Sock Shopにおける異常となる応答時間の予測CPU使用量を閾値としたアラートストームの防止

疋田 辰起[1]　大野 有樹[2]　串田 高幸[1]

概要：マイクロサービスで動作している EC サイトでは，アラートストームが発生することにより，管理者はこれの解読に時間を奪われ障害対応の遅れに繋がり，多大な損失を被る．デモアプリケーションとして Sock Shop がある．本稿では，この Sock Shop を対象にアラートの条件を提案しアラートストームを防止することを目標とする．アラートストームの基準は 30 分に 50 件とする．基礎実験では，CPU 使用率が 80%以上をアラート生成の条件とし，30 分間毎秒のユーザ数 200 の負荷試験をした．結果，77 件のアラートが生成され，アラートストームが発生した．よって，CPU 使用率の閾値によるアラートは適さないとした．提案では，各サービスの CPU 使用量，応答時間を用いて応答時間が異常となる予測 CPU 使用量を累乗近似から求める．この近似曲線は負荷による CPU 使用量と応答時間の増加量が一定でないため累乗近似を用いる．CPU 使用量がこの予測値に対して 80%以上をアラートを生成する条件とした．異常となる応答時間は 3,000(ms) とした．実験では，5 秒に 1 人ユーザを増加させ毎秒の最大ユーザ数 400 による負荷を 30 分間発生させた．結果は，アラートストームの基準とした 50 件未満である 33 件のアラートが生成され，基礎実験の 77 件より 44 件減少した．

1. はじめに

背景

近年，急速に進化するアプリケーションの開発と展開に対応するために企業では従来のモノリシックアーキテクチャからマイクロサービスアーキテクチャへの移行が進んでいる [1]．マイクロサービスのきめ細かいモジュール性と独立して展開可能でスケーラブルな性質により，マイクロサービスアーキテクチャはクラウドのリソースを活用する上で大きな可能性を示している [2].

Sock Shop[*1]は，靴下の EC サイトをシミュレートするマイクロサービスのデモンストレーションアプリである．これは，マイクロサービスのデモンストレーションとテストを支援するために設計された広く使用されているマイクロサービスベンチマークである [3]．このようなベンチマークは新しいマイクロサービスアーキテクチャやプログラミングモデルのパフォーマンスへの影響を調査することや，マイクロサービスの開発手法，自動化テクノロジの有効性と制限の比較に有用である [4].

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学大学院 バイオ・情報メディア研究科 コンピュータサイエンス専攻
〒 192–0982 東京都八王子市片倉町 1404–1
[*1] https://microservices-demo.github.io/

マイクロサービスアーキテクチャにより，拡張性，柔軟性，管理性に優れたシステムの開発が可能になる．ただし，そのようなアーキテクチャはサービスの粒度が高く分散されているため，実行時にアプリケーションを監視するという課題を悪化させる [5]．これは，近年注目されている研究の方向性であり，これらの研究者の中には，マイクロサービス監視の研究に重点を置いている [6]．監視システムはシステムの稼働状況をリアルタイムで把握するために，一般的に各マイクロサービスコンポーネントから CPU 使用率や応答時間を収集する [7].

障害が発生した場合には，複雑で相互依存するマイクロサービスにより多数の異常が検出され，多くのアラートが発せられる [8]．ただし，サービス障害の発生は，1 つまたは 2 つのアラートではなく，マイクロサービスは多数のコンポーネントで構成されており，異なるコンポーネントが相互に影響しやすく多くのアラートが生成される [9]．Huawei Cloud では 5 時間の間に合計 2,751 件のアラートが生成された [10]．また，アラートが生成されるたびに管理者は時間をかけてその障害が本物かどうかその原因を調査し，障害が検証されなかった場合，管理者は誤ったアラートに対して費やした時間を節約できた可能性がある [11].

障害の解決が遅れることでサービスの運営元は多大な損失を被ることがある．ダウンタイムにより Cloud Foundry

2 では 1 時間あたりの収益が$336,000 減少すると指摘しており，オンライン決済システムの Paypal では 1 時間あたり $225,000 の減少を経験した [12]．また，Microsoft の電子メールサービスでは，毎日最大 100 億件の電子メールがクラウドに配信されており，配信遅延は重要な SLA(Service Level Agreement) の 1 つとなる [13].

課題

マイクロサービスアーキテクチャを採用している EC サイトでは，セールや販売開始時のスパイクアクセスによる障害で大量のアラートが生成される [9]．これにより，管理者は生成されたアラートを無視や誤解をし，障害解決までの時間が長くなり，対象の EC を運営する企業は大きな損失を被る [14]．このように，アラートが短時間に大量に生成されることをアラートストームと呼ぶ．アラートの数が 1 時間に 100 件を超えた場合，それをアラートストームの発生とする [10].

基礎実験

CPU の使用率が 80%以上[*2]をアラートを生成する条件とし，毎秒のアクセスユーザ数 200 で 30 分間負荷試験を行った．図 1 にアラートストームの発生時のメールを示す．front-end について 74 件，carts について 3 件のアラートが生成された．これは 30 分間におけるアラートストームの基準を 24 件超えている．表 1 に front-end と carts に関しての平均応答時間とメール件数を示す．front-end について 74 件のアラートがされているが，平均応答時間は 20.8(ms) である．CPU 使用率が高負荷となっているが，サービスは正常に動作している．よって，CPU 使用率の閾値によるアラートは適さないとする．

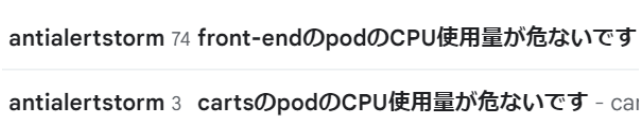

図 1 アラートストーム発生時のメール

表 1 アラートストーム発生時の応答時間とメール件数

サービス名	平均応答時間 (ms)	メール件数
front-end	20.8	74
carts	3,580.9	3

各章の概要

2 章では，本稿の関連研究を述べる．3 章では，課題に対しての提案について説明する．4 章では，実装について説明する．5 章では，評価実験とその分析について述べる．6 章では，提案について議論をする．7 章では，全体のまとめを述べる．

[*2] https://www.ibm.com/docs/ja/iad/7.2.1?topic=usage-cpu

2. 関連研究

故障分析における管理者の負担を軽減するために，同じ故障によってトリガーされる可能性のあるアラートをリンクするための動的グラフニューラルネットワークベースのアプローチの研究がある [15]．これは生成されたアラートに対して依存関係を導き冗長なアラートをリンクする提案であり，生成されるアラートの数については変化しない．

アラートを削減しその根本原因を特定するために大量のアラートを効果的に圧縮し，根本原因の特定の効率を向上できるデータ駆動型の教師なしアラート分析フレームワークについての研究がある [16]．この提案では冗長なアラートを削減できるとしているが，削減されるアラートの適否については議論されていない．

教師あり学習に基づく新しいアプローチを通じてアラートのセマンティック表現と動作表現を集約し，アラート間の相関関係を判断する研究がある [17]．これは，新しく報告されたアラートをオンラインで要約することができ，生成されたアラートに対して有効な提案であるが，元の生成されるアラートの数は減少しない．

MIC ベースの相関アルゴリズムを使用して，アラートをより高いレベルにグループ化し管理者に通知するためのより有益なハイパーアラートについての研究がある [18]．応答時間の偏差によってサービスの問題を認識し，アラートを生成している．マイクロサービスでは応答時間のみといった単一のメトリクスでなく CPU 使用量を用いるといった複数のメトリクスによるアラートの生成がされるべきである．

3. 提案

基礎実験

Sock Shop に負荷を発生させた際の各サービス orders と carts について，CPU 使用量の増加による応答時間の変化を調査する実験を行った．毎秒のユーザ数は 100，150，200，250，300，350， 400 で負荷試験のシナリオは Sock Shop の開発者が提供しいているシナリオ[*3]を使用する．

図 2 に carts と orders においての毎秒のユーザ数ごとの平均 CPU 使用量を示す．両サービス共に，ユーザ数 200 から CPU 使用量の上昇が 200 以前と比べ緩やかになっている．

図 3 に carts と orders においての毎秒のユーザ数ごとの平均応答時間を示す．両サービス共に，ユーザ数の増加に伴い応答時間も増加している．CPU 使用量とは異なりユーザ数 200 からは上昇量が 200 以前より増加している．

[*3] https://github.com/microservices-demo/load-test

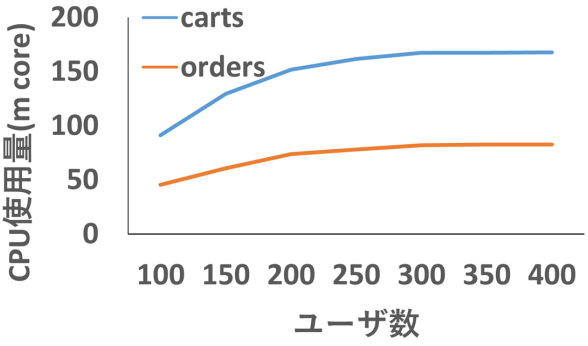

図 2 ユーザ数ごとの carts と orders の平均 CPU 使用率

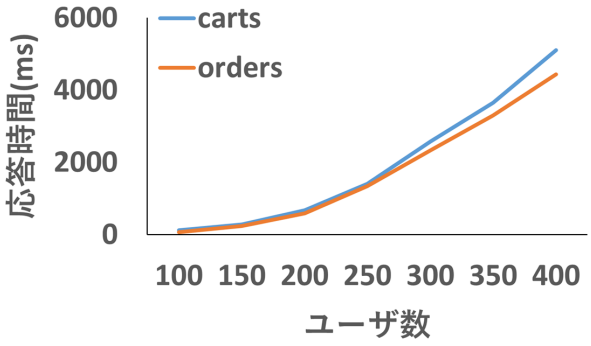

図 3 ユーザ数ごとの carts と orders の平均応答時間

図 4 に carts においてユーザ数ごとの平均 CPU 使用量と平均応答時間の散布図と近似曲線を示す．CPU 使用量と応答時間は変化量が一定ではないため，近似曲線には累乗近似を用いた．近似曲線は $y = 51.41x^{0.15}$ で表されており，決定係数 R^2 は 0.81 であった．

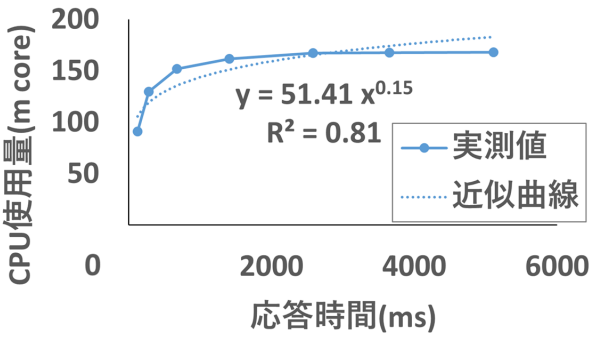

図 4 carts の CPU 使用量と応答時間の近似曲線

図 5 に orders においてユーザ数ごとの平均 CPU 使用量と平均応答時間の散布図と近似曲線を示す．carts と同様に近似曲線は累乗近似を用いている．近似曲線は $y = 26.91x^{0.14}$ で表されており，決定係数 R^2 は 0.90 であった．

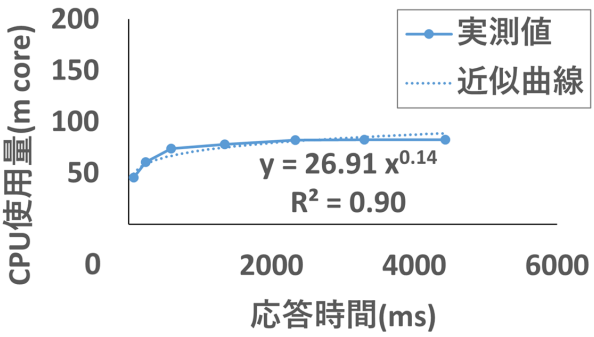

図 5 orders の CPU 使用量と応答時間の近似曲線

以上のことから，ユーザ数 100 から 200 までは負荷の上昇により CPU 使用量と応答時間は増加し，200 以降は以前と比べ緩やかに増加することが分かる．また，CPU 使用量と応答時間の近似曲線の決定係数が carts は 0.81，orders は 0.90 であることより，応答時間が異常となる予測 CPU 使用量を求めることに使用できるとする．提案ではこれらを用いてアラートを生成する条件を設定する．

提案方式

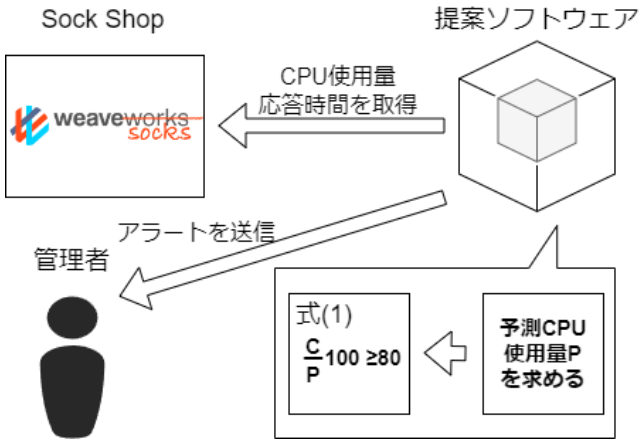

図 6 提案の概要

アラートストームを防ぐためにアラートの適切な条件を提案する．異常とする応答時間を 3000(ms)[*4]とする．各サービス間の応答時間が 3000(ms) となる予測 CPU 使用量を，毎秒取得する各サービスの CPU 使用量と応答時間による近似曲線から求める．近似曲線は累乗近似を使用した．これは，CPU 使用量と応答時間の変化量が一定ではないため累乗近似を用いる．この各サービスの予測 CPU 使用量に対して CPU 使用量が 80%以上[*2]をアラートを生成する条件とする．これはアラートを受け取った管理者が，応答時間が異常となる前に対処をする時間を確保するため

[*4] https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/

である．アラートはメールで管理者に通知される．

提案の概要を図 6 に示す．提案ソフトウェアでは，Sock Shop から取得した各サービスの CPU 使用量と応答時間の近似曲線を求める．これを用いて応答時間が 3000(ms) となる予測 CPU 使用量 P を求める．式 (1) より，P に対して，取得した各サービスの CPU 使用量 C の値が 80%以上の際にアラートを行う．

$$\frac{C}{P}100 \geq 80 \tag{1}$$

ユースケース・シナリオ

本稿ではファッション EC サイトをユースケースとして想定している．ファッション EC サイトである ZOZOTOWN[*5]では，セールイベントや新商品の販売開始時に短時間で多くのユーザがアクセスすることがある．このスパイクアクセス時に，提案ソフトウェアによりアラートストームとならない量のアラートを生成し，管理者の適切な対応を可能にする．

4. 実装

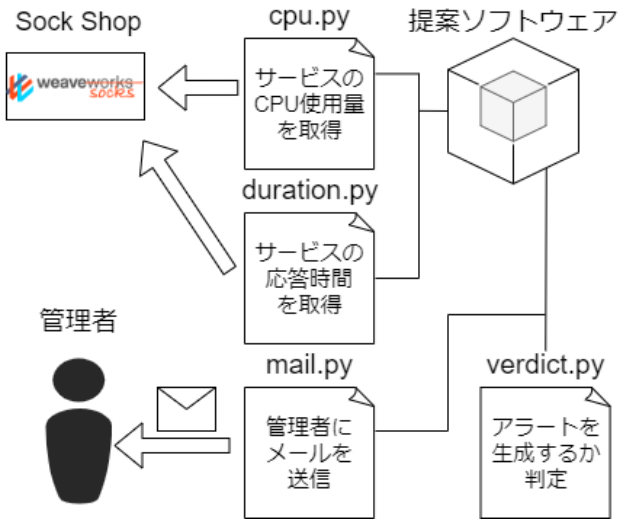

図 7 実装の概要

実装の概要を図 7 に示す．提案ソフトウェアは *cpu.py*，*duration.py*，*mail.py*，*verdict.py* の構成となっている．*cpu.py* にでは，kubectl top コマンドにより CPU 使用量を取得する．*duration.py* では，Istio Jaeger の api から応答時間を取得する．*verdict.py* で提案方式が実装されており，取得した CPU 使用量と応答時間から SciPy[*6]の curvefit 関数を用いて累乗近似による近似曲線の式を求める．これにより，応答時間が 3000(ms) となる際の予測 CPU 使用量を求め，CPU 使用量が予測 CPU 使用量の 80%以上となるかを判定する．判定の結果アラートを生成する際は *mail.py* により管理者にアラートのメールが送信される．

*5 https://techblog.zozo.com/entry/zozotown-load-test
*6 https://scipy.org/

5. 評価実験

図 8 に示す Sock Shop に対して locust で負荷試験を行いスパイクアクセスを発生させる．5 秒に 1 人ユーザ数を増加させ毎秒の最大ユーザ数を 400 で前述のシナリオによる負荷を 30 分間発生させた．その際，提案ソフトウェアにより生成されたアラートについて，アラートストームが発生していないかを評価する．アラートストーム発生の基準は課題で述べた 1 時間で 100 件より，負荷試験の時間に合わせた 30 分に 50 件とする．

実験環境

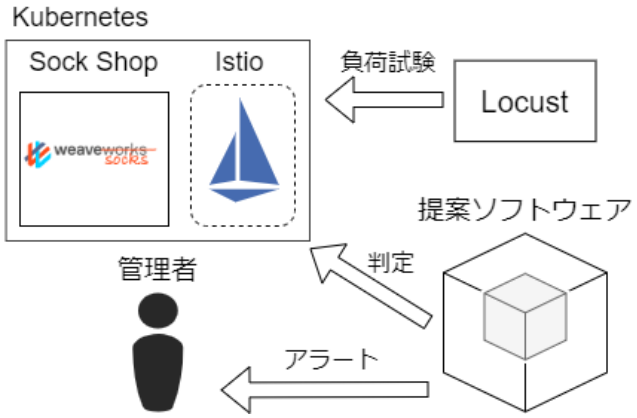

図 8 実験環境の概要

実験環境の概要を図 8 に示す．本稿ではサービスメッシュである Istio を導入した Sock Shop で実験を行う．Sock Shop はワーカーノード 2 つに Pod を分散して構築されている．Pod を分散配置することでより多くのリクエストを処理可能となる．提案ソフトウェアをマスターノードに実装し，提案方式によって管理者にメールでアラートがされる．Sock Shop に対して，負荷試験ツールである locust を用いて負荷を発生させ実験を行う．locust はマスターノード 1 つワーカーノード 2 つのクラスターで構築されている．

負荷試験のシナリオは Sock Shop の開発者が提供しているシナリオ[*3]を使用する．開発者が提供しているシナリオは．(1) から (7) となっている．ユーザは最初に Sock Shop のフロントページにアクセスをし，アカウント登録をする．その後カタログページにアクセスをし，商品ページに移動する．商品をカートに追加し，決済ページに移動し購入を確定する．

(1) フロントページにアクセス
(2) アカウント登録
(3) カタログページにアクセス
(4) ランダムな商品ページにアクセス
(5) 商品をカートに追加
(6) 決済ページ

(7) 購入

実験結果と分析

図 9 に受信したアラートメールを示す．30 分間に carts に関するアラートが 18 件，orders に関するアラートが 15 件であり，合計で 33 件のアラートを受信した．これはアラートストームの基準である 30 分間に 50 件の約 67%の件数であり，基礎実験の 77 件より 44 件少ない件数である．本稿の提案では，条件を満たしたときにアラートが生成されるため，アラートの内容は冗長なものとなっている．

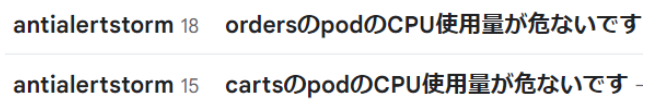

図 9 受信したアラートメール

6. 議論

実験結果より，アラートが合計 17 件であった．アラートは提案の条件を満たしたときに生成されるものであるため，アラートメールの内容は冗長なものとなっている．冗長なアラートはアラートの有効性を下げてしまうため改善する必要がある．管理者へアラートの表示方法をメールやメッセージではなく，異常，平常の表示をモニターで行うことで改善が可能である．

ユースケースとしてスパイクアクセス時のアラートを想定している．スパイクアクセス時では短時間でサービスに障害が発生する状況となる．よって，アラートを生成し管理者に適切な対処を促すことでは間に合わない．これを解決するために改善する必要がある．提案では異常と判断したサービスのアラートを生成することができる．よって，サービスに対して Pod のリソースをオートスケールするような障害の原因特定から修正まで行うことで解決が可能である．

提案の基礎実験より，ユーザ数の増加による CPU 使用量の増加について，ユーザ数が 200 からは増加が緩やかになっている．対して応答時間は，増加量が増している．これは，負荷によりサービスがリクエストを処理しきれない状態となっているからである．そのため，新たに受信したリクエストが処理待ち状態になる．これにより，CPU 使用量の増加が緩やかになり，応答時間は増加し続ける状況が発生している．この状況では CPU 使用量が収束するような値になるため，異なる近似式が成り立つ．CPU 使用量の収束値と累乗近似からアラート生成の条件とすることでこの状況に対応可能である．

7. おわりに

アラートストームが発生した際，管理者の対応が遅れダウンタイムが長くなり大きな損失となってしまう．本稿ではアラートストームとならないアラートを生成する条件を提案した．実験を行った結果，30 分間で生成されたアラートの数は 17 件であった．これは基礎実験より 60 件少く，アラートストームの基準未満の件数であり，提案によるアラートストームの防止がされた．提案の改善として，提案で生成されるアラートは冗長であるため，アラートの表示方法の変更が必要である．また，アラートを生成する際の提案の条件で異常となるサービスを特定し，オートスケールの実装が可能である．

参考文献

[1] Velepucha, V. and Flores, P.: A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges, *IEEE Access*, Vol. 11, pp. 88339–88358 (online), DOI: 10.1109/ACCESS.2023.3305687 (2023).

[2] Pallewatta, S., Kostakos, V. and Buyya, R.: Placement of Microservices-Based IoT Applications in Fog Computing: A Taxonomy and Future Directions, *ACM Comput. Surv.*, Vol. 55, No. 14s (online), DOI: 10.1145/3592598 (2023).

[3] Lin, J., Chen, P. and Zheng, Z.: Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments, *Service-Oriented Computing* (Pahl, C., Vukovic, M., Yin, J. and Yu, Q., eds.), Cham, Springer International Publishing, pp. 3–20 (2018).

[4] Aderaldo, C. M., Mendonça, N. C., Pahl, C. and Jamshidi, P.: Benchmark Requirements for Microservices Architecture Research, *2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*, pp. 8–13 (online), DOI: 10.1109/ECASE.2017.4 (2017).

[5] Castro, J., Laranjeiro, N. and Vieira, M.: Exploring Logic Scoring of Preference for DoS Attack Detection in Microservice Applications, *2023 IEEE International Conference on Web Services (ICWS)*, pp. 573–584 (online), DOI: 10.1109/ICWS60048.2023.00076 (2023).

[6] Su, X., Tolba, A., Lu, Y., Tan, L., Wang, J. and Zhang, P.: An Attention Mechanism-Based Microservice Placement Scheme for On-Star Edge Computing Nodes, *IEEE Access*, Vol. 11, pp. 114341–114351 (online), DOI: 10.1109/ACCESS.2023.3324222 (2023).

[7] Wang, L., Zhao, N., Chen, J., Li, P., Zhang, W. and Sui, K.: Root-Cause Metric Location for Microservice Systems via Log Anomaly Detection, *2020 IEEE International Conference on Web Services (ICWS)*, pp. 142–150 (online), DOI: 10.1109/ICWS49710.2020.00026 (2020).

[8] Behera, A., Panigrahi, C. R., Behera, S., Patel, R. and Bera, S.: trACE-Anomaly Correlation Engine for Tracing the Root Cause on a Cloud based Microservice Architecture, *Computación y Sistemas*, Vol. 27, No. 3 (2023).

[9] Zhao, N., Chen, J., Peng, X., Wang, H., Wu, X., Zhang, Y., Chen, Z., Zheng, X., Nie, X., Wang, G., Wu, Y., Zhou, F., Zhang, W., Sui, K. and Pei, D.: Understanding and Handling Alert Storm for Online Service Systems, *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '20, New York, NY, USA, Association for Computing Machinery, p. 162–171

(online), DOI: 10.1145/3377813.3381363 (2020).

[10] Yang, T., Shen, J., Su, Y., Ren, X., Yang, Y. and Lyu, M. R.: Characterizing and Mitigating Anti-patterns of Alerts in Industrial Cloud Systems, *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 393–401 (online), DOI: 10.1109/DSN53405.2022.00047 (2022).

[11] Nobre, J., Pires, E. S. and Reis, A.: Anomaly Detection in Microservice-Based Systems, *Applied Sciences*, Vol. 13, No. 13, p. 7891 (2023).

[12] Endo, P. T., Rodrigues, M., Gonçalves, G. E., Kelner, J., Sadok, D. H. and Curescu, C.: High availability in clouds: systematic review and research challenges, *Journal of Cloud Computing*, Vol. 5, No. 1, pp. 1–15 (2016).

[13] Zeng, Z., Zhang, Y., Xu, Y., Ma, M., Qiao, B., Zou, W., Chen, Q., Zhang, M., Zhang, X., Zhang, H., Gao, X., Fan, H., Rajmohan, S., Lin, Q. and Zhang, D.: TraceArk: Towards Actionable Performance Anomaly Alerting for Online Service Systems, *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 258–269 (online), DOI: 10.1109/ICSE-SEIP58684.2023.00029 (2023).

[14] Liu, G., Mok, A. and Yang, E.: Composite events for network event correlation, *Integrated Network Management VI. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management. (Cat. No.99EX302)*, pp. 247–260 (online), DOI: 10.1109/INM.1999.770687 (1999).

[15] Chen, Y., Zhang, C., Dong, Z., Yang, D., Peng, X., Ou, J., Yang, H., Wu, Z., Qu, X. and Li, W.: Dynamic Graph Neural Networks-based Alert Link Prediction for Online Service Systems.

[16] Li, M., Yang, M. and Chen, P.: Alarm Reduction and Root Cause Inference based on Association Mining in Communication Network, *Frontiers in Computer Science*, Vol. 5, p. 1211739.

[17] Chen, J., Wang, P. and Wang, W.: Online Summarizing Alerts through Semantic and Behavior Information, *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, New York, NY, USA, Association for Computing Machinery, p. 1646–1657 (online), DOI: 10.1145/3510003.3510055 (2022).

[18] Xu, J., Wang, Y., Chen, P. and Wang, P.: Lightweight and Adaptive Service API Performance Monitoring in Highly Dynamic Cloud Environment, *2017 IEEE International Conference on Services Computing (SCC)*, pp. 35–43 (online), DOI: 10.1109/SCC.2017.80 (2017).

