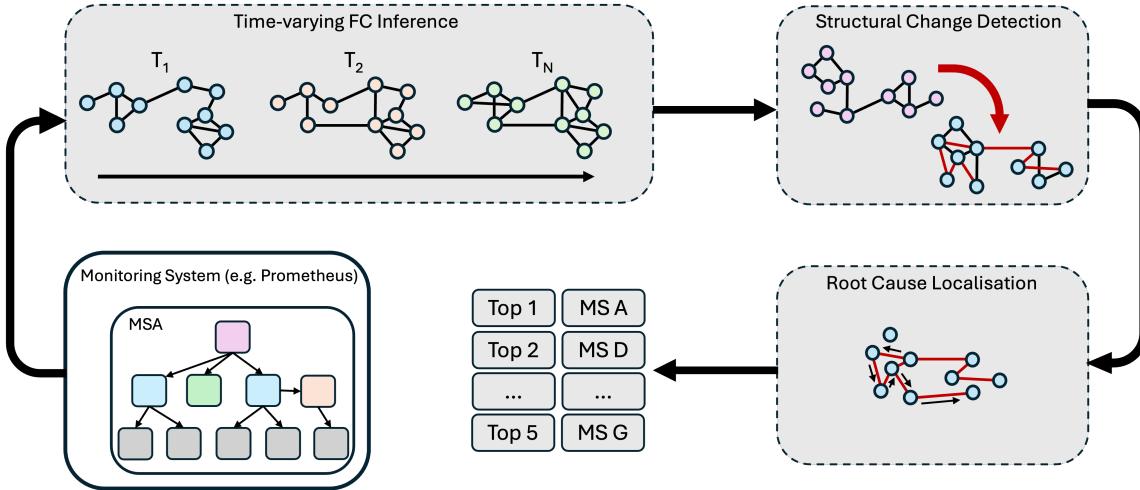


Figure 1: An overview of the FC-ADL framework

- The insufficient consideration of the time-varying properties of MS interdependencies that can lead to false inference on the presence of an anomaly;
- The lack of end-to-end frameworks for anomaly detection and localisation;
- The reliance of either detection or root-cause analysis on methods that typically do not scale to large real-world deployments;

Depicted in Figure 3, FC-ADL involves three main processes, time-varying FC inference and structural distance quantification from MS metrics, structural change-based anomaly detection, and FC-based localisation, which we detail below.

### 3.1 Inferring Time-Varying MS Functional Connectivity

In neuroscience, the concept of *functional connectivity* denotes the graph of statistical dependencies among neurophysiological activities, offering insights into functional relationships without requiring knowledge of structural (physical) connectivity. Besides neuroscience, this concept has been shown to be applicable to the analysis of a number of large scale distributed systems, including computer networks [41] and systems of systems [18], with further work showing how it can accelerate root-cause analysis [63]. In what follows, we describe how to extract time-varying FC from correlations between the fluctuations in the metric observability data collected from MSs.

**3.1.1 FC-Based Dependency Graph Construction.** The FC of a system is represented as a graph,  $G = (V, E)$ , where the set of vertices,  $V$ , and undirected edges,  $E$ , represent the components of the system and the strength of statistical dependency between their activities respectively. Within this work we leverage MS-level CPU usage metrics collected using Prometheus for two primary reasons. Firstly, CPU usage is a standard metric exposed by pods, this means that this metric can also be collected in the black box scenario where a cloud provider is monitoring customer workloads, whereas metrics

like request latency cannot due to privacy concerns and lack of ownership over exposed metrics [72]. Secondly, we argue that CPU usage better represents instantaneous MS activity as it directly reflects the processing of work in real time, whereas other standard metrics such as memory usage can remain allocated for extended periods regardless of whether a MS is actively handling requests. As we demonstrate in our results in Sections 4.6 and 4.7, CPU usage is generally sufficient to detect and localise various non-CPU related faults.

Constructing the FC from the CPU usage of MS systems is non-trivial, as FC-ADL must be able to update on the timescale of seconds even for very large scale MS deployments, necessitating scalability. Moreover, MS CPU usage data is non-stationary, demonstrating strong auto-correlation, seasonality, and trends. This non-stationarity can be handled by sophisticated time series models [3, 55], however, these approaches often have scalability issues. Therefore, in this work, we rely on Pearson's correlation, which is often used to infer FC in neuroscience [11], and CPU usage first-order differencing. This combined approach mitigates the effects of non-stationarity whilst leveraging the computational efficiency of calculating Pearson's correlations. To capture time-varying changes in MS dependencies, we generate consecutive temporal snapshots of FC by calculating Pearson's correlations over sliding windows. When each new CPU usage sample is collected from the system, the windows are shifted by one data point to generate a new FC graph. To mitigate the delay in detecting changes in dependencies, we use the exponentially weighted moving average (EWMA) Pearson correlation coefficient, which applies exponential smoothing to correlations calculated within the windows to give more weight to recent observations [45].

As measurements from MS metrics are often noisy in nature, a large number of Pearson correlations will be negligible but non-zero. This dense property can mask the true FC structure, and potentially obscure changes in it. Based on the assumption that small correlations are more likely to be associated with noise than true relationships between MSs we apply a nominal threshold in

all cases of 0.1. Thresholding or graph filtering are typical steps in constructing stable FC's within neuroscience [16].

### 3.2 FC Structural Distance

To track how the FC between MSs evolves over time we characterise the structural distance between subsequent FC graphs. Here, we follow the methodology described in [37] for detecting sequences of system states in temporal networks but implement DeltaCon as our distance measure rather than the eigenspectrum. We choose DeltaCon over the eigenspectrum due to its better sensitivity to local changes in network structure and improved robustness to noise, which is more suitable when dealing with MS-level faults in potentially noisy environments. In our approach each inferred sliding-window FC after thresholding is treated as a weighted adjacency matrix  $A_n$  of a graph  $G_n$ . To compute the DeltaCon distance we follow the steps outlined in [23] by computing the node affinity matrix

$$S_n = (I + \epsilon^2 D_n - \epsilon A_n)^{-1} \quad (1)$$

where  $\epsilon = 10^{-2}$ ,  $D_n = \text{diag}(A_n \mathbf{1})$  is the degree matrix, and  $I$  is the identity. We then measure the graph-distance between windows  $m$  and  $n$  via the root-Euclidean distance on these affinities

$$d_{m,n} = \sqrt{\sum_{i,j} (\sqrt{S_m^{ij}} - \sqrt{S_n^{ij}})^2} \quad (2)$$

The resulting distance matrix  $D$  with entries  $d_{m,n}$  captures the structural dissimilarity of FC graphs over time.

### 3.3 FC-Based Anomaly Detection

In our FC-ADL framework we propose that changes in dependencies between MS, inferred using metric-based FC, can directly be used to determine when an anomaly has occurred, rather than focusing on deviations of a single metric in isolation. To determine when a change point within the FC structure of MSs occurs we apply HDBSCAN [4] to cluster the DeltaCon distances between FC graphs into communities. HDBSCAN has two beneficial properties for our application, firstly it is density-based and therefore automatically determines the appropriate number of clusters  $k$ . This means that, unlike other approaches that have been used for MSA anomaly detection [25, 51], the approach does not require prior knowledge as to whether an anomaly has occurred. Secondly, once fit, HDBSCAN can provide a probability metric, based on membership strength, as to how well new inferred FC-graphs fit into existing clusters. Thirdly, HDBSCAN provides outlier detection which both improves robustness to noisy behaviour of the MSA, but also provides an approach, in conjunction with the probability metric, to provide online re-clustering in response to changing dependencies within the MSA.

### 3.4 FC-Based Localisation

The principal concept behind RCA within FC-ADL is that structural changes in FC dependencies can also indicate what caused the change. This assumption is rooted in prior RCA research, where

causality analysis is used to infer the directed relationships between metrics during an anomaly to identify the root cause [35, 66]. However, we propose instead that this same process can be efficiently carried out on undirected relationships inferred based on correlation. To carry out RCA in FC-ADL we first identify the FC dependencies that have changed between the prior normal state of the system and the current abnormal state. To identify these dependencies we average the collection of FC graphs associated with the normal and abnormal state of the system to produce the centroid graphs  $G_n$  and  $G_a$  that represents the average normal and abnormal FC dependency structure respectively. Next, to isolate the changes in FC that are only associated with the anomalous state of the system we subtract  $G_n$  from  $G_a$  to give us the dependency change graph  $G_c$ . We take the absolute value of  $G_c$  and apply the same nominal threshold of 0.1 to remove noisy low-strength edges. The resultant weighted graph  $G_c$  denotes which FC dependencies have changed between the two states of the system, and the strength of their change.

To provide a list of candidate root causes, we apply a random walk to  $G_c$  to quantify the contribution of each MS to changes in dependencies between the two states. Based on the propagating nature of anomalies within MSAs, we assume that the MS that has the most impact on the structural changes in FC dependencies is more likely to be the root cause of the anomaly. Unlike previous approaches [33, 35] our random walker implements only a single type of transition probability as the graph is undirected and only forward transitions are required. Unlike causal approaches [65], FC-ADL inherently captures the strength of edges between vertices, which is incorporated into the walker without further steps required for inferring weights for individual MS. For our random walker, the probability of walking from MS  $v_i$  to  $v_j$  is proportional to the strength of the edge  $e_{i,j}$ . We generate random walks from each node in  $G_c$  and then sum the visitation frequency of each vertex from all walks. The output of this process is a list of visitation frequency in descending order, with the top-K MS(s) in the list being provided as the most probable root cause(s).

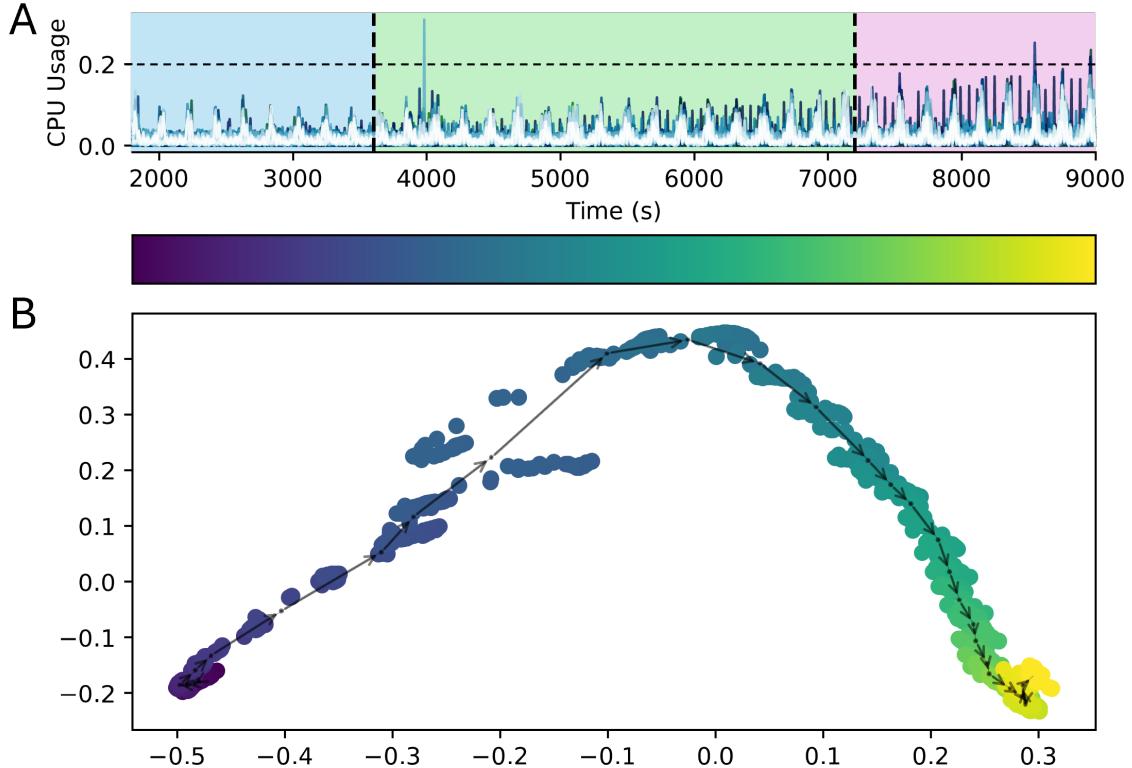
## 4 Implementation and Evaluation

### 4.1 FC-ADL Implementation

The components of FC-ADL have several free parameters, which, for this evaluation, are set as follows. The window size was set to 360 to capture the correlations between 30 minutes worth of CPU usage data, and the step size was set to 1, producing a FC graph for every newly sampled data point. For clustering, we used *scikit-learn*'s HDBSCAN implementation with the minimum cluster size set to  $|D|/5$  and all other parameters set to default. For root cause localisation we apply the random walker to generate 10 walks from each vertex in the graph, with a maximum walk-length of 10. Full implementation details can be found in the open-source implementation of FC-ADL released with this paper [1].

### 4.2 Experimental Datasets

**4.2.1 DeathStarBench.** is an open-source benchmark suite for MS deployments [17]. In this work, we use the social network deployment that provides an online social media application in which users can follow specific users, compose social media posts and

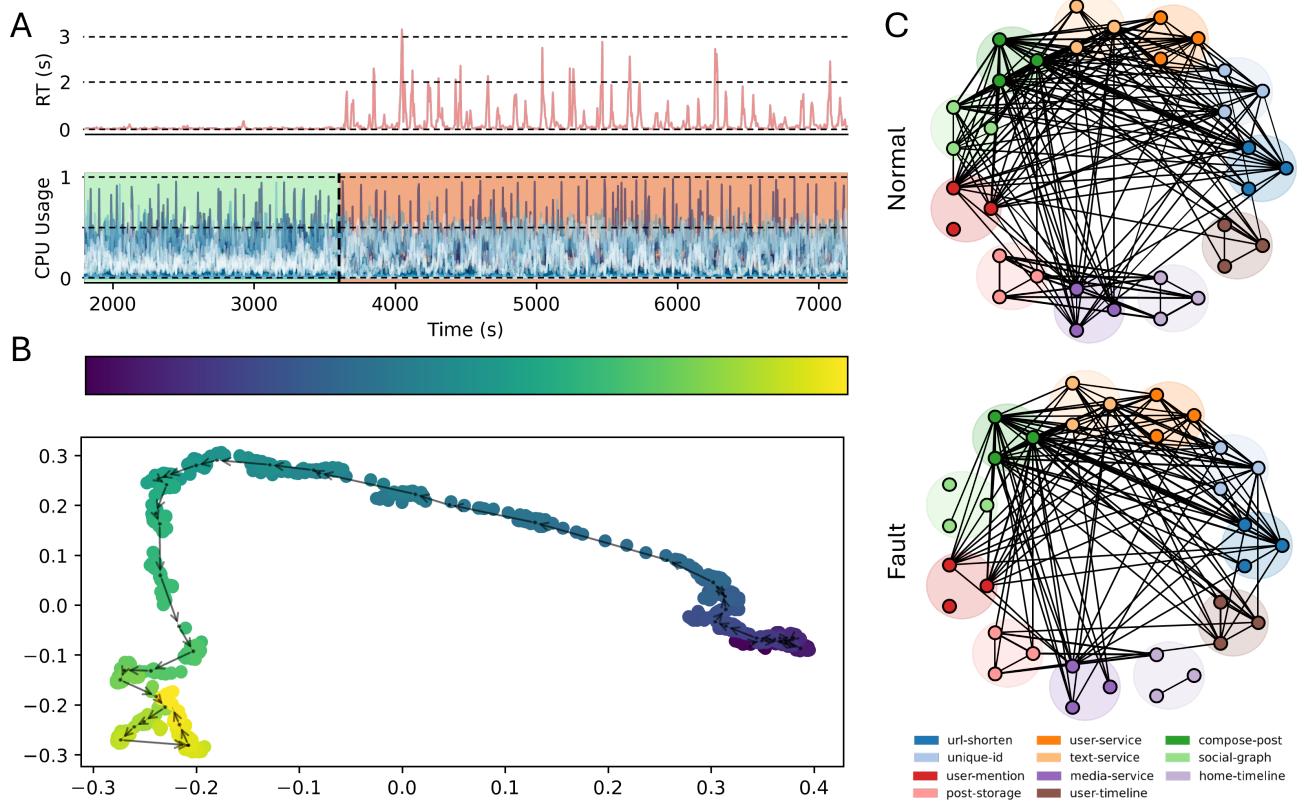


**Figure 2: Panel A:** CPU usage for each individual MS instance over the course of the experiment, with the background colour indicating only *read home timeline* user behaviour (blue), time-varying mixed user behaviour (green), and stationary user behaviour with changing amplitude (pink). **Panel B:** The LMDS 2D embedding of spectral distances between FC graphs, with colour indicating temporal ordering.

view post timelines. This deployment comprises 13 unique MS, supported by 12 in-memory and on-disk data stores. We deploy the MS system on Kubernetes instrumented with Prometheus and Jaeger for metric and trace collection. The social network application provides three different functionalities to users: reading a user’s own timeline, reading another user’s timeline, or composing a new social media post. To simulate the system under load previous research generally use the provided *wrk2* HTTP workload generator, however, this workload generator only produces static, non-fluctuating, traffic. One of the difficulties of anomaly detection and fault localisation from MSAs is that they are in flux in response to changing workload [31, 62]. Therefore, we provisioned the social media application with *Locust* [30] instead and generated time-varying workload behaviour based on the experimental design described in [21]. Specifically, workloads were generated as the sum of sinusoidal waves with randomly generated frequencies in the range  $0.01\text{Hz} \leq f \leq 0.06\text{Hz}$ . In all cases, traces were collected in a probabilistic manner at a rate of 0.5% and metrics were collected at a fixed sampling period of 5s from all MS. To improve the realism of the deployment we used *minikube* [2] to deploy the MSA across 5 virtualised worker nodes each with 2 CPU cores and 8000 GB of memory.

To cover a wide array of different fault conditions for evaluation in Section 4.6 and 4.7 we injected three different types of faults commonly employed within the literature: CPU hog, memory leak, and network delay. Memory leak was injected in two different manners, either with or without MS-specific resource usage limits. In the latter case, rather than being throttled the injected MS utilises most of the available memory on its deployed node, causing resource contention with other co-habiting MSs. This fault represents the memory leak noisy neighbour (NN) condition. Each fault was injected into either the compose post, home timeline, user timeline, or user mention MSs. To create each fault dataset the system was first run with time-varying load for 1hr before fault injection, after which the system runs in the degraded state for another hour before observability data collection. Each condition was repeated 5 times, producing 64 datasets in total, in all cases results in Section 4.6 and 4.7 represent the averages across all collected datasets.

**4.2.2 Alibaba MS Dataset.** The Alibaba 2022 MS dataset provides runtime metric data during 14 days of operation for a large-scale MS deployment on Alibaba’s production cluster supporting a wide array of front-end services [31]. The dataset provides CPU usage information at a MS level with a sampling rate of 60s. However, the dataset does not provide any ground truth as to anomalies that may have occurred during the 13 days of collection, nor what services



**Figure 3:** Panel A: Average response time of HTTP requests (top) and CPU utilisation (bottom) over the course of the experiment with background colour indicating normal (green) and faulty (orange) operation. Panel B: The LMDS 2D embedding of spectral distances between FC graphs, with colour indicating temporal ordering. Panel C: Centroids representing the average FC graph associated with the normal and faulty states of the system.

are being provided. Nonetheless, it provides a good basis in which to benchmark the scalability of detection and localisation approaches.

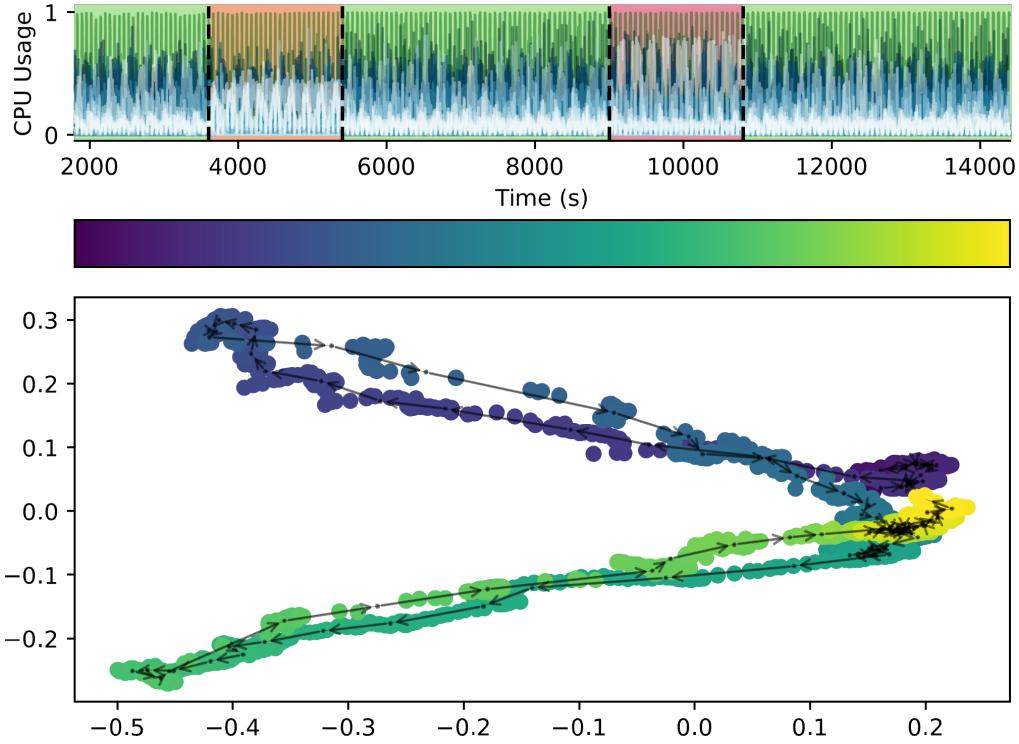
### 4.3 Detection of Time-Varying Changes in MS Dependencies

In this section, we visualise FC-ADL’s ability to capture time-varying changes in dependencies between MSs by inferring and quantifying FC structure rather than focusing on individual metric values.

We generated mixed user traffic for 2.5hrs. For the first 3600s, there was a strict preference for only read home timeline behaviour over user timeline and compose post behaviour. After this initial period, the preference for read home timeline behaviour gradually decreased while the preference for compose post behaviour was increased for a 3600s period. This simulates changing user behaviour at a cohort level. At  $T = 7200$ s, we stopped varying behaviour preferences and instead only increased the number of requests, simulating a spike in traffic, creating a similar pattern of increasing CPU usage but without any underlying change in relationships.

We applied FC-ADL and used landmark multi-dimensional scaling (LMDS) [13] to embed the resultant distance matrix into 2-dimensional space for visualisation, Figure 2. The projection demonstrated that the FC graphs showed an initial jump, representing the

change from only read home timeline behaviour to mixed behaviour, before following a gradual trajectory in space, aligning temporally with the gradual changes in user behaviour. When user behaviour changes stopped but amplitude increased, the velocity of changes within the system slowed dramatically, with only a slight drift halting almost altogether by the end of the experiment. To some extent, this drift could reflect an artifact of using a sliding windowed approach. Nevertheless, these results demonstrate that our approach can accurately uncover time-varying dependency changes between MS, and highlight when visually observable changes in metric data do not actually imply changes in correlation structure. Approaches that only inspect statistical changes in metrics in isolation would not be able to differentiate between true changing behaviour and demand spikes as they only consider the changes in the absolute values of single metrics. Importantly, FC-ADL does not assume *a priori* information about whether a change indicates a fault, in this case the change in user behaviour may or may not represent an anomaly depending on the cause of the change. Rather FC-ADL detects time-varying changes in dependency structure and informs operators as to what has caused these changes, providing actionable insights if required.



**Figure 4:** Panel A: CPU usage for each individual MS instance over the course of the experiment, with background colour indicating normal (green) and faulty (orange) operation. Panel B: The LMDS 2D embedding of spectral distances between FC graphs, with colour indicating temporal ordering.

#### 4.4 Detecting Faulty Behaviour

In this section, we visually explore whether faulty behaviour in the MS system can be captured through changes in the correlation structure of MS CPU usage and whether these changes could provide informative feedback to network operators. For the purpose of this experiment, we simulate a bottle-neck fault by injecting faults into two of three instances of the home timeline service that caused them to lose functionality.

First, we generated mixed user traffic for 2 hours. For the first hour all MSs operated normally. However, after the first hour a fault was injected within two instances of the home timeline service, resulting in increased system latency (Figure 3A). Using the CPU usage data from this two hour period, we applied FC-ADL. When embedding the resulting FC graphs into a 2-dimensional space using LMDS, we found two distinct groupings corresponding to normal and faulty periods, with a transition period after fault onset (Figure 3B). This visualisation clearly indicates a change at  $T = 3600$ s, which persists until the end of the recording.

To investigate as to whether the structural changes detected by DeltaCon provide insights into the root cause of the injected fault, we applied HDBSCAN to cluster the inferred FC graphs as outlined in Section 3.3. We found that the resultant clusters grouped the inferred FC graphs based on the corresponding normal and faulty period. When inspecting the centroid FC dependency graph for each cluster, depicted in Figure 3C, we identified clear structural markers for the faulty state. Most notably, we found that the FC dependencies

between the faulty instances of the home timeline services and other MSs were notably absent, reinforcing our conjecture that the difference between the normal and abnormal FC dependency structure can be utilised for RCA information.

#### 4.5 Detecting Multiple Faulty Behaviours

The previous section showed the ability of our approach to detect transitions between healthy and faulty states for a single fault type. However, real world MS systems are susceptible to a wide array of faults. In this section, we examine FC-ADL's ability to detect transitions between different fault states. We generated mixed traffic for 4 hours with all MSs operating as normal for the first hour. After this period, a transient bottleneck fault was injected into the home timeline service for 30 minutes. After a further hour of normal operation, a transient bottleneck fault was instead injected into the compose post service. Again, this fault was set to last for just 30 minutes.

When applying our framework and using LMDS to visualise the resultant distance matrix in 2-dimensional space, we observed distinct trajectories for the two faults, Figure 4. During the first fault, the system transitioned from the healthy cluster to a cluster representing the home timeline fault, before returning to the healthy cluster. When the second fault was triggered, the system made a second transition from the healthy cluster to another, distinct, cluster representing the compose post fault, again returning to the healthy cluster upon fault remediation. The separation of these fault

**Table 1: Precision, Recall, F1 score, and time taken for each approach when applied to the four different fault conditions (best scores and fastest times in bold).**

Fault	Method	F1	Precision	Recall	Time (s)
CPU Hog	N-Sigma	0.77	0.65	0.98	1.83
	BIRCH	0.48	0.38	0.67	<b>0.01</b>
	SPOT	0.72	0.56	<b>1.00</b>	1.98
	BARO	0.72	0.77	0.73	129.77
	FC-ADL (ours)	<b>0.90</b>	<b>0.95</b>	0.89	0.88
Memory Leak	N-Sigma	0.80	0.67	1.00	1.83
	BIRCH	0.64	0.50	0.88	<b>0.01</b>
	SPOT	0.73	0.57	<b>1.00</b>	2.27
	BARO	<b>0.84</b>	<b>0.91</b>	0.84	133.43
	FC-ADL (ours)	0.81	0.90	0.76	0.84
Memory Leak (NN)	N-Sigma	0.74	0.59	<b>1.00</b>	1.84
	BIRCH	0.00	0.00	0.00	<b>0.01</b>
	SPOT	0.72	0.56	<b>1.00</b>	1.56
	BARO	<b>0.94</b>	<b>1.00</b>	0.89	128.24
	FC-ADL (ours)	0.91	0.97	0.87	0.82
Network Delay	N-Sigma	0.75	0.60	<b>1.00</b>	1.82
	BIRCH	0.45	0.35	0.62	<b>0.00</b>
	SPOT	0.72	0.56	<b>1.00</b>	1.90
	BARO	0.14	0.19	0.11	126.28
	FC-ADL (ours)	<b>0.82</b>	<b>0.85</b>	0.82	0.86

clusters suggests our approach can distinguish between different types of faults, enabling cluster-based fault labelling. This could enhance the diagnostic capabilities of the approach by enabling quicker identification and resolution of issues based on historical responses to the same fault.

## 4.6 Anomaly Detection Evaluation

To evaluate the effectiveness of FC-ADL in detecting anomalous behaviour indicative of faults within MSAs we follow the steps outlined in previous work [9, 44], treating detection models as binary classifiers. The metric data collected across all fault conditions outlined in Section 4.2 is labelled as normal for data points that occurred prior to fault injection time, and abnormal for all data points after. Similarly to [44] for all benchmark algorithms and FC-ADL the first detected change-point is taken as the anomaly decision boundary, labelling all prior points as normal and all subsequent points as abnormal. Predictions can then be compared to the ground-truth for each dataset to find True Positives (TP) based on correct abnormal labelling, True Negatives (TN) based on correct normal labelling, False Positives (FP) based on incorrect abnormal labelling, and False Negatives (FN) based on incorrect normal labelling. These values can then be used to calculate the Precision, Recall, and F1 scores for each approach.

**4.6.1 Baselines.** For anomaly detection we select four baselines: N-Sigma [25, 27], BIRCH [65, 67], SPOT [24, 25, 40], and Multivariate Bayesian Online Change point Detection (MBOCPD) as implemented by BARO [44]. These approaches have been selected

due to their common use for anomaly detection RCA research as well as state-of-the-art performance [44].

- **N-Sigma:** is a simple anomaly detection method utilised across many RCA works for fault detection [24, 25, 27, 70]. In this work we use the 3-Sigma rule implementation [44], being the most common version of N-Sigma, which relies on the fact that the vast majority of data points in a normal distribution fall within three standard deviations of the mean. Therefore, historical data is used to estimate the mean and standard deviations of metrics, and then for anomaly detection anything that falls outside of the three standard deviation threshold during runtime is flagged as anomalous.
- **BIRCH:** this method of anomaly detection has been implemented in prior RCA work [65, 67] and uses BIRCH clustering applied to historical, normal, metric values [65, 67]. In this paper we use the implementation of BIRCH described in [44] in which BIRCH considers a point to be anomalous when it is assigned a different cluster with consecutive data points.
- **SPOT:** is an approach based off of Extreme Value (EV) Theory [56]. In this work we use the dSPOT variant of SPOT used in [40] and previous benchmarks [44], hereafter any references to SPOT refer to the dSPOT variant. This variant of SPOT is designed to handle shifts in the underlying time series by first removing local trends via a sliding-window model and maintaining a dynamic EV threshold based on

iterative re-fitting. Values that exceed the EV threshold are considered anomalous.

- **BARO:** carries out anomaly detection via Multivariate Bayesian Change Point Detection, an approach that models the number of consecutive points since the last distribution change to detect change points in multivariate time series metrics. Unlike the above approaches, but similar to our approach, BARO inherently captures the relationships between metrics making it more robust for change point detection from MSAs [44].

**4.6.2 Results.** Our results in Table 1 demonstrate that FC-ADL outperformed N-Sigma, BIRCH, and SPOT in all cases in terms of F1 score. When compared to BARO, we find that FC-ADL performs similarly for fault detection, having slightly higher F1 scores in two cases (network delay and memory leak) and slightly lower F1 scores in two others (CPU hog and memory leak noisy neighbour). However, BARO’s runtime was significantly higher than any other anomaly detection approach, being 152x slower than FC-ADL on average. The high computational overhead of BARO comes from the the inherent scalability limitations of BOCPD discussed in Section 1 as the number of variables and number of samples increases. Overall, these results demonstrate that FC-ADL delivers state-of-the-art accuracy in detecting the onset of anomalous MSA behaviour from FC inferred from MS metrics, while remaining highly efficient when compared to BARO.

Whilst N-Sigma and SPOT consistently demonstrated the highest recall results their overall F1 scores were significantly affected by their poor precision due to routinely predicting anomaly onsets early. These results are likely a product of our more realistic experimental setup with varying user load (see Section 4.2). Without consideration for the relationship between metrics, like with BARO and FC-ADL, these methods are sensitive to changes in metric values caused by normal changes in user demand. Overall, these findings illustrate the importance of developing anomaly detection approaches for MSAs that consider their inherent inter-dependent nature and testing approaches in more realistic and time-varying environments.

One interesting finding from our evaluation is that faults not immediately directly affecting CPU usage, such as memory leaks and network delay, can still be accurately detected using only CPU usage data, especially in the case of BARO and FC-ADL. This indicates that subtle shifts in CPU usage reflect fault-induced performance changes and coupling between microservices. Consequently, this raises questions about the minimal variety of metric sources needed for effective anomaly detection, potentially enabling more efficient observability.

## 4.7 Root Cause Localisation Evaluation

To evaluate the capability of FC-ADL to accurately identify the root cause of faults we compare its performance to several baselines in extracting the top-K most likely root cause from the datasets discussed in Section 4.2. All baselines and FC-ADL were applied to the full dataset for anomaly detection and subsequent RCA. For baselines that do not include anomaly detection (CIRCA,  $\epsilon$ -Diagnosis), the ground-truth fault onset time is given instead as input. For each type of fault (CPU hog, memory leak, memory

leak noisy neighbour, and network delay) we report the average top-K accuracy, indicating how often each method listed the true root cause MS that received the injected fault in the top 1, 3, or 5 candidate faulty MS(s). In each case, we also report the average time taken for exclusively RCA.

**4.7.1 Baselines.** For RCA we select six baseline approaches:  $\epsilon$ -Diagnosis [51], N-Sigma [25, 27], CIRCA [25], PC [57], LiNGAM [53, 54], and BARO [44]. As in Section 4.6, these approaches were selected due to their open-source implementations, state-of-the-art performance, and use in previous work as benchmarks.

- **$\epsilon$ -Diagnosis:** an unsupervised RCA method that utilises non-parametric two-sample hypothesis testing using  $\epsilon$ -statistics [51] to quantify how much the distribution of metrics between normal and abnormal data has changed into a p-value. Metrics whose p-value falls below a confidence threshold are flagged as candidate root causes and are ranked by total p-value.
- **N-Sigma:** carries out RCA by comparing the mean and standard deviation of each metric over the normal data to the detected abnormal period [25, 27]. Each metric is then scored based on the maximum z-score, quantifying how far each value is from its pre-fault average. Each metric is ranked by descending score to give the top candidate root causes.
- **CIRCA:** creates a causal graph by using tracing data and domain knowledge [25]. As in prior work [44] we supply a causal graph constructed by the PC algorithm from metrics based on the normal functioning of the system. A regression model is then trained on normal data to estimate its conditional distribution. The model can then be applied to the anomalous data to quantify how much each metric’s values deviate from the normal period. Candidate root causes are then ranked based on their total deviation.
- **PC** [57]: a popular approach for constructing causal graphs between metrics is the PC algorithm [28, 33–35, 67] and thus is a common baseline in prior research [44, 67]. Following the design of MicroDiag [65] we provide edge-weights based on the correlation between metrics and inverse directed edges before applying PageRank to list the top-K root causes.
- **LiNGAM** [53, 54]: another popular approach for constructing causal graphs between metrics is the LiNGAM algorithm [65]. Similarly to above we follow MicroDiag [65] by providing edge-weights based on the correlation between metrics and inverse directed edges before applying PageRank to list the top-K root causes.
- **BARO:** similarly to  $\epsilon$ -Diagnosis and N-Sigma, BARO carries out fault localisation via non-parametric hypothesis testing using their *Robust Scorer*, comparing the statistical distributions of the detected normal and abnormal data. Unlike previous approaches, *Robust Scorer* uses the median and interquartile range which are more robust to outliers making the approach more resilient to imprecise anomaly time onset estimates.

**Table 2: Average top-1, top-3, and top-5 accuracy alongside corresponding computation time for each approach when applied to the four different fault conditions (best scores and fastest times in bold).**

Fault	Method	Avg@1	Avg@3	Avg@5	Time (s)
CPU Hog	$\epsilon$ -Diagnosis	0.00	0.08	0.15	0.70
	PC+PR	0.00	0.00	0.00	4.93
	DirectLiNGAM+PR	0.06	0.06	0.06	1.63
	CIRCA	0.71	0.83	<b>0.92</b>	3.06
	N-Sigma	0.00	0.06	0.15	<b>0.01</b>
	BARO	0.06	0.44	0.77	0.03
	FC-ADL (ours)	<b>0.85</b>	<b>0.85</b>	0.85	0.02
Memory Leak	$\epsilon$ -Diagnosis	0.06	0.19	0.19	0.70
	PC+PR	0.00	0.00	0.00	4.79
	DirectLiNGAM+PR	0.00	0.06	0.06	1.64
	CIRCA	<b>0.81</b>	<b>0.94</b>	<b>1.00</b>	3.10
	N-Sigma	0.12	0.12	0.19	<b>0.00</b>
	BARO	0.00	0.44	0.88	0.02
	FC-ADL (ours)	0.68	0.81	0.90	0.02
Memory Leak (NN)	$\epsilon$ -Diagnosis	0.00	0.06	0.25	0.70
	PC+PR	0.00	0.00	0.00	1.27
	DirectLiNGAM+PR	0.00	0.00	0.00	1.65
	CIRCA	0.00	0.19	0.38	1.99
	N-Sigma	0.00	0.12	0.38	<b>0.01</b>
	BARO	0.00	0.00	0.25	0.02
	FC-ADL (ours)	<b>0.10</b>	<b>0.34</b>	<b>0.55</b>	0.03
Network Delay	$\epsilon$ -Diagnosis	0.19	0.19	0.25	0.70
	PC+PR	0.00	0.12	0.19	5.62
	DirectLiNGAM+PR	0.00	0.12	0.12	1.64
	CIRCA	0.00	0.31	0.38	2.89
	N-Sigma	0.00	0.00	0.06	<b>0.01</b>
	BARO	0.00	0.00	0.00	0.01
	FC-ADL (ours)	<b>0.28</b>	<b>0.50</b>	<b>0.51</b>	0.02

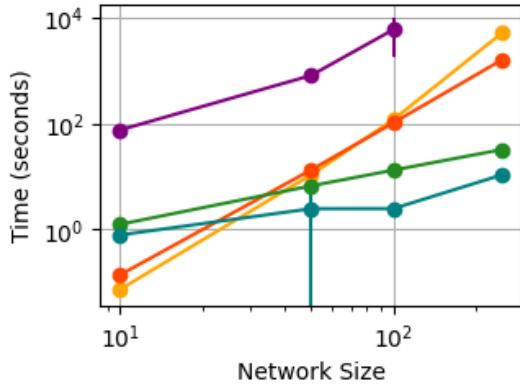
**4.7.2 Results.** Our results in Table 2 demonstrate that across all faults FC-ADL outperformed, or performed on-par with, benchmark approaches in accurately identifying candidate root causes. Whilst most approaches struggled with identifying the true root cause in the top-1 of candidates, both FC-ADL and CIRCA managed to achieve impressive accuracy for CPU hog and memory leak faults. Most approaches saw improved top-3 and top-5 accuracy across all fault types, notably, FC-ADL performed above all baselines in accurately identifying the root cause of memory leak (NN) and network delay faults. Whilst CIRCA demonstrated impressive performance across all fault types the approach does not provide end-to-end fault detection and localisation, and thus ground truth injection time was supplied. In practice, it's almost impossible to pinpoint the exact moment a fault occurs so as to cleanly partition the metrics into pre-fault and post-fault sets, and the resulting data contamination is likely to impair the accuracy of the logistic regression model. The performance of BARO in identifying the true root cause in the top-1 and top-3 candidates was unexpectedly poor given its excellent performance in Section 4.6, however, across CPU Hog and memory leak faults BARO saw notable improvements in

top-5 accuracy to near comparable performance of CIRCA and FC-ADL. The overall poor performance of many baseline approaches for RCA on the generated datasets with time-varying demand and thus non-stationary metrics may indicate the necessity to test such approaches on more realistic environments to better understand their potential real-world performance.

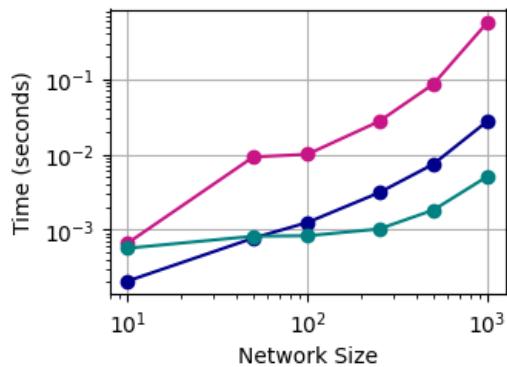
Our results showed that most approaches saw significant reductions in performance in accurately locating memory leak (NN) and network delay faults despite these same reductions not being substantially noticeable in the evaluation of anomaly detection in Section 4.6. As this same decrease was not observed in the normal memory leak scenario, it suggests that the resource contention, and thus indirect fault propagation with co-located MSs, in the memory leak NN scenario may mask performance signatures that RCA approaches rely on. As indirect fault propagation through cohabitation is expected within real-world MSAs [35], but is under-represented in RCA test bed research, future work should consider such fault propagation when benchmarking their approaches to better understand their performance under diverse fault conditions. The decrease in performance for network delay faults may indicate that whilst CPU usage in isolation can help accurately identify

the onset of an anomaly, it may not be sufficient in all cases to accurately identify the root cause.

Whilst the computational overhead of all approaches was low in our experiments, we note that even at such a small scale the computational efficiency of FC-ADL is noticeable. Overall, FC-ADL demonstrated a 202x and 75x speed up compared to PC and DirectLiNGAM respectively.



**Figure 5: Scalability of BARO (purple), the PC algorithm (yellow), DirectLiNGAM (orange), N-Sigma (green), and FC-ADL (teal) to the Alibaba deployment.**



**Figure 6: Scalability of each component of FC-ADL, EWMA inference (blue), DeltaCon (pink), and HDBSCAN (teal) for a single window.**

#### 4.8 Scalability Analysis

To illustrate FC-ADL’s scalability, we used a real-world dataset released by Alibaba [32] to test the scalability of FC-ADL and other baseline RCA approaches. As there is no ground-truth information in the dataset as to the onset of faults, we only include RCA approaches that do not require specific fault onset timings. We applied each approach to increasingly large numbers of randomly sampled MSs from the Alibaba dataset using 1080 samples of CPU

usage data from each MS, corresponding to 12 hours of recordings. We only demonstrate up to 250 MSs, and only 100 MSs for BARO, due to the extreme computational overheads of approaches beyond these points. Our results in Figure 5, demonstrate that for just 100 nodes BARO’s end-to-end approach exceeds processing times of 1hr. Whilst DirectLiNGAM and PC perform better, they still near 1hr processing times when applied to just 250 MSs. On the other hand, FC-ADL took 3s for 100 MSs and just 11s for 250 MSs to provide actionable RCA information. It is worth noting that N-Sigma achieved similar performance to that of FC-ADL, however, we observe that N-Sigma performs poorly in both anomaly detection and localisation accuracy (Sections 4.6 and 4.7). Therefore, it is unclear as to how valuable the insights produced by N-Sigma would be despite its efficiency. These results highlight the inability of prior causal inference and multivariate based approaches to scale to large-scale deployments, and demonstrates the comparative computational efficiency of FC-ADL.

One of the benefits of FC-ADL is that, unlike other approaches evaluated in this paper [25, 25, 27, 44, 51, 65], it can be applied as data arrives to construct new FC-based dependencies in an online manner. Our results in Figure 6 demonstrate that when FC-ADL is applied to construct a single window, calculate DeltaCon distance, and use HDBSCAN to predict the cluster, the approach remains sub-second even up to 1000 MSs. This further reinforces the scalability of FC-ADL when applied to large-scale MSAs, highlighting its suitability for real-time, incremental fault localisation where continuous monitoring and rapid insights are essential.

Utilising only metric data comes with additional benefits of reduced data storage and collection costs. The uncompressed size on disk of both CPU and memory usage data for all MS instances, around 18x greater than the number of unique MS, over a day is 52.3GB after post-processing. On the other hand, the size on disk of distributed tracing data, only for unique MSs and sampled at just 0.5%, is 713.5GB after post-processing. This signifies a 92.7% reduction in data size whilst retaining a much higher spatial granularity of data at an instance-level.

#### 5 Conclusion

In this paper we have introduced FC-ADL, a framework for monitoring time-varying dependencies within MS systems by applying FC to MS usage data. We demonstrated that by tracking changes in these dependencies, FC-ADL can accurately detect the occurrence of anomalies. Furthermore, we demonstrated that by examining the dependency structure before and after a fault has occurred, the most likely candidate cause for the anomaly can be found. We demonstrated that overall, FC-ADL can achieve state-of-the-art performance in both anomaly detection and RCA on a wide variety of fault conditions and MSs, and does so with low computational overhead.

A design choice prevalent throughout FC-ADL is a focus on scalability, imparted due to the usage of efficient correlation calculations to infer the time-varying FC between MSs. This efficiency enables FC-ADL to be deployed to large-scale MSAs to provide timely fault detection and localisation information, which is difficult or impossible with prior work. One important contribution of FC-ADL is demonstrating that causation is not necessarily required

for accurate fault localisation. Indeed, we suggest that correlation-based approaches may be preferred in cases where timely fault localisation is paramount, especially in cases where the number of variables is very large or when fault data is limited due to decreased complexity of modelling.

Whilst FC-ADL is indeed computationally efficient, the use of Pearson's correlations can result in somewhat noisy FC inference. This can mean that more data is required per window to infer meaningful correlations, which can then mean that the framework is slower to adapt to changes in dependency structure, even when mitigated by a EWMA. Future iterations of FC-based anomaly detection and RCA within MSAs could explore alternative, more computationally demanding, statistical measures that can provide more accurate and robust measures of functional coupling [16]. Another avenue for future work is to explore the use of FC-based prior knowledge to reduce the computational overhead of causal inference approaches, which has been demonstrated to achieve significant speed-ups when applied to non-MSA architectures [63]. These avenues could provide options for a trade-off between fast and efficient RCA and potentially more accurate RCA, providing a hierarchical triage framework.

## References

- [1] 2025. *FC-ADL/FC-ADL-SoCC-2025*. <https://github.com/FC-ADL/FC-ADL-SoCC-2025>
- [2] 2025. Minikube: Run Kubernetes Locally. <https://minikube.sigs.k8s.io/>. Accessed: 2025-07-14.
- [3] Luc Bauwens, Sébastien Laurent, and Jeroen V. K. Rombouts. 2006. Multivariate GARCH models: a survey. *J. Appl. Econom.* 21, 1 (2006), 79–109. doi:10.1002/jae.842
- [4] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. 2013. Density-Based Clustering Based on Hierarchical Density Estimates. In *Adv. Knowl. Discov. Data Min. (PAKDD) (Lecture Notes in Computer Science, Vol. 7819)*. Springer, 160–172. doi:10.1007/978-3-642-37456-2\_14
- [5] Chaos Mesh. 2021. *chaos-mesh*. <https://github.com/chaos-mesh/chaos-mesh>
- [6] Hongyang Chen, Pengfei Chen, and Guangbu Yu. 2020. A Framework of Virtual War Room and Matrix Sketch-Based Streaming Anomaly Detection for Microservice Systems. *IEEE Access* 8 (2020), 43413–43426. doi:10.1109/ACCESS.2020.2977464
- [7] Junjie Chen, Xiaoting He, Qingwei Lin, Yong Xu, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. An Empirical Investigation of Incident Triage for Online Service Systems. In *2019 IEEE/ACM 41st ICSE-SEIP*, 111–120. doi:10.1109/ICSE-SEIP.2019.00020
- [8] Jian Chen, Fagui Liu, Jun Jiang, Guoxiang Zhong, Dishi Xu, Zhaoglu Tan, and Shangsong Shi. 2023. TraceGra: A trace-based anomaly detection for microservice using graph deep learning. *Computer Comm.* 204 (2023), 109–117. doi:10.1016/j.comcom.2023.03.028
- [9] Pengfei Chen, Yong Qi, and Di Hou. 2019. CauseInfer: Automated End-to-End Performance Diagnosis with Hierarchical Causality Graph in Cloud Environment. *IEEE Trans. Serv. Comput. (TSC)* 12, 2 (2019), 214–230. doi:10.1109/TSC.2016.2607739
- [10] Adrian Cockcroft. 2023. *Microservices Retrospective – What We Learned (and Didn't Learn) from Netflix*. <https://www.infoq.com/presentations/microservices-netflix-industry/>
- [11] Marlene R. Cohen and Adam Kohn. 2011. Measuring and interpreting neuronal correlations. *Nat. Neurosci.* 14, 7 (2011), 811–819. doi:10.1038/nn.2842
- [12] Diego Colombo and Marloes H. Maathuis. 2014. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.* 15, 1 (2014), 3921–3962. doi:10.1145/3221269.3221292
- [13] Vin De Silva and Joshua B Tenenbaum. 2004. *Sparse multidimensional scaling using landmark points*. Technical Report. technical report, Stanford University.
- [14] Christina Eder, Stefan Winzinger, and Robin Lichtenthäler. 2023. A Comparison of Distributed Tracing Tools in Serverless Applications. In *Proc. IEEE Int. Conf. Serv.-Oriented Syst. Eng. (SOSE)* (Athens, Greece, 2023). doi:10.1109/SOSE58276.2023.00018
- [15] André Fachat. [n. d.]. *Challenges and benefits of the microservice architectural style*. <https://developer.ibm.com/articles/challenges-and-benefits-of-the-microservice-architectural-style-part-1/complex-system-communications>
- [16] Karl J Friston. 2011. Functional and effective connectivity: a review. *Brain Connect.* 1, 1 (2011), 13–36. doi:10.1089/brain.2011.0008
- [17] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rath, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. 2019. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud and Edge Systems. In *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.* (Providence, RI, USA), 3–18. doi:10.1145/3297858.3304013
- [18] David Graf, Lisa Spitzl, Michael Steiner, Wieland Schwinger, Werner Retschitzegger, Elisabeth Kapsammer, Birgit Pröll, and Norbert Baumgartner. 2022. Semantic-Driven Mining of Functional Dependencies in Large-Scale Systems-of-Systems. In *Inf. Technol. Syst.* (Cham, 2022). Springer, 344–355. doi:10.1007/978-3-030-96293-7\_31
- [19] Anton Gulenko, Florian Schmidt, Alexander Acker, Marcel Wallschläger, Odej Kao, and Feng Liu. 2018. Detecting Anomalous Behavior of Black-Box Services Modeled with Distance-Based Online Clustering. In *Proc. 11th IEEE Int. Conf. Cloud Comput. (CLOUD)*, 912–915. doi:10.1109/CLOUD.2018.00134
- [20] Lexiang Huang and Timothy Zhu. 2021. tprof: Performance profiling via structural aggregation and automated analysis of distributed systems traces. In *Proc. ACM Symp. Cloud Comput. (SoCC)* (Seattle WA USA). doi:10.1145/3472883.3486994
- [21] Azam Ikram et al. 2022. Root Cause Analysis of Failures in Microservices through Causal Discovery. *Adv. in Neur. Inf. Pro. Sys.* 35 (2022), 31158–31170.
- [22] Mingxu Jin et al. 2020. An Anomaly Detection Algorithm for Microservice Architecture Based on Robust Principal Component Analysis. *IEEE Access* 8 (2020), 226397–226408. doi:10.1109/ACCESS.2020.3044610
- [23] Danai Koutra, Joshua T. Vogelstein, and Christos Faloutsos. 2013. DeltaCon: A Principled Massive-Graph Similarity Function. In *Proc. SIAM Data Mining Conference* Society for Industrial and Applied Mathematics, 162–170. doi:10.1137/1.9781611972832.18
- [24] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R. Lyu. 2023. Eadro: An End-to-End Troubleshooting Framework for Microservices on Multi-Source Data. In *Proc. 45th Int. Conf. Softw. Eng.* (Melbourne, Victoria, Australia), 1750–1762. doi:10.1109/ICSE48619.2023.00150
- [25] Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchang Zhang, Kaixin Sui, and Dan Pei. 2022. Causal Inference-Based Root Cause Analysis for Online Service Systems with Intervention Recognition. In *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Min. (KDD)* (Washington DC, USA), 3230–3240. doi:10.1145/3534678.3539041
- [26] Min Li, Dingyong Tang, Zepeng Wen, and Yunchang Cheng. 2021. Universal Anomaly Detection Method Based on Massive Monitoring Indicators of Cloud Platform. In *Proc. 2021 IEEE Int. Conf. Softw. Eng. Artif. Intell. (SEAI)* (Xiamen, China, 2021), 23–29. doi:10.1109/SEAI52285.2021.9477532
- [27] Jinjin Lin, Pengfei Chen, and Zibin Zheng. 2018. Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments. In *Proc. Int. Conf. Serv.-Oriented Comput. (ICSO)* (Hangzhou, China), 3–20. doi:10.1007/978-3-030-03596-9\_1
- [28] Weilan Lin, Meng Ma, Disheng Pan, and Ping Wang. 2018. FacGraph: Frequent Anomaly Correlation Graph Mining for Root Cause Diagnose in Micro-Service Architecture. In *Proc. 2018 IEEE 37th Int. Perform. Comput. Commun. Conf. (IPCCC)* (Orlando, FL, USA), 1–8. doi:10.1109/PCCC.2018.8711092
- [29] Ping Liu, Haowen Xu, Qianyu Ouyang, Rui Jiao, Zhekang Chen, Shenglin Zhang, Jiahai Yang, Linlin Mo, Jice Zeng, Wenman Xue, and Dan Pei. 2020. Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks. In *Proc. 2020 IEEE 31st Int. Symp. Softw. Reliab. Eng. (ISSRE)*, 48–58. doi:10.1109/ISSRE5003.2020.00014
- [30] Locust.io. [n. d.]. *Locust.io*. <https://locust.io/>
- [31] Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. 2021. Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis. In *Proc. ACM Symp. Cloud Comput. (SoCC)* (Seattle, WA, USA), 412–426. doi:10.1145/3472883.3487003
- [32] Shutian Luo, Huanle Xu, Kejiang Ye, Guoyao Xu, Liping Zhang, Guodong Yang, and Chengzhong Xu. 2022. The power of prediction: microservice auto scaling via workload learning. In *Proc. 13th ACM Symp. Cloud Comput. (SoCC)*, San Francisco, CA, USA, 355–369. doi:10.1145/3542929.3563477
- [33] Meng Ma, Weilan Lin, Disheng Pan, and Ping Wang. 2019. Ms-rank: Multi-metric and self-adaptive root cause diagnosis for microservice applications. In *Proc. IEEE Int. Conf. Web Serv. (ICWS)* (Milan, Italy), 60–67. doi:10.1109/ICWS2019.000022
- [34] Meng Ma, Weilan Lin, Disheng Pan, and Ping Wang. 2021. ServiceRank: Root Cause Identification of Anomaly in Large-Scale Microservice Architectures. *IEEE Trans. Depend. Secure Comput.* 19, 5 (2021), 3087–3100. doi:10.1109/TDSC.2021.3083671
- [35] Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. 2020. AutoMAP: Diagnose Your Microservice-based Web Applications Automatically. In *Proc. WWW '20* (Taipei, Taiwan), 246–258. doi:10.1145/3366423.3380111

- [36] Leonardo Mariani, Cristina Monni, Mauro Pezzé, Oliviero Riganelli, and Rui Xin. 2018. Localizing faults in cloud systems. In *Proc. IEEE 11th Int. Conf. Softw. Test. Verif. Valid. (ICST)* (Västerås, Sweden). 262–273. doi:10.1109/ICST.2018.00034
- [37] Naoki Masuda and Petter Holme. 2019. Detecting sequences of system states in temporal networks. *Sci. Rep.* 9, 1 (2019), 795. doi:10.1038/s41598-018-37534-2
- [38] Lun Meng, Feng Ji, Yao Sun, and Tao Wang. 2021. Detecting anomalies in microservices with execution trace comparison. *Future Gener. Comput. Syst.* 116 (2021), 291–301. doi:10.1016/j.future.2020.10.040
- [39] Lun Meng, Yao Sun, and Shudong Zhang. 2020. Midagi: A Sequential Trace-Based Fault Diagnosis Framework for Microservices. In *Proc. 17th Int. Conf. Serv. Comput. (SCC 2020), Part of Serv. Conf. Fed. (SCF 2020), Honolulu, HI, USA, Sep. 2020* (Honolulu, HI, USA). 137–144. doi:10.1007/978-3-030-59592-0\_9
- [40] Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. 2020. Localizing Failure Root Causes in a Microservice through Causality Inference. In *Proc. IEEE/ACM 28th Int. Symp. on Quality of Service (IWQoS)*. 1–10. doi:10.1109/IWQoS49365.2020.9213058
- [41] Antoine Messager et al. 2019. Inferring Functional Connectivity From Time-Series of Events in Large Scale Network Deployments. *IEEE Trans. Netw. Serv. Manage.* 16, 3 (2019), 857–870. doi:10.1109/TNSM.2019.2932896
- [42] Sasho Nedelkoski, Jorge Cardoso, and Odej Kao. 2019. Anomaly Detection and Classification using Distributed Tracing and Deep Learning. In *Proc. 2019 IEEE/ACM 19th Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*. 241–250. doi:10.1109/CCGRID.2019.00038
- [43] Mahsa Panahandeh, Abdelrahab Hamou-Lhadj, Mohammad Hamdaqa, and James Miller. 2024. ServiceAnomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics. *J. Syst. Softw.* 209 (2024), 111917. doi:10.1016/j.jss.2023.111917
- [44] Luan Pham, Huong Ha, and Hongyu Zhang. 2024. BARO: Robust Root Cause Analysis for Microservices via Multivariate Bayesian Online Change Point Detection. *Proc. ACM on Softw. Eng.* 1 (2024). doi:10.1145/3660805
- [45] F. Pozzi, T. Di Matteo, and T. Aste. 2012. Exponential smoothing weighted correlations. *Eur. Phys. J. B* 85, 6 (2012), 175. doi:10.1140/epjb/e2012-20697-x
- [46] Juan Qiu et al. 2020. A Causality Mining and Knowledge Graph Based Method of Root Cause Diagnosis for Performance Anomaly in Cloud Applications. *Appl. Sci.* 10, 6 (2020), 2166. doi:10.3390/app10062166
- [47] Areeg Samir, Nabil El Ioini, Ilenia Franza, Hamid R Barzegar, Van Thanh Le, and Claus Pahl. 2020. Anomaly Detection and Analysis for Reliability Management in Clustered Container Architectures. *Int. J. Adv. Syst. Meas.* 12, 3 (2020), 247–264.
- [48] Areeg Samir and Claus Pahl. 2019. DLA: Detecting and Localizing Anomalies in Containerized Microservice Architectures Using Markov Models. In *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*. 205–213. doi:10.1109/FiCloud.2019.00036
- [49] V. Sesagiri, D. Huye, L. Liu, A. Wildani, and R. R. Sambasivan. 2022. [SoK] Identifying Mismatches Between Microservice Testbeds and Industrial Perceptions of Microservices. *Proc. ACM Symp. Cloud Comput. (SoCC)* 2, 1 (2022). doi:10.5070/RS32157839
- [50] Shayan Shahini and Hossien Momeni. 2024. Autoencoder-based Anomaly Detection in Microservices using Distributed Tracing. In *Proc. 20th CSI Int. Symp. Artif. Intell. Signal Process. (AISP)* (Babol, Iran). doi:10.1109/AISP61396.2024.10475273
- [51] Huasong Shan, Yuan Chen, Haifeng Liu, Yunpeng Zhang, Xiao Xiao, Xiaofeng He, Min Li, and Wei Ding. 2019.  $\epsilon$ -diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In *Proc. The Web Conference* (San Francisco, CA, USA). 3215–3222. doi:doi:10.1145/3308558.3313653
- [52] Junxian Shen, Han Zhang, Yang Xiang, Xingang Shi, Xinrui Li, Yunxi Shen, Zijian Zhang, Yongxiang Wu, Xia Yin, Jilong Wang, Mingwei Xu, Yahui Li, Jiping Yin, Jianchang Song, Zhuofeng Li, and Runjie Nie. 2023. Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code. In *Proc. ACM SIGCOMM* (New York, NY, USA). 420–437. doi:10.1145/3603269.3604823
- [53] Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, and Antti Kerminen. 2006. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *J. Mach. Learn.* 7, 72 (2006), 2003–2030. doi:10.5555/1248547.1248619
- [54] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvärinen, Yoshinobu Kawahara, Takashi Washio, Patrik O. Hoyer, and Kenneth Bollen. 2011. DirectLiNGAM: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model. *J. Mach. Learn. Res.* 12 (Jul 2011), 1225–1248.
- [55] Robert H. Shumway and David S. Stoffer. 2017. ARIMA Models. In *Time Series Analysis and Its Applications: With R Examples*. Springer, 75–163.
- [56] Alba Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly detection in streams with extreme value theory. In *Proc. 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*. 1067–1075.
- [57] Peter Spirtes, Clark Glymour, and Richard Scheines. 1993. *Causation, Prediction, and Search*. Lect. Notes Stat., Vol. 81. Springer.
- [58] Cindy Sridharan. 2018. The Need for Observability. In *Distributed Systems Observability*. O'Reilly Media Inc., California, United States.
- [59] Owen Vallis, Jordan Hochenbaum, and Arun Kejariwal. 2014. A novel technique for long-term anomaly detection in the cloud. In *Proc. 6th USENIX Conf. Hot Top.* *Cloud Comput.* (Philadelphia, PA, USA). 15. doi:10.5555/2696535.2696550
- [60] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. 2018. CloudRanger: Root Cause Identification for Cloud Native Systems. In *Proc. IEEE/ACM 18th Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)* (Washington, DC, USA). 492–502. doi:10.1109/CCGRID.2018.00076
- [61] Tao Wang, Wenbo Zhang, Jiwei Xu, and Zeyu Gu. 2020. Workflow-Aware Automatic Fault Diagnosis for Microservice-Based Applications With Statistics. *IEEE Trans. Netw. Serv. Manage.* 17, 4 (2020), 2350–2363. doi:10.1109/TNSM.2020.3022028
- [62] Giles Winchester, George Parisis, and Luc Berthouze. 2023. On the Temporal Behaviour of a Large-Scale Microservice Architecture. In *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)* (Miami, FL, USA). 1–6. doi:10.1109/NOMS56928.2023.10154427
- [63] Giles Winchester, George Parisis, Robert Harper, and Luc Berthouze. 2022. Accelerating Causal Inference Based RCA Using Prior Knowledge From Functional Connectivity Inference. In *Proc. 18th Int. Conf. Netw. Serv. Manage. (CNSM)* (Thessaloniki, Greece). doi:10.23919/CNSM55787.2022.9964900
- [64] Li Wu, Jasmin Bogatinovski, Sasho Nedelkoski, Johan Tordsson, and Odej Kao. 2020. Performance diagnosis in cloud microservices using deep learning. In *Proc. Int. Conf. Serv.-Oriented Comput.* 85–96. doi:10.1007/978-3-030-76352-7\_13
- [65] Li Wu, Johan Tordsson, Jasmin Bogatinovski, Erik Elmroth, and Odej Kao. 2021. MicroDiag: Fine-grained Performance Diagnosis for Microservice Systems. In *Proc. IEEE/ACM Int. Work. Cloud Intell. (CloudIntelligence)*. 31–36. doi:10.1109/CloudIntelligence52565.2021.00015
- [66] Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. 2020. MicroRCA: Root Cause Localization of Performance Issues in Microservices. In *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)* (Budapest, Hungary). 1–9. doi:10.1109/NOMS47738.2020.9110353
- [67] Ruyue Xin, Peng Chen, and Zhiming Zhao. 2023. CausalRCA: Causal inference based precise fine-grained root cause localization for microservice applications. *J. Syst. Softw.* 203, C (Sept. 2023), 13 pages. doi:10.1016/j.jss.2023.111724
- [68] Jingmin Xu, Yuan Wang, Pengfei Chen, and Ping Wang. 2017. Lightweight and Adaptive Service API Performance Monitoring in Highly Dynamic Cloud Environment. In *Proc. IEEE Int. Conf. Serv. Comput. (SCC)* (Honolulu, HI, USA). 35–43. doi:10.1109/SCC.2017.80
- [69] Jingjing Yang, Yuchun Guo, Yishuai Chen, and Yongxiang Zhao. 2023. Hi-rca: A hierarchy anomaly diagnosis framework based on causality and correlation analysis. *Appl. Sci.* 13, 22 (2023), 12126. doi:10.3390/app132212126
- [70] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinxing Sun, and Xiaoyun Li. 2021. MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments. In *Proc. Web Conf. 2021* (Ljubljana, Slovenia). 3087–3098. doi:10.1145/3442381.3449905
- [71] Guangba Yu, Zicheng Huang, and Pengfei Chen. 2023. TraceRank: Abnormal service localization with dis-aggregated end-to-end tracing data in cloud native systems. *J. Softw. Evol. Process* 35, 10 (2023). doi:10.1002/sm.2413
- [72] Yi Zhai, Junzhou Luo, and Jianru Liu. 2025. NRCAC: Non-Intrusive Microservice Root Cause Analysis Framework for Cloud Providers. In *roc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*. 1–10.
- [73] Shenglin Zhang, Zhongjie Pan, Heng Liu, Pengxiang Jin, Yongqian Sun, Qianyu Ouyang, Jiaju Wang, Xueying Jia, Yuzhi Zhang, Hui Yang, Yongqiang Zou, and Da Pei. 2023. Efficient and Robust Trace Anomaly Detection for Large-Scale Microservice Systems. In *Proc. 2023 IEEE 34th Int. Symp. Softw. Reliab. Eng. (ISSRE)* (Florence, Italy). 69–79. doi:10.1109/ISSRE59848.2023.00012