マイクロサービスにおける適切なアラートの設定

疋田 辰起[1]　飯島 貴政[2]　串田 高幸[1]

概要：EC サイトのデモアプリケーション Sock Shop はマイクロサービスで構成されている．サービスに問題が発生した場合，状況を素早く認知するためにマイクロサービスでは監視を行う．アプリケーションの規模と複雑さが増大するにつれ，監視する必要のあるリソースの数，種類が増加する．処理しきれないリクエストが発生した際に，イベントログが膨大になる．アラートの条件を適切に設定しなければ，無駄なアラートにより，管理者は適切な判断を下せない．これを解決するために，各サービス間の応答時間 3000ms 以上とサービスのメモリ使用率 80%以上を条件とし，アラートをするソフトウェアを提案した．実験として Sock Shop が公開している負荷テストシナリオを基に負荷テストを行った．適切にアラートが出されるか，どのようなエラーに対応できるかを検証した．実験の結果，アラートは適切に出されていた．また，アラートが出されている時刻に 503 エラーを確認した．

1. はじめに

背景

マイクロサービスとは，独立して展開，スケーリング，テストできる小さなアプリケーションの集合である [1]．マイクロサービスと対をなすものとしてモノリシックアーキテクチャがある．モノリシックアーキテクチャはすべての機能が 1 つのアプリケーションにカプセル化されている従来のソフトウェア開発方法である．マイクロサービスはモノリシックアーキテクチャを一連の小さなサービスに分解し，軽量メカニズムを介して相互に通信させることにより，モノリシックアーキテクチャの課題を克服する代替方法である．従来のモノリシックソフトウェアが大きくなりすぎて対処できないため，多くの企業はそれらをマイクロサービスアーキテクチャスタイルに分割することに惹かれている [2].

マイクロサービスは EC サイトの Amazon, ZOZOTOWN のシステムで用いられている．マイクロサービスの EC サイトのデモアプリケーションとして Sock Shop が存在する．Sock Shop は EC サイトのユーザ向けの部分をシミュレートしており，マイクロサービス及びクラウドネイティブテクノロジーのデモンストレーションとテストを支援することを目的としている[*1]．

EC サイトはサービスに障害が発生することが取扱高の減少に繋がる．Amazon では短時間のうちにアクセスが急増することによって応答時間が増加した際に，100ms につき 1%売上が減少すると報告している [3]．よって，各サービスに問題が発生した場合に，状況を素早く把握する必要がある．そのためにマイクロサービスでは監視を行う．

従来，マイクロサービスの監視には管理者がダッシュボードを監視して異常を確認していた．今日では，統計ツールや機械学習を監視データに適用でき，管理者はこれらのシステムによって提供される自動アラートに依存できる [4].

マイクロサービスのサービスメッシュに Istio がある．Istio はマイクロサービスが相互にデータを共有する方法を制御するオープンソースのサービスメッシュである[*2]．マイクロサービスは各サービス間において API で通信をしいる．サービスが多く存在することで，サービス間の通信の把握が困難になる．Istio を使用することでトラフィック情報を収集・可視化して複雑化したサービス間の通信を把握しやすくする．

課題

課題は，処理しきれないリクエストが発生した際に，アラートの条件を適切に設定しなければ，冗長なアラートが多く出される．そのため管理者は正確な異常箇所の特定ができない．(図 1)．マイクロサービスはアプリケーションの規模と複雑さが増大するにつれ，監視する必要のあるリ

[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/

[*2] https://www.redhat.com/ja/ja/topics/microservices/what-is-istio

ソースの数，種類が増加する．サービスの状態と動作に関する最新の情報が膨大になることで，管理者はタイムリーな障害対応の意思決定を行いづらくなる [5].

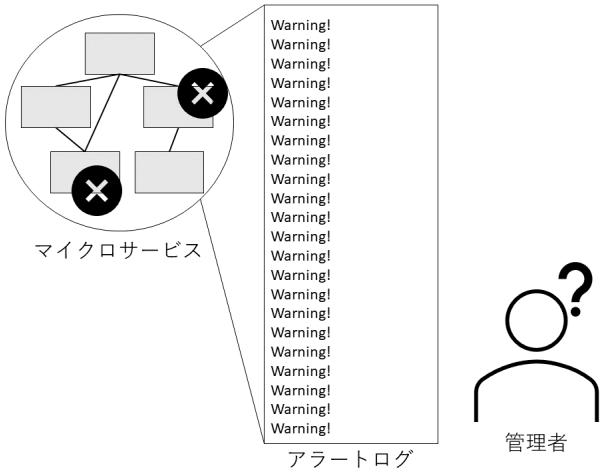

図 1 課題の概要

EC サイトでは，発売開始時やセールで短期間で大量のユーザがサイトにアクセスしている．例として，Alibaba では 2018 年 11 月 11 日に「Singles' Day Global Shopping Festival」を行った際，1 秒間に 491,000 件の販売取引処理を記録した [6].

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，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，全体を簡潔にまとめている．

2. 関連研究

Benjamin Mayer らは，マイクロサービスの監視と管理のためのダッシュボードを提案している [7]．15 人の開発者にマイクロサービスの監視と管理に関するインタビュー調査を行った．調査結果から，サービス API，サービスバージョン，サービスインスタンスの数，システムメトリクス，サービスの相互関係と依存関係，が重要な情報であるとした．これらをダッシュボードの要件としてグラフにより可視化している．しかし，障害が起きた際のアラートを実装していない．

Jacob Stephen らは，プロセスマイニングを適用して，アラートに優先順位を付与し，マイクロサービスベースのアプリケーションに対してより大きな脅威をもたらすアラートを特定する手段が必要であるとした．ビジネスプロセスのマイニングは，アプリケーションのログデータからプロセスモデルの形式で情報を出力する方法論であり，マイクロサービスの状況認識を向上させた [8].

Yang Tianyi らは，アラートの無効性をアラートのアンチパターンと呼び，アンチパターンを軽減する方法に関する実証研究を行った．Huawei Cloud での 2 年間に発生した数百万件のアラートの定量分析と 18 人のエンジニアへのアンケート調査を行った．その結果，4 つの個別のアンチパターンと 2 つの集合的なアンチパターンをまとめた．また，アンチパターンを軽減するための 4 つの現在の反応と，アラート戦略の構成に関する一般的な予防ガイドラインもまとめている [9].

3. 提案

提案方式

適切なアラートを出すための条件を設定するソフトウェアを提案する．提案の概要を図 2 に表す．マイクロサービスから，各サービス間の応答時間，各サービスのメモリ使用率を収集する．提案ソフトウェアにより，設定された条件からアラートを出すか判定をする．判定から，管理者にアラートを出す．

提案ソフトウェアにより，2 つの条件のを満たすときにアラートを出す．

(1) 各サービス間の応答時間が 3000ms 以上
(2) 各サービスのメモリの使用率が 80%以上

これらの条件を満たすとき，管理者宛にメッセージツールの slack にて通知を行う．(1) の理由として，直帰率がある．ユーザがアクセスした際の応答時間が 3000ms を超えると直帰率が 50%を超える [10]．よって，応答時間 3000ms 以上がアラートをすべき状態であるとした．(2) について，リクエスト数の増加による負荷がメモリ使用率に関係すると想定し条件を 80%とした．

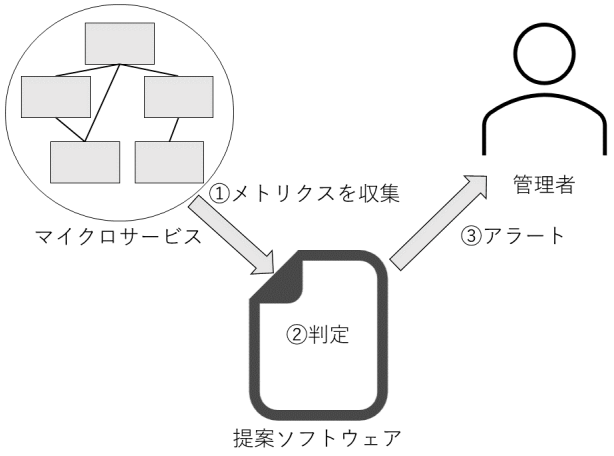

図 2 提案の概要

ユースケース・シナリオ

ユースケースとして EC サイトを想定している．セールや新商品販売時に平常時より多くユーザがアクセスするこ

とで障害が発生する．

ZOZOTOWN では年に 4 回のセールを行っている．セール開始直後は大量のアクセスやトラフィックが発生している．適切にアラートを設定することで，障害が起きた際に管理者の意思決定を円滑にすることが求められる．

4. 実装

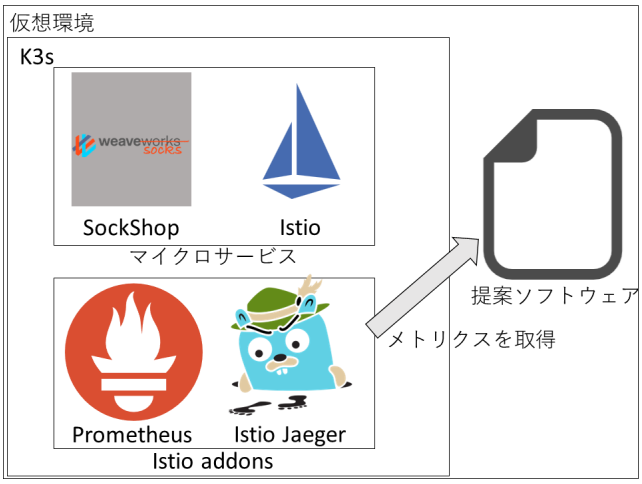

図 3 実装の概要

本研究の実装の概要を図 3 に表す．研究室の仮想環境に K3s を構築し，K3s 上にマイクロサービスのデモアプリケーションである Sock Shop を構築した．メトリクスを取得するために Sock Shop に対してサービスメッシュの Isito を構築している．Istio の addon である Jaeger，Prometheus から提案ソフトウェアにより必要なメトリクスを取得している．

提案ソフトウェアは図 4 のように構成されている．Prometheus からメモリ使用率を取得する memory.py，Jaeger から応答時間を取得する duration.py，slack にメッセージを送る alert.py これらをモジュールとし，main.py によって動作している．

main.py を実行し以下 4 つの順序で動作する．

(1) メトリクスを取得
(2) 取得したメトリクスからアラートを出すか判定
(3) アラートを実行
(4) slack にメッセージを送信

memory.py は prometheus_api_client により，各サービスの main.py 実行時のメモリ使用量とメモリの limit を取得している．その値から使用率を求め返すモジュールとなっている．

duration.py は Jaeger の web dashboard の URL を用いて api で各サービス間の通信に対するデータを json ファイルで取得する．取得した json ファイルから応答時間についてを抜き出し返すモジュールとなっている．

alert.py は main.py でアラートが必要と判定された際に，

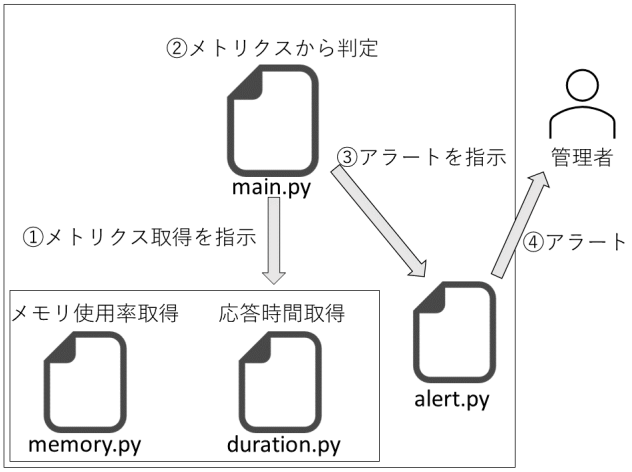

図 4 提案ソフトウェア構成

slack_webhook を用いてアラートを行うモジュールとなっている．

5. 実験と分析

実験環境

実験環境を図 5 に表す．Sock Shop と提案ソフトウェアが構築されている仮想環境とは別の仮想環境で，負荷テストツールの locust を構築する．負荷テストのシナリオとして，Sock Shop のテストシナリオを用いる．シナリオの内容は以下の通りとなっている．

(1) フロントページにアクセス
(2) ログイン
(3) カタログの中からランダムに商品をカートに入れる
(4) 購入

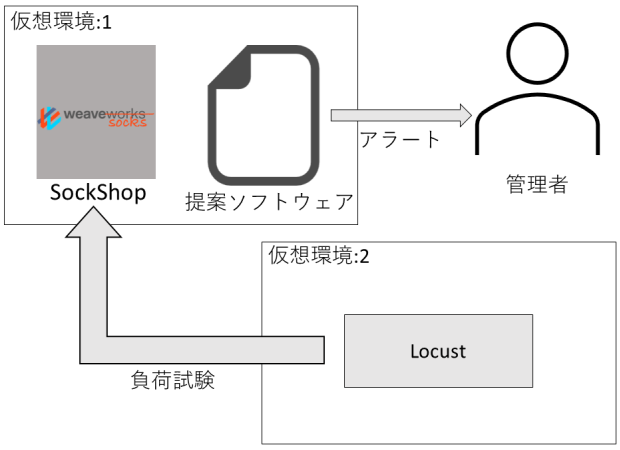

図 5 実験環境の構成

課題である処理しきれないリクエストが発生する状況を再現し，提案ソフトウェアを用いて適切にアラートがされるか実験を行う．Sock Shop のサービス front-end において，毎秒のリクエスト数 100 から 300 の間のメモリ使用率の関

係を図 6 に表す．リクエスト数 250 からメモリの使用率は上昇し始めている．この数のリクエスト数では最大でも 14%と提案の条件としているメモリ使用率 80%の状態にはならない．実験ではアラートがされる条件となるまで負荷を発生させる．

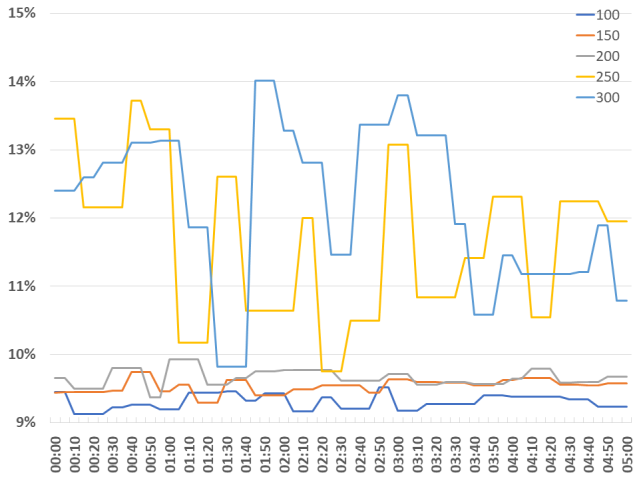

図 6　front-end リクエスト数毎のメモリ使用率

実験結果と分析

ユーザ数を 1500 人とし，1 ユーザが毎秒 1 回リクエストを行い，適切にアラートされているか分析する．また，発生しているエラーを取得して，提案ソフトウェアがどのエラーに対応できるかを検証する．

負荷試験の結果, 提案の条件でアラートがされていることを確認した．また，アラートがされている同時刻に HTTP ステータスコードに 503 を Jaeger dashboard で確認した (図 7)．503 エラー発生時にアラートがされてることより，

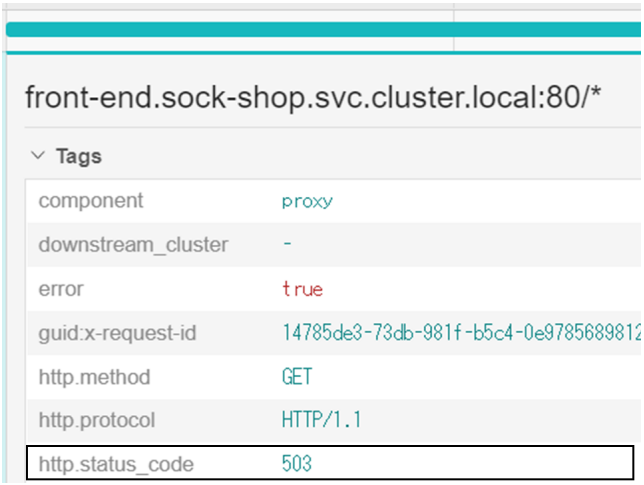

図 7　発生したエラー

提案ソフトウェアでは同時アクセス数の制限を超える状況を対応できることがわかる．平均応答時間が 3000ms を超える負荷を発生させた際に，Istio Jaeger，Prometheus の Pod が Evicted され実験継続が不可能となった (図 8)．

```
jaeger-c4fdf6674-ntc4m                  0/1     Completed
jaeger-c4fdf6674-2x5rr                  0/1     Evicted
jaeger-c4fdf6674-8jvxq                  0/1     Evicted
jaeger-c4fdf6674-7z2b7                  0/1     Evicted
jaeger-c4fdf6674-trwfn                  0/1     Evicted
jaeger-c4fdf6674-h4629                  0/1     Evicted
jaeger-c4fdf6674-7mn7r                  0/1     Evicted
jaeger-c4fdf6674-77482                  0/1     Evicted
jaeger-c4fdf6674-x6c7h                  0/1     Evicted
prometheus-85949fddb-rbqjw              0/2     Evicted
prometheus-85949fddb-sbbpc              0/2     Completed
jaeger-c4fdf6674-zq9xh                  0/1     Evicted
prometheus-85949fddb-94627              0/2     Evicted
jaeger-c4fdf6674-9fb88                  0/1     Evicted
prometheus-85949fddb-2fjh2              0/2     Evicted
jaeger-c4fdf6674-5grmz                  0/1     Evicted
```

図 8　Pod の状態

6. 議論

本稿では，各サービス間の応答時間と各サービスのメモリ使用率を条件としてアラートを設定した．この提案による課題の解決について，実験により 503 エラー発生時以外にアラートがされていないことから，アラートの無効性がないとする．無駄なアラートを減らすという点は，この提案以外のアラートについての数値が求められるため，今後実験を行う必要がある．

提案について応答時間では直帰率の関係から 3000ms 以上のときとしたが，メモリ使用率に関して，メモリ使用率とサービスの状態に相関を見つけることができなかった．サービスの過去のメモリ使用率の傾向から平常時の値を求め，その値を超える状況をを瞬間的に多くのリクエストが発生した際とし，応答時間の条件と組み合わせることでより適切なアラートとなる．今後の展望として過去のサービスのメトリクスの傾向から，アラートの閾値を自動的に判断するようなソフトウェアを提案する．

負荷試験の際に，Pod が Evicted され実験の継続ができなかった．提案ソフトウェアのアラートがメトリクスを条件としていることより，Pod のエラーに対してのアラートがされないことがわかる．エラーの原因としてリソース不足がある．Kubernetes は kube-system の Pod のリソースを最優先で確保する．負荷試験により Sock Shop，Istio の Pod がリソースを圧迫したことで，Kubernetes が Evicted を実行した．node のリソースを増やす必要がある．更に負荷試験を行い，503 エラー以外のエラーに提案ソフトウェアが対応できるか検証する必要がある．

7. おわりに

EC サイトで用いられているマイクロサービスでは，発

売開始時やセールで短期間で処理しきれないリクエストが発生する．これにより障害が起きた際に，管理者は膨大なイベントログから原因を特定しなければならない．そのため，アラートを設定し管理者に通知を行う．しかし，適切にアラートを設定しなければ，アラートの洪水となってしまう．提案ソフトウェアでは，応答時間とメモリ使用率からアラートを設定した．実験により提案ソフトウェアは適切に動作し，アラートがされている際に 503 エラーが出ていることが確認された．

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] Kazanavičius, J. and Mažeika, D.: Migrating Legacy Software to Microservices Architecture, *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pp. 1–5 (online), DOI: 10.1109/eStream.2019.8732170 (2019).

[3] Teo, Y. M.: Modelling flash crowd performance in peer-to-peer systems: Challenges and opportunities, *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, pp. 3–4 (online), DOI: 10.1109/ISMS.2014.165 (2014).

[4] Marie-Magdelaine, N., Ahmed, T. and Astruc-Amato, G.: Demonstration of an Observability Framework for Cloud Native Microservices, *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 722–724 (2019).

[5] Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J. and Tilkov, S.: Microservices: The Journey So Far and Challenges Ahead, *IEEE Software*, Vol. 35, No. 3, pp. 24–35 (online), DOI: 10.1109/MS.2018.2141039 (2018).

[6] Li, F.: Cloud-Native Database Systems at Alibaba: Opportunities and Challenges, *Proc. VLDB Endow.*, Vol. 12, No. 12, p. 2263–2272 (online), DOI: 10.14778/3352063.3352141 (2019).

[7] Mayer, B. and Weinreich, R.: A Dashboard for Microservice Monitoring and Management, *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pp. 66–69 (online), DOI: 10.1109/ICSAW.2017.44 (2017).

[8] Jacob, S., Lee, B. and Qiao, Y.: Applying process mining to improve microservices cyber security situational awareness. (2020).

[9] 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, *arXiv preprint arXiv:2204.09670* (2022).

[10] 富田啓太，中川翔太，串田 高幸：二分探索法を用いた Kubernetes Pod 数の決定の 自動化による応答時間の遅延の低減，技術報告，クラウド・分散システム研究室，CDSL-TR-100 (2022).