Sock Shopにおける負荷試験を反映した PodのCPUとメモリ使用率の閾値によるアラート設計

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構成されている EC サイトではスパイクアクセスの発生による障害時にアラートストームが起こる．これにより，管理者の対処が阻害され障害復旧の遅れにつながる．復旧が遅れることで，サービスの運営元は多大な損失を被る．マイクロサービスで動作している EC サイトの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 を対象にアラートストームを起こさないための提案をしている．提案として，仮に CPU 使用率とメモリ使用率の値が 80%を閾値としアラートを出す条件とし，負荷試験を行い閾値を適当な値にする．本稿で構築した Sock Shop のスパイクアクセス発生タイミングと提案で使用するメトリクスとエラーの関係に関しての実験をした．実験の結果，CPU 使用率は負荷の大きさによって値に変化があったが，メモリ使用率は値の変化が極めて小さかった．また，提案で定めた閾値の 80%のような一意の値は，アラートの閾値に適さないことが判明した．今後は，値の上がり幅やレスポンスタイムのような別のメトリクスをアラート条件として実験をする．

1. はじめに

背景

マイクロサービスは複数のアプリケーションの集合であり，それぞれが独立していて，スケーリングが可能なアーキテクチャである [1]．このマイクロサービスのデモアプリケーションとして Sock Shop が公開されている[*1]．マイクロサービスはアプリケーションの集合という特性からサービス間の通信が複雑になる．この複雑さは，障害分析やデバッグの様なエンジニアが運用する上で特有の課題をもたらす [2]．運用環境では，マイクロサービスの障害の大部分は，複数のインスタンス，環境構成，マイクロサービスの非同期相互作用の様な複雑で動的な相互作用やランタイム環境に関連しており，これらの障害を検出し，運用環境の実行時に障害を特定して，開発者が効率的に障害を解決できるようにすることが望ましい [3]．

障害の解決が遅れることで多大な経済的損失を被る可能性がある．例として，EC サイトの Amazon が行う年に 1 度の大規模セールイベントのプライムデーは 1 時間のダウンタイムの推定コストが最大 1 億ドルとされている．また，米国の 63 のデータセンターを対象に実施された調査によると，サービスのダウンタイムの平均コストは，2010 年には$505,502 であったのが 2016 年では$740,357 と以前から着実に増加している [4]．また，マイクロサービスのリソース使用量は，実装された機能と作業量によって異なるため，アクセスが集中し負荷が突然上昇するスパイクアクセスが発生すると，SLO 違反を起こす可能性がある [5]．したがって，サービスプロバイダーは通常，インフラストラクチャとサービスを運用するために大規模なシステム管理チームを設立している． 実際には，管理者はインフラストラクチャのステータスを追跡するために，システムから大量の監視データを収集する．大量のデータをリアルタイムに手動でチェックして解釈することは不可能であるため，データをチェックしてアラートをトリガーするアラートシステムを設定している． CPU 使用率のような選択した指標が所定の閾値を超えた場合，アラートが生成され，検査をフォローアップする管理者に通知される [6]．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，それぞれのコンポーネントには膨大な監視データがあるため，異なるコンポーネントが相互に影響しやすく，1 分間に数百から数千のアラートが発生することがある [7]．

課題

スパイクアクセスが発生し，大量のアラートが発生する．

[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/

大量のアラートを受信することで管理者は無視や誤解をしてしまい，問題診断のための待ち時間が長くなり，管理対象のネットワークやシステムに依存している企業にとって大きな損失となる [8]．このように，アラートが短時間に大量に発生することをアラートストームと呼ぶ．

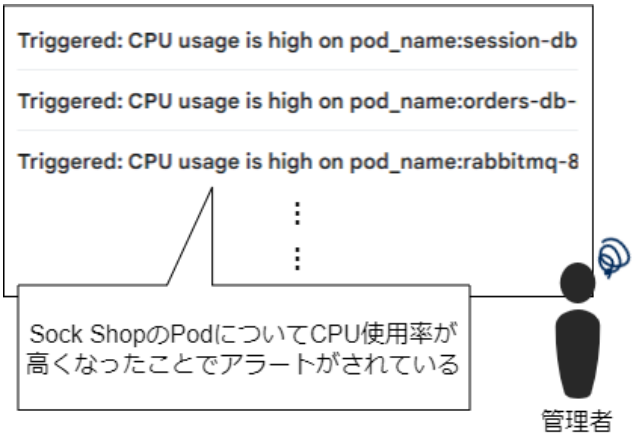

図 1　課題の概要図

図 1 は既存の監視サービスである Datadog を用いてアラートを設定した際のメールボックスの画像である．1 分間で 20 件のメールが届きアラートストームが発生した．

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，基礎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. 関連研究

管理者が問題判別プロセスに優先順位を付けるためにランキングを参照できるように，アラートの重要性をランク付けする新しいピアレビューメカニズムの研究がある [6]．上位にランクされたアラートは，他のルールからより多くの合意を得ているため，より重要となる．管理者は，アラートの順序を参照して，問題判別プロセスに優先順位を付けることができる．しかし，対象のサービスが無く提案の精度を評価することが困難としている．

教師なし機械学習を使用して，アラートとインシデントチケット内のテキストに基づいてアラートとインシデントチケットをクラスター化するフレームワークについての研究がある [9]．このフレームワークによって優先順位を付け，サービスの信頼性と可用性を向上させることができる．この研究では機械学習を行うため予め膨大な量のデータが必要となる．

侵入検知システムが生成する大量のアラートを管理するデータマイニングフレームワークの研究がある [10]．提案されたアラートデータマイニングフレームワークは，侵入検知システムのリアルタイム応答とアラートの量の削減に使用できる．このフレームワークは多くのマイニングタスクを使用して分析することを必要としている．

多くのアラートや誤ったアラートの発生を防ぐソリューションについて，超大規模システムに拡張する研究がある [11]．次世代システムの異質性と複雑さに合わせ，監視および管理するためのフレームワークを提案している．アラートの閾値を設定して単一の視点から中核的な問題を特定するのに役立つが，閾値の設定方法については述べていない．

3. 提案

提案方式

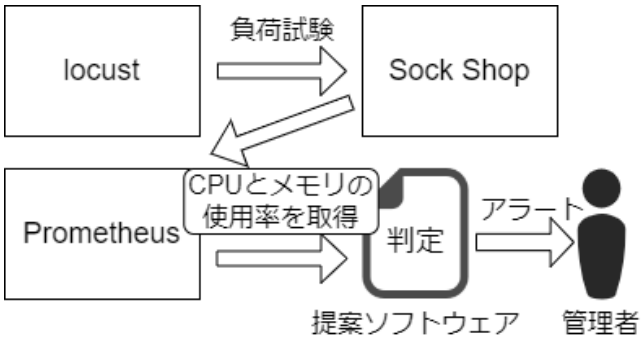

図 2　提案の概要

アラートストームを防ぐためにアラートの適切な閾値を負荷試験で決める提案をする．負荷試験により各サービスの Pod のメトリクスの値と発生するエラーの関係を求め，スパイクアクセスが発生した際，アラートストームを防ぐ閾値を決定する．提案の概要を図 2 に表す．提案ソフトウェアにより Sock Shop から Prometheus を用いて CPU とメモリの使用率を取得する．取得したメトリクスが提案方式により定める閾値を超えるときを条件として管理者にメールでアラートをする．よって提案では以下の (1)，(2) を仮の閾値とする．

(1) 各サービスの Pod の CPU の使用率が 80%以上
(2) 各サービスの Pod のメモリ使用率が 80%以上

(1)，(2) について，スパイクアクセスが発生したことによる負荷が CPU とメモリの使用率に影響するとし閾値に使用するメトリクスとした．また，問題が発生する可能性を事前に検知し予防的な対応を取ることができる値として 80%を仮の閾値とした．この閾値によってアラートストームを防ぐことができるかを負荷試験で検証する．アラートストームの基準は，課題の節で述べた 1 分間に 20 件のアラートとする．負荷試験から (1)，(2) でアラートストームの基準を超える場合，閾値を変更する必要がある．基準を超える場合は閾値を 5%増加する．増加を 5%としたのは適

当な閾値に変更するためである．

ユースケース・シナリオ

本稿のユースケースとしてファッション EC サイトがある．セールイベントや新商品の販売開始時でスパイクアクセスが発生する．その際，提案ソフトウェアによりアラートの数を未実装の場合より少なくすることができる．これにより，アラートストームの発生を防ぎ管理者の対処が円滑に行われることが可能になる．

4. 実装

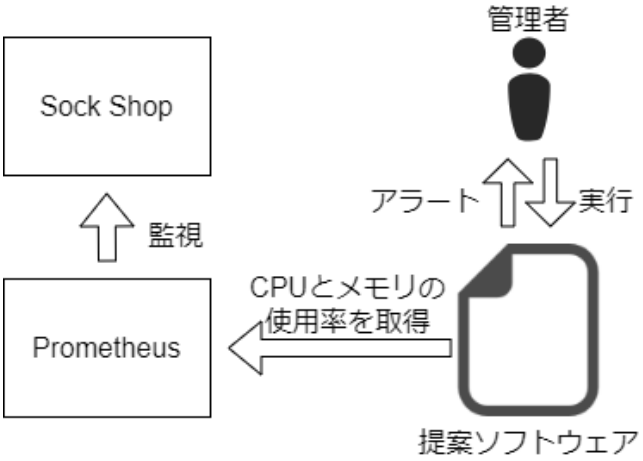

図 3 実装の概要

実装の概要を図 3 に表す．本稿では Kubernetes のディストリビューションである RKE2 を用いて Kubernetes クラスターを構築し実験をした．マスターノード 1 つ，ワーカーノード 3 つで構築されている．Sock Shop は 2 つのワーカーノードに Pod を分散して配置している．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．Sock Shop の Pod が配置されていないワーカーノードに Prometheus を実装しており，Sock Shop からのメトリクスを取得している．提案ソフトウェアを実装し，提案方式にによって管理者にアラートがされる．本稿では，提案ソフトウェアをに Prometheus と同じノードに実装することで各 Pod の名前を円滑に取得できるようにしている．

5. 実験と分析

図 3 に示した Sock Shop に対して負荷試験を行いスパイクアクセスを発生させる．その際の各サービスの Pod のメトリクスと発生するエラーに関しての分析をする．分析するメトリクスの値は提案で閾値を定めた CPU 使用率とメモリ使用率とする．

実験環境

実験環境の概要を図 4 に表す．RKE2 の Kubernetes クラスター内にある Sock Shop に対して，クラスター外の VM に構築している負荷試験ツール locust を用いて実験をする．クラスター外に構築することで，Sock Shop が動作する VM のリソースを locust が消費しない．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．locust をクラスター構成とすることでより多くのリクエストを送信可能としている．

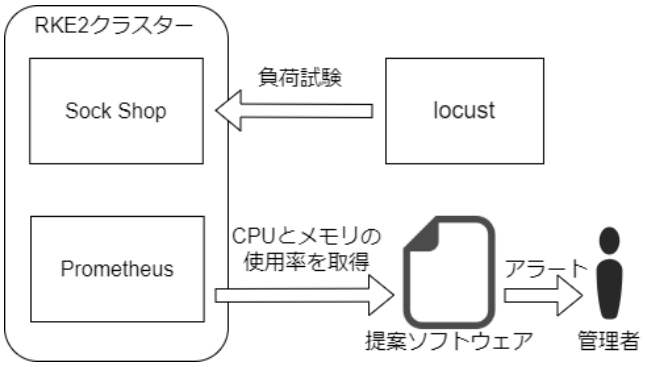

図 4 実験環境の概要

負荷試験のシナリオファイルは Sock Shop の開発者が提供しているものの修正版である．開発者が提供しているシナリオファイルは，1 つのユーザー情報を用いて同時に複数のリクエストを送信するため，操作が衝突し適切に負荷試験ができない．

負荷試験のシナリオは (1) から (7) の順である．
(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

Sock Shop に対して locust で負荷試験をした．前節で説明したシナリオを毎秒のユーザー数を 100，200，300 で 5 分間負荷を与える実験である．スパイクアクセスとなるユーザー数，また，CPU とメモリの使用率と発生するエラーの関係を分析していく．

図 5 にユーザー数 100 の各 Pod の CPU 使用率を表す．front-end と carts の CPU 使用率が上昇している．他のサービスについては変化が見られない．

図 6 にユーザー数 100 の各 Pod のメモリ使用率を表す．

carts, orders, shipping, queue-master のメモリ使用率が上昇している．CPU 使用率が少なかった orders, shipping, queue-master に関してはメモリ使用率が高い．

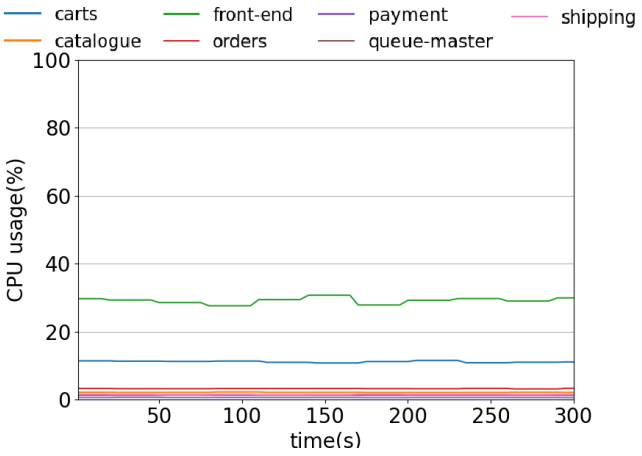

図 5 ユーザー数 100 の各 Pod の CPU 使用率

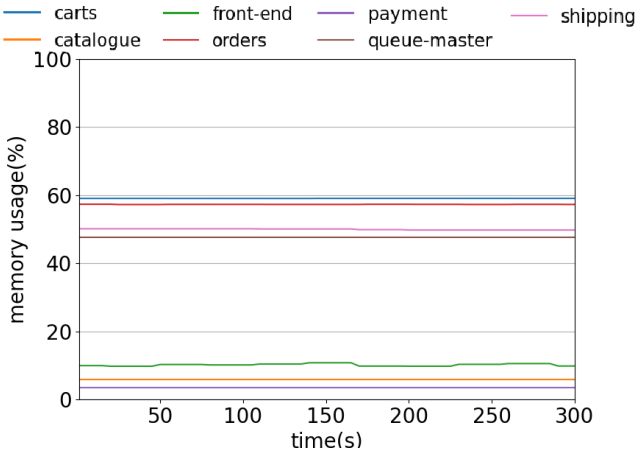

図 6 ユーザー数 100 の各 Pod のメモリ使用率

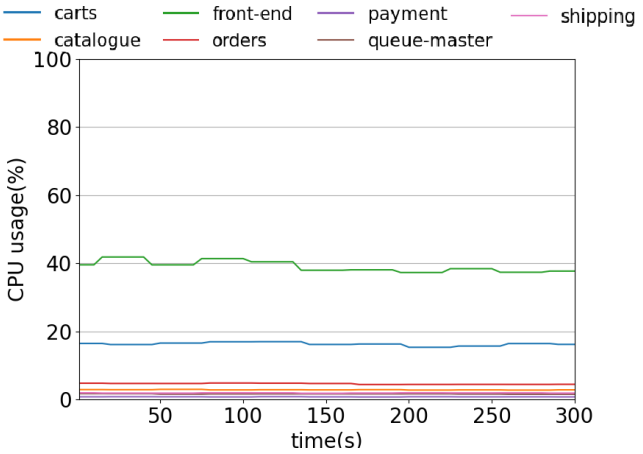

図 7 ユーザー数 200 の各 Pod の CPU 使用率

ユーザー数 100 の負荷試験では Sock Shop のサービスはエラーが発生しなかった．

図 7 にユーザー数 200 の各 Pod の CPU 使用率を表す．front-end と carts に関してユーザー数 100 と比べ CPU 使用率が上昇している．

図 8 にユーザー数 200 の各 Pod のメモリ使用率を表す．メモリ使用率に関してはユーザー数 100 と比べ差異が少なかった．

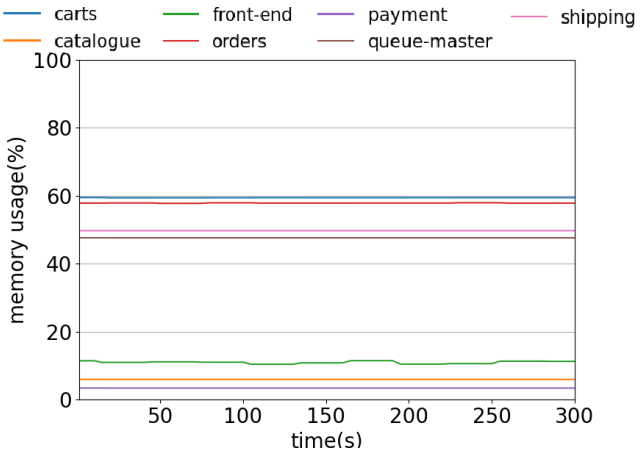

図 8 ユーザー数 200 の各 Pod のメモリ使用率

表 1 ユーザー数 200 の orders で発生したエラー

requests	fails	error
1,272	197	HTTPError(500)

表 1 にユーザー数 200 に orders で発生したエラーを表す．5 分間の間で orders に送られたリクエスト数 1,272 のうち 15%の 197 件が 500 エラーを返された．

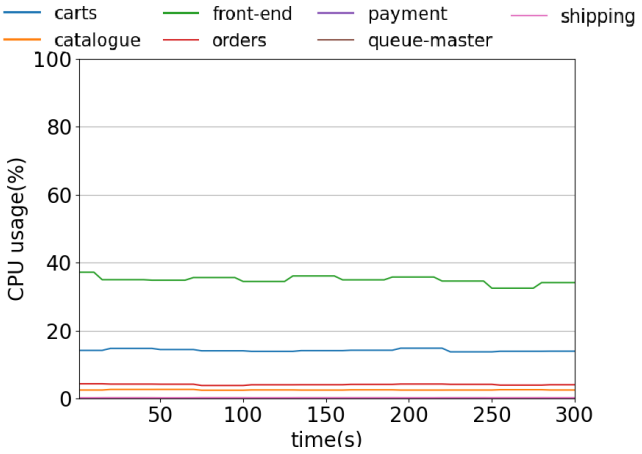

図 9 ユーザー数 300 の各 Pod の CPU 使用率

図 9 にユーザー数 300 の各 Pod の CPU 使用率を表す．

front-end と carts 共に，ユーザー数 200 と比べ使用率が低下する結果となった．

図 10 にユーザー数 300 の各 Pod のメモリ使用率を表す．ユーザー数 200 と同様に差異が少ない結果となった．

表 2 ユーザー数 300 の orders で発生したエラー

requests	fails	error
1,207	1,201	HTTPError(500)

表 2 にユーザー数 300 の orders で発生したエラーを表す．ユーザー数 200 よりも発生するエラーの数が増加している．5 分間の間で orders に送られたリクエスト数 1,207 のうち 1,201 件が 500 エラーを返された．99%以上のリク

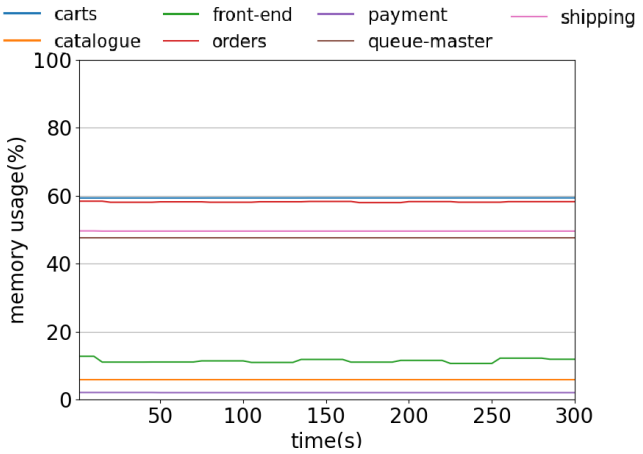

図 10 ユーザー数 300 の各 Pod のメモリ使用率

エストがエラーを返している．これらのことから，locust によるユーザー数 200 から 300 の間の負荷を与えることでエラーが発生する．

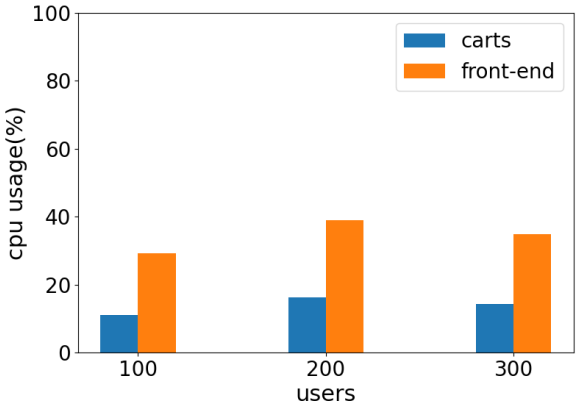

図 11 carts と front-end の Pod の平均 CPU 使用率

図 11 に値に変化のあった carts と front-end の CPU 使用率に対してユーザー数 100，200，300 で 5 分間負荷を与えた際の平均を表す．ユーザー数 300 の CPU 使用率が 200 よりも低くなることに関して，orders のリクエストがエラーを返すのみで処理が行われないことが原因と考えられる．

6. 議論

本稿では，各サービスの Pod の CPU 使用率とメモリ使用率の閾値を決定しアラートの条件とする提案をした．提案が課題の解決となるかについて実験結果をもとに議論する．orders からエラーを多く発生したユーザー数 300 の実験結果より，エラーが発生した CPU 使用率約 15%，メモリ使用率 60%となる前にアラートが出されるべきであった．しかし，提案では CPU とメモリ使用率共に 80%以上の場合としている．よって，提案の閾値は適さない結果となる．

CPU 使用率に関してはユーザー数の変化により値にも変化があるが，メモリ使用率は変化が小さい．このことからメモリ使用率はアラートの閾値に適さないといえる．CPU 使用率を閾値とする際に一意の値とすると，今回の orders ような値が小さい場合に作用しないことがある．CPU 使用率の上がり幅やサービス間のレスポンスタイム，Requests Per Second(rps) のような別のメトリクスを用いることで提案の改善が可能である．

7. おわりに

マイクロサービスで動作している EC サイトではセールイベントや販売開始時にスパイクアクセスが発生する．その際にアラートストームが起こり，管理者の対応を阻害しサービスの復旧が遅れ，損失が生じる．提案では，アラートストームとならないために，CPU とメモリ使用率を閾値とした．実験の結果からメモリ使用率と提案で定めた閾値の 80%のような一意の値による閾値は適さないとした．提案の改善に，値の上がり幅やサービス間のレスポンスタイム，rps のような別のメトリクスを用いることができる．

参考文献

[1] Thönes, J.: Microservices, *IEEE Software*, Vol. 32, No. 1, pp. 116–116 (online), DOI: 10.1109/MS.2015.11 (2015).

[2] Zhou, X., Peng, X., Xie, T., Sun, J., Ji, C., Li, W. and Ding, D.: Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study, *IEEE Transactions on Software Engineering*, Vol. 47, No. 2, pp. 243–260 (online), DOI: 10.1109/TSE.2018.2887384 (2021).

[3] Zhou, X., Peng, X., Xie, T., Sun, J., Ji, C., Liu, D., Xiang, Q. and He, C.: Latent Error Prediction and Fault Localization for Microservice Applications by Learning from System Trace Logs, *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, New York, NY, USA, Association for Computing Machinery, p. 683–694 (online), DOI: 10.1145/3338906.3338961 (2019).

[4] Chen, J., He, X., Lin, Q., Xu, Y., Zhang, H., Hao, D., Gao, F., Xu, Z., Dang, Y. and Zhang, D.: An Empirical Investigation of Incident Triage for Online Service Systems, *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 111–120 (online), DOI: 10.1109/ICSE-SEIP.2019.00020 (2019).

[5] Jindal, A., Podolskiy, V. and Gerndt, M.: Performance Modeling for Cloud Microservice Applications, *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, ICPE '19, New York, NY, USA, Association for Computing Machinery, p. 25–32 (online), DOI: 10.1145/3297663.3310309 (2019).

[6] Jiang, G., Chen, H., Yoshihira, K. and Saxena, A.: Ranking the Importance of Alerts for Problem Determination in Large Computer Systems, *Proceedings of the 6th International Conference on Autonomic Computing*, ICAC '09, New York, NY, USA, Associa-

tion for Computing Machinery, p. 3–12 (online), DOI: 10.1145/1555228.1555232 (2009).

[7] 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).

[8] 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).

[9] Lin, D., Raghu, R., Ramamurthy, V., Yu, J., Radhakrishnan, R. and Fernandez, J.: Unveiling Clusters of Events for Alert and Incident Management in Large-Scale Enterprise It, *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, New York, NY, USA, Association for Computing Machinery, p. 1630–1639 (online), DOI: 10.1145/2623330.2623360 (2014).

[10] Shin, M. S. and Jeong, K. J.: An Alert Data Mining Framework for Network-Based Intrusion Detection System, *Proceedings of the 6th International Conference on Information Security Applications*, WISA'05, Berlin, Heidelberg, Springer-Verlag, p. 38–53 (online), DOI: 10.1007/116049384 (2005).

[11] Sukhija, N. and Bautista, E.: Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus, *2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 257–262 (online), DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00087 (2019).

