Hizmet Olarak İletişim Platformunda Mikroservis Etkileşim Tahmini
Microservice Interaction Prediction in Communication Platform as a Service

Kemal AKTAŞ
Bilgisayar Mühendisliği Bölümü
Bahçeşehir Üniversitesi
İstanbul, Türkiye
kemal.aktas@bahcesehir.edu.tr
kemal.aktas@orioninc.com

H. Hakan KILINÇ
Ar-Ge Merkezi
Orion Innovation Turkey
İstanbul, Türkiye
hakan.kilinc@orioninc.com

Nafiz ARICA
Bilgisayar Mühendisliği Bölümü
Bahçeşehir Üniversitesi
İstanbul, Türkiye
nafiz.arica@eng.bau.edu.tr

2022 30th Signal Processing and Communications Applications Conference (SIU) | 978-1-6654-5092-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SIU55565.2022.9864867

***Özetçe* —Telekomünikasyon platformlarında sistemin izlenmesi, olası ve gözlemlenen hatalara karşı hızlı bir şekilde aksiyon alınması ve sistemin sürekli çalışabilirliğinin sağlanması gerekir. Ancak yüksek sayıda kullanıcıya ve yoğun trafiğe sahip mikroservis mimari tabanlı platformlarda hata ayıklama ve problem adresleme işlemleri uzun zaman alabilir. Bu zorluk, mikroservis etkileşimlerinin tespit edilmesi gerekliliğinden kaynaklanır. Bu çalışmanın ilk aşamasında, mikroservis akış ve etkileşimleri sağlanmasının, hata ayıklama sürecinde operasyon ekiplerine önemli ölçüde zaman kazandırdığı gözlenmiştir. Bu doğrultuda mikroservis etkileşimleri tahmini yapan makine öğrenmesi tabanlı farklı modeller geliştirilmiş ve performansları kıyaslanmıştır. Geliştirilen modeller ile, mikroservis log verisi üzerinde log desenleri çıkarılır ve mevcut mikroservisin belirli bir anda etkileşime girdiği önceki ve sonraki mikroservisler tahmin edilerek söz konusu mikroservisin etkileşim haritası oluşturulur. Temel çağrı senaryosu ile çalışan mikroservis logları üzerinde yapılan deneylerde hata ayıklama sürecine olumlu katkı yapabilecek başarılı tahmin sonuçları elde edilmiştir.**

Anahtar Kelimeler—mikroservis, makine öğrenmesi, hata ayıklama, topoloji keşfi, etkileşim tahmini.

***Abstract*—In telecommunication platforms, it is necessary to monitor the system, take quick action against possible and observed errors, and ensure the continuous operability of the system. However, debugging and problem addressing processes can take a long time in microservice architecture-based platforms with high number of users and heavy traffic. This difficulty arises from the necessity of detecting microservice interactions. In the first phase of this study, it has been observed that providing microservice flows and interactions saves operational teams a significant time during the debugging process. In this scope, different machine learning-based models that predict microservice interactions have been developed and performances are compared . In these models, log patterns are extracted on microservice log data and the interaction map of the mentioned microservice is created by estimating the previous and next microservices that the current microservice interacts with at a certain moment. In the experiments on microservice logs working with the basic call scenario, successful estimation results were obtained that could contribute positively to the debugging process.**

Keywords—microservice, machine learning, debugging, topology discovery, interaction prediction.

I. GİRİŞ

Yüksek sayıda kullanıcıya sahip platformlar için en önemli gereksinim, kullanıcılara kesintisiz hizmet sağlayabilmektir. Bu durumda, sistem güncellemeleri, bakım çalışmaları v.b süreçlerin, canlı sistemlerde mümkün olan en kısa sürede tamamlanması gerekir. Diğer yandan, bu sistemler üzerinde meydana gelen hata ve problemlerin hızlı bir şekilde giderilmesi gerekir. Bu gereksinimlerin ortak noktası olan *hızlı aksiyon alma* olgusuna karşılık, yazılım dünyasında *konteyner* ve *mikroservis* kavramları gündeme gelmiş ve kabul görmüştür.

Mikroservisler, tek bir amaca hizmet eder ve sistem üzerindeki diğer mikroservislerden bağımsız olarak, kendi süreçlerini yürütürler [1]. Diğer yandan, bir mikroservis sisteminde, mikroservisler arası etkileşimler kaçınılmazdır. Trafiğin yüksek olduğu mikroservis sistemlerindeki dinamizm, geliştirme ve operasyon mühendisleri için hata ayıklama ve adresleme noktasında zorluklara neden olmaktadır, çünkü mikroservis etkileşimlerinin anlaşılması ve davranışlarının bilinmesi gerekmektedir. Bu zorluk, gün sonunda problemin ve hataların düzeltilmesi için gereken zamanın aşılmasına, hattâ hizmet kesintisinin bir sonucu olarak müşterinin kaybedilmesine de yol açabilir.

Bir teknoloji şirketinde geliştirilen, 4 farklı kıtada bulunan 5 farklı veri merkezinde çalışan ve yüksek sayıda mikroservis içeren bir telekomünikasyon platformu olan Communication Platform as a Service (CPaaS) verileri ışığında yürütülen bu çalışma iki yönlü katkı sunmaktadır:

- Kullanıcı senaryoları özelinde, mikroservisler arasındaki etkileşim ve topolojisinin bilinmesinin, hata giderilme sürecini kısaltabildiği anlaşılmıştır.
- Makine öğrenmesi yöntemleri kullanılarak mikroservis etkileşimleri tahmin modelleri geliştirilmiş ve bu modeller için performans kıyaslaması yapılmıştır.

II. İLGİLİ ÇALIŞMALAR

Monolitik yaklaşımdan mikroservis mimariye geçiş sürecinin doğurduğu karmaşık ve dinamik yapı, hata ayıklama ve anomali tespit etme açısından zorluklara neden olmaktadır. Bu durum, araştırmacıları bulut ve ayrık sistemler için geliştirilen yöntemleri mikroservis mimarisine evirmeye ve yeni yöntemler geliştirme noktasında çalışmalar yapmaya teşvik etmiştir.

X. Zhou ve arkadaşları [2], mikroservis sistemlerinde karşılaşılan tipik hatalar, endrüstride kullanılan mevcut hata ayıklama yöntemleri ve geliştiricilerin karşılaştığı zorluklar ile ilgili endüstriyel bir anket çalışması yapmıştır. Bu çalışmaya göre, izleme ve görselleştirme analiz teknikleri, geliştiricilerin mikroservis etkileşimlerini içeren çeşitli türdeki hataları bulmak için kullandığı yöntemlerdir.

Anomali tespiti, hata analizi ve adreslendirme problemlerine yönelik, log analizi, metrik bazlı ve paket izlemeye dayanan çeşitli çalışmalar yapılmıştır. [3]'de, kubernetes bileşenleri tarafından oluşturulan loglar ile bir referans model oluşturmaya dayalı bir yaklaşım sunulmuştur. T. Jia ve arkadaşları [4], servisler arasındaki topolojiyi, mikroservis etkileşimleri sırasındaki logların oluşma sıklığını kullanan bir model geliştirmiştir. Diğer yandan, Kahuna [5] yaklaşımına göre, sistemdeki tüm düğümler aynı şekilde davranmalıdır ve bir düğüm diğer tüm düğümlerden farklı bir şekilde davranıyorsa, hata ve anomalilerin potansiyel sorumlusudur.

Loud [6], CloudRanger [7] ve Microscope [8] gibi çalışmalar, anomali analizi ve hata adresleme süreçlerinde, sistem ve uygulama metriklerini kullanmaktadır. X-Trace [9] ve Mystery Machine [10] gibi yaklaşımlar ise sisteme gönderilen isteklerin baştan sona yolculuğunu temel alarak bir performans analizi ortaya çıkarmaktadır.

III. CPAAS MIKROSERVIS ETKILEŞIMLERI

A. CPaaS'a Genel Bakış

CPaaS, popüler bulut bilişim modellerinden biri olan "Platform hizmeti" (PaaS- Platform as a Service)'den faydalanarak, kullanıcılara ölçeklenebilir, mikroservis mimari tabanlı bir platform üzerinden iletişim servisleri sunan, bir yazılım platformudur. Diğer yandan, firmaların kendi ihtiyaçları doğrultusunda kullanabilmeleri amacıyla, VoIP API'leri de yine aynı platform üzerinden sunabilmektedir.

B. Problem Tanımı

CPaaS platformu için, hata adresleme süreci, bileşenlere göre değişkenlik göstermektedir. Tablo I'de CPaaS'ta yaşanan bazı örnek hata konuları ve bu hataların gözlemlendiği bileşenler verilmiştir. Ayrıca, hataların adreslenmesi ve giderilmesi için operasyon mühendislerinin harcadığı süreler *yüzde* cinsinden verilmiştir. Bu değerler, görevli mühendislerin tecrübelerine göre değişkenlik göstermekle beraber, CPaaS projesinin yönetildiği *Jira* verilerinden elde edilmiştir.

Tablo I'e göre, operasyon ekipleri için adreslenmesi en zorlu hatalar, *Yönlendirme* ve *Servisler* alanlarındadır ve hizmet işlevlerini içermeleri sebebiyle, direkt olarak kullanıcı senaryolarından raporlanmaktadır. Burada karşılaşılan zorluğun ana nedeni, mikroservis etkileşimlerinin tespit edilmesi gerekliliğidir. Örneğin temel bir arama senaryosu özelinde

TABLO I: ÖRNEK HATA BILEŞENLERI VE ADRESLENME/GIDERILME SÜRELERI

Hata Nedeni	**Hata Bileşeni ve Harcanan Süre**	
	Hata Bileşeni	**Süre**
Disk Kullanımı	Docker, GlusterFS, Elasticsearch	%10
Bellek Kullanımı	Sistem, Tüketim, Platform, Portal servisleri	%10
İletişim Ağı	Keepalived, Weavenet	%10
Yönlendirme	Kong, Nginx	%20
Konteyner	Docker, Harbor, Kubernetes	%11
Veritabanları	Postgresql, Redis, Mariadb, Minio v.b	%9
Servisler	Portal ve Tüketim servisleri	%22
Diğer	Ansible, Rabbitmq, Hazelcast v.b	%8

hata ayıklama çalışması için, etkileşime giren tüm mikroservislerin tanınması ve anlaşılması gerekir. Bu durum, yüksek sayıda mikroservis içeren bir çözüm için zaman açısından bir handikap yaratmaktadır. Diğer yandan, kullanıcı senaryolarına ilişkin hataların giderilmesi için gereken sürenin, senaryo ve mikroservisler arasındaki ilişkinin bilinmesi ile bağlantılı olduğu gözlemlenmiştir.

C. Kullanıcı Senaryoları İçin Hata Ayıklama Yaklaşımı

Operasyon ekiplerinin kullanıcı senaryoları üzerinde gözlemlenen hataların giderilmesine ilişkin yaklaşımına göre, *Kubernetes* sistemi üzerindeki temel gereksinimler [11] kontrol edilmekte ve gerekli aksiyonlar alınmaktadır. *Kubernetes* sisteminin düzgün bir şekilde çalıştığından emin olunduktan sonraki adım, kullanıcı *REST* isteğinin incelenmesi ve *Kong ağ geçidi* üzerinde kontrol edilmesidir. Yönlendirme kurallarına göre, isteğin iletildiği ilk mikroservisin loglarından başlayarak, diğer mikroservisler ile olan etkileşimleri göz önüne alınarak zincirleme bir log analizi yapılmaktadır. Bu yaklaşımdaki en büyük handikap, mikroservisler arasındaki etkileşimin tespit edilmesi için harcanan süredir. Tablo I'de *Yönlendirme* ve *Servisler* için verilen değerlerin harcanan en yüksek sürelere sahip olması, bu handikapı kanıtlar niteliktedir.

D. Kullanıcı Senaryosu Özelinde Mikroservisler Arası Etkileşimler

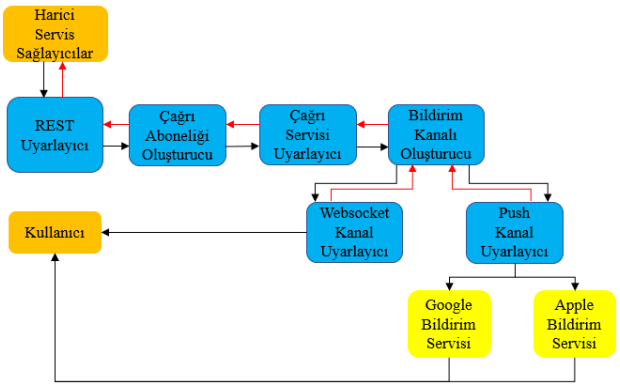
CPaaS platformu üzerinde, temel bir çağrı hizmeti üç temel aşamadan oluşmaktadır. Buna göre, kullanıcıların haberleşebilmesi için bir iletişim kanalı kurulmakta, çağrı servisi abonelikleri oluşturulmakta ve son aşamada çağrı bildirimleri sağlanmaktadır. Bu aşamalardan herhangi birinde hata gerçekleşmesi durumunda, temel çağrı hizmeti kullanım dışı kalmaktadır. Başka bir deyişle, etkileşim halindeki mikroservislerden birinin problem yaşadığı bir durumda, çağrı hizmeti tümüyle etkilenmektedir. Şekil 1, çağrı bildirim gönderimi sırasındaki mikroservis etkileşimlerini temsil etmektedir.

Mikroservisler arası etkileşimin bilinmesi, kullanıcıların yaşadığı hataların giderilebilmesi açısından son derece önemlidir. CPaaS operasyon ekipleri ile yapılan toplantılarda, hata ayıklama görevlilerine, kullanıcı senaryolarına ilişkin beklenen akış ve etkileşimler sağlandığında, hata ayıklama ve giderilme sürelerinin önemli ölçüde kısaldığı bilgisi alınmıştır.

IV. MAKINE ÖĞRENMESI ILE MIKROSERVISLER ARASI ETKILEŞIMLERIN TAHMIN EDILMESI

A. Veri Toplama

CPaaS platformu üzerinde *ELK yığını (Elasticsearch, Kibana & Logstash stack)* ile elde edilen veriler ortalama olarak,



Şekil 1: CPaaS çağrı bildirim hizmeti akış diyagramı.

aylık milyonlar mertebesindedir. Konteyner ve sistem metrikleri, zaman, çalışan kod sınıfı ve metotları (örneğin Java kodu içindeki sınıf ve metotlar), kullanıcı, kayıt seviyesi, mikroservislerin etkileşim sırasında gönderdiği ve aldığı mesajlar, kod çalışması esnasında üretilen fonksiyonel iletiler gibi bilgileri içeren bu veriler, mikroservis etkileşimlerinin ve topolojinin tahmin edilebilmesi için kullanılabilmektedir.

B. Veri İşleme

Veri işleme aşaması, temel olarak verilerin temizlenmesi, özniteliklerin belirlenmesi ve etiket kodlaması işlemlerini kapsamaktadır.

Bölüm IV-A'da elde edilen veriler, mikroservis etkileşimlerine dair kayıtlar içermesinin yanı sıra, sistem metrikleri ya da mikroservis kaynak kodlarının çalışmasına yönelik bilgiler de barındırmaktadır. Bu veriler, mikroservis etkileşimleri açısından anlamlı değildir. Bu yüzden, Algoritma 1 ile "*X aksiyonuna sahip Y tipindeki ileti Z düğümüne senkronize bir şekilde gönderiliyor*" desenindeki kayıtların alınması ve makine öğrenmesi modellerinde kullanılacak baz verilerin oluşturulması amaçlanmıştır.

Desen tanıma süreci sonrası elde edilen metin tipindeki verilerin makine öğrenmesi algoritmalarında kullanılabilmesi için, sayısal değerlere dönüştürülmesi gerekmektedir.

C. Mikroservisler Arası Etkileşim Tahmini Yöntemi

Bu çalışmada, sınırlı bir veri örneğinde makine öğrenimi modellerini değerlendirmek için kullanılan bir yeniden örnekleme prosedürü olan k-kat çapraz doğrulama tekniği kullanılmıştır. Bu teknik, sınırlı bir veri kümesini örtüşmeyen k adet kata böler. Bu katlarından biri test seti olarak kullanılırken, diğer tüm katlar eğitim veri seti olarak kullanılır. Toplam k model değerlendirilir ve ortalama performans rapor edilir.

Mikroservis etkileşimleri tahmini sürecinde, mesaj tipi, senaryo bilgisi, servis aksiyon bilgisi gibi kategorik veriler ile, mevcut mikroservisin yöneleceği sınıflar tahmin edilmektedir. Burada, her sınıf platform üzerindeki başka bir mikroservise karşılık gelmektedir. Tahmin yöntemi olarak, en popüler sınıflandırma algoritmalarından olan *lojistik regresyon*, *destek vektör makinesi*, *karar ağacı*, *rassal orman* sınıflandırma algoritmaları kullanılmış ve model performansları karşılaştırılmıştır. Ayrıca, *çok katmanlı algılayıcı (Multilayer Perceptron-MLP)* algoritması ile elde edilen sonuçlar da performans karşılaştırma ve değerlendirme sürecine dahil edilmiştir.

Algoritma 1: Beklenen Desene Sahip Mesajların Veri Setinden Ayrıştırılması

for *Her bir mikroservis* **do**
  input : *Kibana*'dan elde edilen mikroservis log dosyası, *ms.csv*.
  for *Dikey yönde veri sayısı, i=0,1,..,uzunluk(ms.csv).* **do**
    Senaryo adını *senaryo* dizisine ekle.
    Mikroservis adını *mevcut düğüm* dizisine ekle.
    Mesaj satırını oku.
    Mesaj metnini parçala ve bir *msg* nesnesine ata.
    msg[3]'ü *servis aksiyonu* dizisine ekle.
    msg[4]'ü *mesaj tipi* dizisine ekle.
    if *"Senkron gönderme" deseni* ***in*** *Mesaj* **then**
      msg nesnesini $mesaj$ dizisine ekle.
      if *"Cevap"* ***in*** $mesaj$ **then**
        mesaj[6]'yı *önceki düğüm* dizisine ekle.
        "-" değerini *sonraki düğüm* dizisine ekle.
      end
      else if *"Talep" veya "Bildirim"* ***in*** $mesaj$ **then**
        mesaj[6]'yı *sonraki düğüm* dizisine ekle.
        "-" değerini *önceki düğüm* dizisine ekle.
      end
      else
        "-" değerini *önceki düğüm* dizisine ekle.
        "-" değerini *sonraki düğüm* dizisine ekle.
      end
    end
  end
  senaryo, servis aksiyonu, mesaj tipi, mevcut düğüm, önceki düğüm, sonraki düğüm listeleri ile bir veri seti oluştur.
end

V. Deneyler

A. Test Verilerinin Oluşturulması

Bu çalışma, *temel çağrı hizmeti* senaryosu sırasında etkileşime giren mikroservis verileri kullanılarak yapılan deneyler ışığında oluşturulmuştur. Temel çağrı hizmeti sırasında, CPaaS platformu üzerinde 6 adet tüketim mikroservisi etkileşime girmektedir. Senaryo, CPaaS test-otomasyon aracı ile art arda 10 kez çalıştırılmış ve bu şekilde 15310 adet veri satırı elde edilmiştir.

B. Özniteliklerin Çıkarılması

Test senaryolarının çalıştırılması sırasında etkileşime giren mikroservislerin verileri, Kibana üzerinde uygun zamanlama filtreleri kullanılarak elde edilmiş ve python pandas kütüphanesine aktarılmıştır. Elde edilen geniş çaplı veriden, etkileşim açısından önemli sayılabilecek baz verilerin elde edilebilmesi için Algoritma 1 kullanılmıştır. Örneğin, *çağrı uyarlayıcı* servisi kayıtlarında yakalanan "*çağrı oluşturucuya bir çağrı başlatma yanıtı gönderiyorum*" mesajı, Algoritma 1'in uygulanmasıyla

TABLO II: DESEN TANIMA SONRASI ELDE EDILEN BAZ VERI ÖRNEĞI

Öznitelik Adı	Örnek Değer
Senaryo	çağrı
Servis Aksiyonu	çağrı başlat
Mesaj Tipi	yanıt
Mevcut Düğüm	çağrı uyarlayıcı
Önceki Düğüm	çağrı oluşturucu
Sonraki Düğüm	-

Tablo II'deki şekli almıştır. Bu süreç, her bir mikroservis verisi üzerinde uygulanmış olup, Tablo II formatında bir veri seti elde edilmiştir. Bu veri setinde, *Önceki Düğüm* ve *Sonraki Düğüm* sütunları hedef sınıflar olarak etiketlenmiştir.

Metin tipindeki veri sütunları kategorik tipe evrilmiş ve sonrasında popüler bir yöntem olan etiket kodlaması (label encoding) yöntemi ile sayısal değerlere dönüştürülmüştür. Bu işlem için, python scikit-learn kütüphanesi kullanılmıştır. Etiket kodlaması sonucu, veri setindeki her bir sütunun sahip olduğu farklı veri sayısı Tablo III'de verilmiştir. Örneğin *Servis Aksiyonu* özniteliği 18 farklı değere sahiptir. *Senaryo* özniteliğinin tek değere sahip olmasının nedeni, deneylerde sadece temel çağrı hizmeti verilerinin kullanılmasıdır.

TABLO III: ÖZNITELIKLERIN İÇERDIĞI FARKLI DEĞER SAYILARI

Değişken Adı	Farklı Değer Sayısı
Senaryo	1
Servis Aksiyonu	18
Mesaj Tipi	3
Mevcut Düğüm	6

C. Makine Öğrenmesi Performans Analizi

Mikroservisler arası etkileşim tahmin modellerinin performans analizinde *k-kat çapraz doğrulama* tekniği kullanılmış ve k = 5 değeri için testler yapılmıştır. Bölüm IV-C'de belirtilen her bir algoritma için tekrarlanan testlerde ortalama doğruluk değeri hesaplanmıştır. Oluşturulan modeller ile elde edilen performans değerleri Tablo IV'te verilmiştir. *Önceki Düğüm*'lerin tahmin edilmesi sonucunda elde edilen ortalama doğruluk değerleri %79-92 arasında değişkenlik göstermektedir. Diğer yandan *Sonraki Düğüm*'lerin değerlendirmesinde elde edilen ortalama başarı %68-89 aralığındadır. Test edilen algoritmalar arasında en yüksek başarı, *Önceki Düğüm* tahminlerinde %92 ve *Sonraki Düğüm* tahminlerinde %89 olarak, *çok katmanlı algılayıcı* algoritması ile oluşturulan model ile elde edilmiştir.

TABLO IV: MIKROSERVIS ETKILEŞIMLERI TAHMINI DENEYSEL SONUÇLARI

Yöntem	Ortalama Doğruluk Değeri	
	Önceki Düğüm	Sonraki Düğüm
Lojistik Regresyon	%79	%68
Destek Vektör Makinesi	%85	%75
Karar Ağacı	%88	%85
Rassal Orman	%87	%87
Çok Katmanlı Algılayıcı	%92	%89

VI. SONUÇ

Yüksek sayıda kullanıcı ve yoğun trafiğe sahip olan, mikroservis mimari tabanlı bir telekomünikasyon platformu olan CPaaS için, hata adresleme süreci, sistem bileşenlerine göre değişkenlik gösterebilmektedir. Yapılan incelemelerde, operasyon ekiplerinin hata adresleme sürecinde en çok süreyi, kullanıcı senaryolarında yaşanan, yönlendirme ve servis bileşenlerine işaret eden hataların giderilmesi için harcadığı gözlemlenmiştir. Diğer yandan, mikroservisler arası etkileşim ve sistem topolojisinin bilinmesinin, hata giderme sürelerini kısalttığı sonucuna varılmıştır.

CPaaS platformundan elde edilen veriler, etkileşim ve topoloji keşfi amacıyla kullanılabilmektedir. K-kat çapraz doğrulama ve 5 farklı sınıflandırma algoritması kullanılarak, mikroservisler arası etkileşimler tahmin edilmiş ve model performansları karşılaştırılmıştır.

Mikroservis etkileşimlerinin keşfedilmesinin, hataların adreslenmesi üzerindeki önemini vurgulayan ve mikroservis etkileşimleri tahmini yapan bu çalışmanın bir sonraki adımı, farklı kullanıcı senaryolarıyla zenginleştirilen mikroservis verilerini kullanarak model performansını arttırmak, bir otomatik topoloji ve etkileşim keşfi algoritması geliştirmektir.

KAYNAKLAR

[1] G. Blinowski, A. Ojdowska and A. Przybyłek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," in IEEE Access, vol. 10, pp. 20357-20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[2] X. Zhou et al., "Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study," in IEEE Transactions on Software Engineering, vol. 47, no. 2, pp. 243-260, 1 Feb. 2021, doi: 10.1109/TSE.2018.2887384.

[3] J. Xu, P. Chen, L. Yang, F. Meng and P. Wang, "LogDC: Problem Diagnosis for Declartively-Deployed Cloud Applications with Log," 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE), 2017, pp. 282-287, doi: 10.1109/ICEBE.2017.52.

[4] T. Jia, P. Chen, L. Yang, Y. Li, F. Meng and J. Xu, "An Approach for Anomaly Diagnosis Based on Hybrid Graph Model with Logs for Distributed Services," 2017 IEEE International Conference on Web Services (ICWS), 2017, pp. 25-32, doi: 10.1109/ICWS.2017.12.

[5] Jiaqi Tan, Xinghao Pan, E. Marinelli, S. Kavulya, R. Gandhi and P. Narasimhan, "Kahuna: Problem diagnosis for Mapreduce-based cloud computing environments," 2010 IEEE Network Operations and Management Symposium-NOMS 2010, 2010, pp. 112-119, doi: 10.1109/NOMS.2010.5488446.

[6] L. Mariani, C. Monni, M. Pezzé, O. Riganelli and R. Xin, "Localizing Faults in Cloud Systems," 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), 2018, pp. 262-273, doi: 10.1109/ICST.2018.00034.

[7] P. Wang et al., "CloudRanger: Root Cause Identification for Cloud Native Systems," 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2018, pp. 492-502, doi: 10.1109/CCGRID.2018.00076.

[8] Lin J., Chen P., Zheng Z. (2018) Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments. In: Pahl C., Vukovic M., Yin J., Yu Q. (eds) Service-Oriented Computing. ICSOC 2018. Lecture Notes in Computer Science, vol 11236. Springer, Cham.

[9] Rodrigo Fonseca, George Porter, Randy H. Katz, Scott Shenker, and Ion Stoica. 2007. X-trace: a pervasive network tracing framework. In Proceedings of the 4th USENIX conference on Networked systems design and implementation (NSDI'07). USENIX Association, USA, 20.

[10] Chow, Michael, David Meisner, Jason Flinn, Daniel Peek and Thomas F. Wenisch. "The Mystery Machine: End-to-end Performance Analysis of Large-scale Internet Services." OSDI (2014).

[11] Vayghan, Leila and Saied, Mohamed and Toeroe, Maria and Khendek, Ferhat. (2019). Kubernetes as an Availability Manager for Microservice Applications.