システム障害の復旧時におけるPod数の最も多い対象のアラートを選出することによるチケット登録件数の限定

平尾 真斗[1]　串田 高幸[1]

概要：障害を迅速に解決することは，ビジネスへの影響を最小限に抑え，ユーザの信頼を維持するうえで重要である．障害の発生時に通知されたアラートはチケットシステムへ登録される．チケットシステムへ登録されたチケットは 1 次調査の対応者によって調査される．障害発生時に障害発生箇所に関連して障害発生箇所以外からアラートが通知されチケットとして登録される場合がある．課題は障害の発生箇所と障害の発生箇所以外のアラートが通知されチケットとして登録されてしまうことで，1 次の調査者がどのチケットから対応すればいいか判別がつきにくくなることである．提案では，障害発生時にチケットシステムに登録されるアラートを限定するために Pod 数を用いる．監視対象に配置されている Kubernetes の Pod の台数を対象ごとに取得し，その中で最も Pod 数の多い対象にアラートを限定し，チケットとして登録する．基礎実験では，2025 年 4 月 26 日に発生したシステム障害で登録されたチケットの件数をまとめる．障害は Kubernetes クラスタの Node 上に配置していた vmware exporter の Pod が，メトリクス取得元である ESXi の電源停止により正常に動作しなくなっていた事例である．この障害は監視対象である Node がダウンしたことによって Pod のアラートが通知され，チケットとして登録された．そのため，障害発生箇所を Node のチケットとし，障害発生箇所以外を Pod のチケットとした．チケットの件数は，障害発生箇所である Node のチケットが 1 件，障害発生箇所以外で通知された Pod のチケットが 8 件登録されており，障害発生箇所以外のチケットも登録されていることを確認した．

1. はじめに

背景

企業で運用されているシステムは，数千ものコンポーネントによって構成され，ユーザを支援している [1]．このような大規模システムでは，ユーザに対して安定したサービスを継続的に提供することが求められる [2]．障害が発生した際に，迅速な復旧対応ができなければ，企業は重大な金銭的損失を被る可能性がある．例えば，2017 年に世界最大級の EC サイトである Amazon で発生した障害では，1 億 5,500 万ドルを超える損失が発生したと推定されている [3].

障害を迅速に解決することで，ビジネスへの影響を最小限に抑え，ユーザからの信頼を高めることができる．監視は，運用中のシステムの可用性を向上させるための重要な手段として活用されている [4, 5]．監視システムは，対象内に設置されたエージェントからデータを収集し，その値が事前に設定された閾値を超えた場合にアラートを通知する [6–8]．アラートはチケットシステムに送信され，チケットとして 1 次調査の担当者に割り当てられる [9].

チケットはアラートから生成されるジョブであり，アラートの内容にもとづいて重大度，要約，障害の発生時刻の情報が記載される [10]．障害の原因が 1 次調査の担当者による初期調査で判明しない場合は，2 次調査の担当者にチケットが引き継がれ，調査が実施される．調査の際には，Runbook と呼ばれる手順書が参照される [11]．Runbook には，アラート発生時の具体的な対応手順が記載されており，1 次調査の担当者は Runbook を元に調査と対処を行う．

障害対応次のチケットに関する調査と対処を行うユースケースとして，東京工科大学コンピュータサイエンス学部の研究室である Cloud and Distributed Systems Laboratory（以下，CDSL と呼ぶ）を対象とする．図 1に，CDSL で稼働しているシステムと調査までの流れを示す.

CDSL では,10 台の物理機器が配置されており，それぞれの物理機器には，米ブロードコム社の VMware 製品である ESXi がインストールされている．ESXi 上には Virtual Machine(以後:VM と呼ぶ) が配置されている [12]．図中の Jasmine は ESXi が配置された物理機器である．Prometheus は exporter から Metrics を受け取る [13, 14].

[1] 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 クラウド・分散システム研究室

Metrics はそれぞれの exporter から取得した監視対象のデータである．vmware exporter は ESXi 上のホストや VM の CPU，メモリ量，ディスク使用量を監視する．Archive-server は ESXi 上に保存されている VM のデータの中でいらなくなったデータをアーカイブする目的で使用される．Node exporter は監視対象上の OS で動作し，Node の状態を監視する．Alertmanager は Prometheus 上で設定されたアラートルールが閾値を超えた場合にアラートを通知する．Ticket Create は Alertmanager から通知されたアラートからチケットを作成し，Redmine に登録する．Redmine には Ticket Create が作成したチケットが保存される．1 次調査の担当者は障害発生時にチケットの調査と対処を行う．

これらのチケットに対して調査および対処を行うため，CDSL では日常的に監視作業が実施されている．監視作業では Ticket Create が作成したチケットをもとに 1 次調査を行う．監視作業は，研究室の学生 5 名によってシフト制で担当されており，9 時 00 分〜10 時 30 分，10 時 30 分〜12 時 00 分，12 時 00 分〜14 時 30 分，14 時 30 分〜17 時 30 分の 4 交代制で行われている．1 次調査チケットの割り振りは，Ticket Create によって，その時間帯に割り当てられた監視担当者へ自動的に行われる．監視担当者は，受け取ったチケットに対して 1 次調査と初期対応を行う．1 次調査で原因が特定できない，もしくは対応が困難な場合には，監視担当者が 2 次チケットを発行し，さらなる調査を行う．

2025 年 4 月 26 日に発生した障害では，Kubernetes クラスタの Node 上に配置していた vmware exporter の Pod が，メトリクス取得元である ESXi の電源停止により正常に動作しなくなっていた．クラスタは Node1 および Node2 の 2 台で構成されている．Node1 には，Prometheus, Alertmanager，Grafana，Redmine，MariaDB，cAdvisor，Blackbox Exporter, Metrics Server の 8 つの Pod を配置し，Node2 には，vmware exporter の Pod を 8 つ配置していた．ESXi の電源が停止し，vmware exporter の Pod がホストと通信ができなくなった際に，過剰にメモリを消費し，Node2 のメモリが枯渇した．結果，Node2 のステータスは「NotReady」，vmware exporter の Pod のステータスは「Terminating」となった．

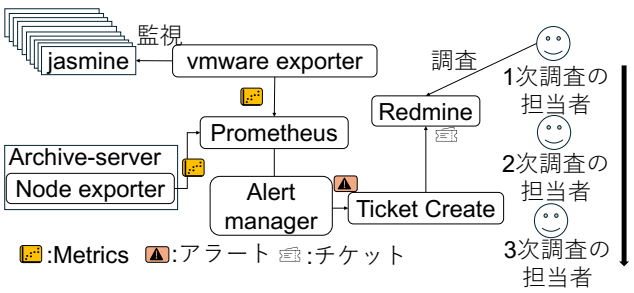

図 1: CDSL で稼働しているシステムと調査までの流れ

課題

課題は障害の発生箇所と障害の発生箇所以外のアラートが通知されチケットとして登録されてしまうことで，1 次の調査者がどのチケットから対応すればいいか判別がつきにくくなることである．2025 年 4 月 26 日に発生した障害では，障害発生箇所である Node から 1 件，vmware exporter の Pod から 8 件のアラートが通知され，チケットとして登録された．障害の復旧の際に Node がダウンしている場合，Pod の調査をしても復旧には至らない．Node の調査を行わなければ，障害の復旧はできない．そのため障害の発生箇所である Node のチケットに限定できなければ，1 次の調査者がどのチケットから対処するのか判断がつきにくくなってしまう．

各章の概要

第 2 章では関連研究について議論する．第 3 章では，課題に対しての提案方式について説明する. 第 4 章では，実装したソフトウェアについて説明する．第 5 章では，評価実験について説明する．第 6 章では，提案手法についての議論をする．第 7 章では，本稿のまとめを行う．

2. 関連研究

クラウドサービス上で発生したノイズと誤検知を削減する研究がある [15]．アプローチとして，閾値ベースの監視方式ではなくウィンドウベースの方式を用いることで，一時的に閾値を超えるノイズや誤検知を削減する．これにより突発的に閾値を超えた障害に対するアラートの削減ができる．一方で継続的に閾値を超える障害のアラートを削減することに関して改善の余地がある．

管理者が作成したインシデントチケットから見逃された重大なアラートを自動的に検出する研究がある [16]．この論文では，チケット内に含まれるドメイン語と呼ばれる単語に焦点を当て，そのチケットに関連する障害に対してのアラートを削減する．一方で，障害がチケットに過去に登録されておらず，ドメイン語を抽出できない障害に対しては，アラートを削減できないことに関して改善の余地がある．

タイトルや本文中の文字列の類似性にもとづいて，チケットを統合する手法が提案されている [17]．この手法では，TF-IDF やベクトル表現，類似度スコアを用いて，チケット同士の関連性を比較し，類似した内容のチケットをまとめることで，対応作業の効率化を図っている．一方で，障害の発生箇所がシステム内の異なるレイヤーにまたがる場合，同一の障害に起因して発生した複数のチケットであっても，タイトルや本文の記述が異なるために，同一障害に関するチケットとして統合されないことがある．具

体的には，Kubernetes 環境において Node とその上で動作する Pod の両方に障害が波及するようなケースでは，それぞれに対して個別のチケットが発行されるにもかかわらず，これらが同一原因であると判断されず，統合されないという問題がある．

3. 提案

本提案の目的は，障害の発生箇所のみにアラートを限定して通知し，チケットシステムに登録されるチケットの件数を削減することである．アラートを障害の発生箇所に限定するために，監視対象に配置されている Pod 数を用いる．アラートはチケットシステムへ登録し，障害の根本原因となるアラートのみをチケットとして登録する．本研究で扱う監視対象は Kubernetes を用いたクラスタを対象とする．提案の概要を 2に示す．

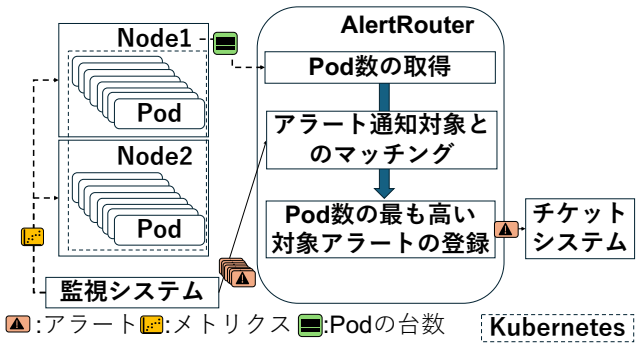

図 2: 提案の概要

監視システムは対象の監視を行うためにメトリクスを取得する．事前に設定されたアラートルールの閾値が下回った場合，アラートを通知する．監視対象は Node1 と Node2 のノードとそれぞれの Node 上に 8 つずつ Pod が配置されている．チケットシステムは通知されたアラートによって登録されたチケットを保存する．Pod 数は Kubernetes の KubeAPI を配置した Node に問い合わせ確認する．AlertRouter は Pod の台数と通知されたアラートにある監視対象のアドレスをもとにアラートを限定して通知する．AlertRouter は 3 つの機能をもとに通知されるアラートを限定する．1 つ目は Pod 数の取得であり，監視対象である Kubernetes のクラスタの中で KubeAPI を配置した Node に問い合わせを行い，Node ごとに配置されている Pod の数を取得する．アラート通知対象とのマッチングでは，通知されたアラートの中からどの対象からアラートが通知されているのかを抽出し，通知されたアラートと同じ対象の関係性を持つテーブルを選択する．3 つ目は，Pod 数のもっと高い対象のアラートの登録であり，選択されたテーブルの中で最も Pod 数が多い対象のアラートに限定してチケットを登録する．

Pod 数の取得

Pod 数の取得では，Kubernetes の KubeAPI を配置した Node に問い合わせを行い，それぞれの Node に配置されている Pod の台数を取得する．Pod 数はこの後の，アラート通知対象とのマッチングと Pod 数のもっと高い対象のアラートの登録で使用する．

アラート通知対象とのマッチング

アラート通知対象とのマッチングでは，監視対象から通知されたアラートに含まれる対象のアドレスをもとにどの対象の Node の Pod 数を選択するかを決定する．図 3にアラート通知対象とのマッチングの流れを示す．監視システムはアラートを通知する．Instance はアラート内に含まれる監視対象のアドレスが含まれる．Instance Node2 は Node2 のアドレスを示す．例えば，Node2 とその Node に配置されている 8 つの Pod がダウンし，9 件のアラートが通知された際に Node2 と Node2 に配置されている 8 つの Pod の配置構成を示すテーブルが選択される．

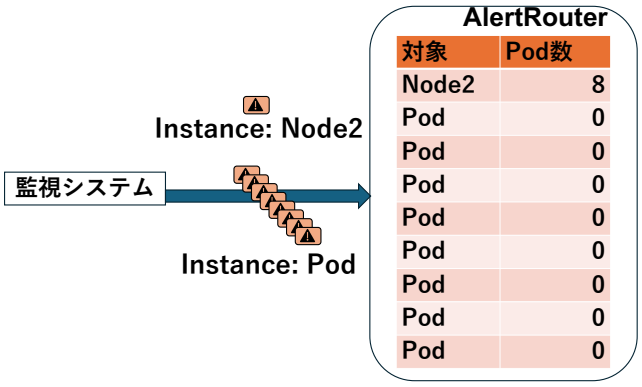

図 3: アラート通知対象とのマッチング

Pod 数が最も高い対象アラートの登録

Pod 数が最も高い対象アラートの登録では，選択されたテーブルの中からどの監視対象のアラートを限定してチケットシステムに登録するかを選択する．その際に式 3を用いて選択を行う．

$$Score\_MAX = \max(S_1, S_2, \ldots, S_n) \qquad (0)$$

Score_MAX は Pod 数の中で最も高い値を示す．Node2 は Pod 数が 8 であり，Node2 上に配置されている Pod は Pod 数が 0 である．そのため Score_MAX の値は，8 つとなり，Node2 が選択される．最後に Node2 のアドレスを含むアラートを選択し，チケットシステムに登録する．チケットを Pod 数が最も高い箇所に限定することで，アラートの中で Node を示すアラートに限定できる．

ユースケース・シナリオ

ユースケースシナリオとして障害発生時におけるアラートからチケットが登録されるまでの流れを想定する．図4にユースケースシナリオを示す．

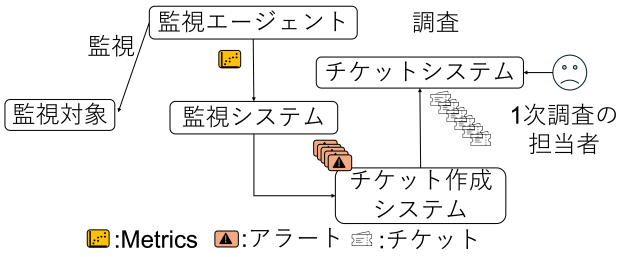

図 4: ユースケースシナリオ

監視対象は CDSL で運用されているサーバである．監視エージェントは対象の監視を行い，メトリクスを監視システムに提供する．監視システムは対象のメトリクスの値が，事前に設定した閾値を下回った場合アラートを通知する．チケット作成システムは，アラートを元にチケットシステムにチケットを登録する．チケットシステムはチケットの保存を行っている．障害の種類によっては，障害発生箇所と障害発生箇所以外でアラートが通知され，チケットとして登録される場合がある．

4. 実装

本提案のソフトウェアは Python3.10.12 を使用し作成する．ライブラリは，Flask，subprocess，python-redmine である．提案するソフトウェアは AlertRouter とする図 5に，実装する AlertRouter の概要を示す．

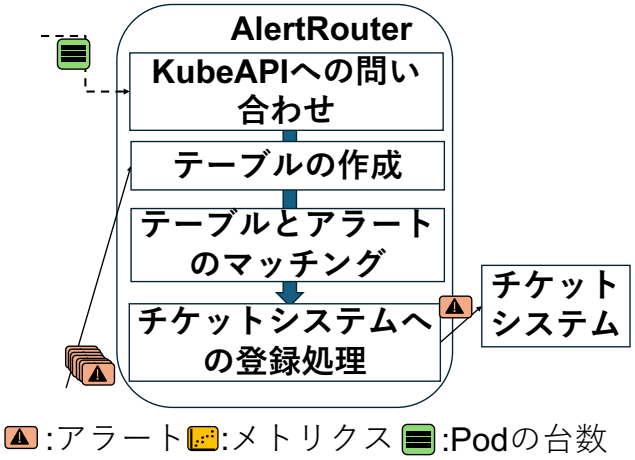

図 5: AlertRouter の概要

チケットシステムにはチケットを登録する．AlertRouter には 4 つの機能がある．1 つ目は KubeAPI への問い合わせであり，監視対象である Kubernetes の Node にどの Pod が配置されているかや，Node の構成を取得する．2 つ目は，テーブルの作成である．監視対象の Node ごとにどの Pod が配置されているかをテーブルとしてまとめる．3 つ目は，テーブルとアラートのマッチングであり，通知されたアラートに含まれる監視対象のアドレスとマッチする対象のテーブルを取得する．4 つ目は，チケットシステムへの登録処理であり，通知されたアラートの中で Pod 数が最も多い対象のアラートを選出し，チケットシステムへ登録する．

5. 評価実験

評価実験では，障害の事例を再現しチケットシステムに登録されたチケットの件数と，障害の発生箇所に限定してチケットが登録できているかを比較する．事例は，2025 年 4 月 26 日 7 時 24 分に起きた障害であり，Kubernetes クラスタの Worker のメモリ使用量が枯渇したことで Pod がダウンした事例を再現する．

基礎実験

基礎実験では，2025 年 4 月 26 日 7 時 24 分に発生した，Kubernetes クラスタの Worker のメモリ使用量が枯渇したことで Pod がダウンした事例でチケットシステムに登録されたチケットの件数をまとめる．

実験環境

図 6に実験環境を示す．実験環境は 2025 年 4 月 26 日

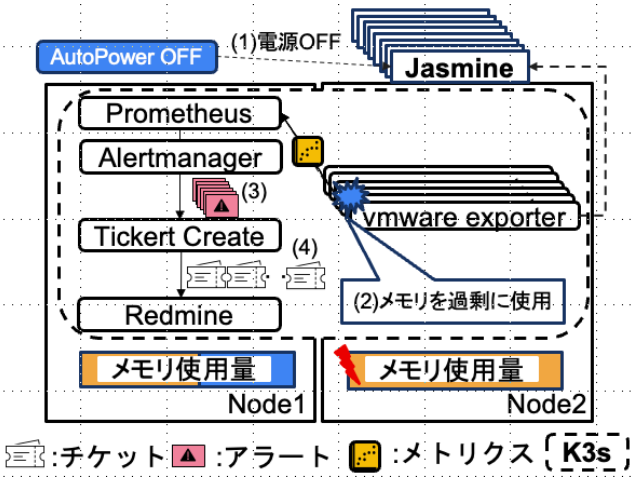

図 6: 実験環境

7 時 24 分の時点で配置されていたシステムの図である．AutoPewer OFF は定期的に ESXi のサーバの電源を落とすシステムである．監視対象である ESXi のサーバは 10 台配置されており，そのうちの 8 台が監視されていた．図中に示されている Jasmine は CDSL 上で運用されている ESXi がインストールされたサーバである．Node の構成は 2 台であり，それぞれ Node1 と Node2 とする．Node

の性能は vCPU: 4[core],RAM:4[GB],SSD:40[GB] で作成されていた．Prometheus は監視サーバであり，事前に設定された閾値が異常値になった際にアラートを発行する．vmware exporter は監視エージェントであり，ESXi であるサーバを監視する．vmware exporter は 8 台配置されている．Alertmanager はアラートを通知するコンポーネントであり，Prometheus が発行したアラートを通知する．Ticket Create は通知されたアラートを元にチケットを作成する．Redmine はチケットを保存する．

次に障害発生の流れを示す．(1) は，AutoPower OFF によって ESXi である Jasmine が停止してことを示している．(2) は，Jasmine を監視する vmware exporter が Jasmine からメトリクスを取得できなくなり，メモリを過剰に使用していることを示している．その際に Node2 のメモリが枯渇し，障害が発生した．(3) は，Prometheus が監視対象の Node1 と Node2 に配置されている Pod にアクセスできなくなり，アラートを発行していることを示している．その際に Alertmanager が Prometheus が発行したアラートを通知した．(4) は Ticket Create が Redmine にチケットを登録している様子を示す．

登録されたチケットの件数

図 7に，Redmine に登録されていたチケットの件数を示す．

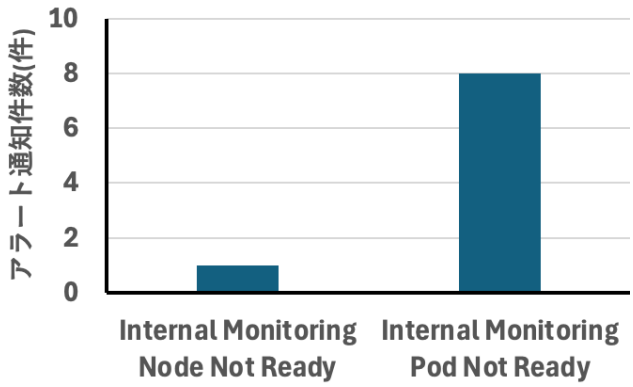

図 7: 登録されたチケットの件数

「Internal Monitoring Node Not Ready」は，Kubernetes の Node のステータスが「NotReady」になった際に通知されるアラートである．その名前と同じチケットが Redmine に登録された．また，「Internal Monitoring Pod Not Ready」は，Node 上の Pod が「Running」以外のステータスになった際に通知される．その名前と同じチケットが Redmine に登録された．Node2 が「NotReady」となり，Node2 上に配置されていた vmware exporter の Pod（8 台）も同時に異常となったため，合計 9 件のチケットが登録された．

今回の障害では，Node2 がダウンしたことで Pod の障害が引き起こされた．つまり，障害の原因箇所が Node で障害の発生箇所以外が Pod の障害となる．障害の発生箇所と障害の発生箇所以外のアラートが通知されチケットとして登録されてしまうことで，1 次の調査者がどのチケットから対応すればいいか判別がつきにくくなる．障害の復旧の際に Node がダウンしている場合，Pod の調査をしても復旧には至らない．Node の調査を行わなければ，障害の復旧はできない．そのため，Node のアラートのみを限定して通知し，チケットとして登録する必要がある．

6. 議論

AlertRouter は Node と Pod に関連するアラートが同時に通知された際には有効であるが，Pod のみのアラートが通知された際に改善点がある．例えば，2025 年 6 月 24 日の 4 時 36 分に発生した障害では，Kubernetes クラスタの Node の Disk 使用率が 90%を超えた．その際に，Node 上に配置されている Pod のステータスが，「evicted」になり，Pod の台数分のアラートが通知された．Pod だけのアラートの場合，Pod 数が全て 0 になるため，アラートを限定はできない．そこで 0 でまとまった対象のアラートはまとめて通知し，チケットシステム上でチケット 1 つに対して残りのアラートをコメントとして追加することで，チケット数は 1 つにできる．

障害発生時に通知されたアラートは AlertRouter を経由してチケットシステムに登録される．その際に Prometheus，Alertmanager，AlertRouter，チケットシステムのように経由するコンポーネントが多くなれば，チケットとして登録される時間も増加してしまう．障害の復旧は迅速に行わなければならないため，1 次調査の担当者にいかに迅速にチケットを提供できるかが鍵となる．改善案として Node が落ちた際は Pod のアラートを優先せずに Node のみをアラートとして通知するように Prometheus のファイルを更新する．そのようにすれば監視対象に異常が起きた際に迅速にチケットシステムへの登録を行うことができる．

7. おわりに

障害を迅速に解決することは，ビジネスへの影響を最小限に抑え，ユーザの信頼を維持するうえで重要である．障害の発生時に通知されたアラートはチケットシステムへ登録される．チケットシステムへ登録されたチケットは 1 次調査の対応者によって調査される．障害発生時に障害発生箇所に関連して障害発生箇所以外からアラートが通知されチケットとして登録される場合がある．課題は障害の発生箇所と障害の発生箇所以外のアラートが通知されチケットとして登録されてしまうことで，1 次の調査者がどのチケットから対応すればいいか判別がつきにくくなることである．提案では，障害発生時にチケットシステムに登録されるアラートを限定するために Pod 数を用いる．監視対象

に配置されている Kubernetes の Pod の台数を対象ごとに取得し，その中で最も Pod 数の多い対象にアラートを限定し，チケットとして登録する．基礎実験では，2025 年 4 月 26 日に発生したシステム障害で登録されたチケットの件数をまとめる．障害は Kubernetes クラスタの Node 上に配置していた vmware exporter の Pod が，メトリクス取得元である ESXi の電源停止により正常に動作しなくなっていた事例である．この障害は監視対象である Node がダウンしたことによって Pod のアラートが通知され，チケットとして登録された．そのため，障害発生箇所を Node のチケットとし，障害発生箇所以外を Pod のチケットとした．チケットの件数は，障害発生箇所である Node のチケットが 1 件，障害発生箇所以外で通知された Pod のチケットが 8 件登録されており，障害発生箇所以外のチケットも登録されていることを確認した．

参考文献

[1] Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y. and Chen, X.: Log Clustering Based Problem Identification for Online Service Systems, *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 102–111 (2016).

[2] Malhotra, A., Elsayed, A., Torres, R. and Venkatraman, S.: Evaluate Solutions for Achieving High Availability or Near Zero Downtime for Cloud Native Enterprise Applications, *IEEE Access*, Vol. 11, pp. 85384–85394 (online), DOI: 10.1109/ACCESS.2023.3303430 (2023).

[3] He, S., He, P., Chen, Z., Yang, T., Su, Y. and Lyu, M. R.: A Survey on Automated Log Analysis for Reliability Engineering, *ACM Comput. Surv.*, Vol. 54, No. 6 (online), available from ⟨https://doi.org/10.1145/3460345⟩ (2021).

[4] Bashir, A. and Soomro, T.: Comparative Study on Incident Management, *International Journal of Applied Information Systems*, Vol. 3, pp. 21–23 (2012).

[5] Zong, B., Wu, Y., Song, J., Singh, A. K., Cam, H., Han, J. and Yan, X.: Towards scalable critical alert mining, *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. 1057–1066 (online), DOI: 10.1145/2623330.2623729 (2014).

[6] Renita, J. and Elizabeth, N. E.: Network's server monitoring and analysis using Nagios, *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 1904–1909 (2017).

[7] Tang, L., Li, T., Pinel, F., Shwartz, L. and Grabarnik, G.: Optimizing system monitoring configurations for non-actionable alerts, *2012 IEEE Network Operations and Management Symposium*, pp. 34–42 (2012).

[8] Zhao, N., Jin, P., Wang, L., Yang, X., Liu, R., Zhang, W., Sui, K. and Pei, D.: Automatically and Adaptively Identifying Severe Alerts for Online Service Systems, *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 2420–2429 (online), DOI: 10.1109/INFOCOM41043.2020.9155219 (2020).

[9] Zhou, W., Tang, L., Li, T., Shwartz, L. and Grabarnik, G. Y.: Resolution recommendation for event tickets in service management, *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 287–295 (online), DOI: 10.1109/INM.2015.7140303 (2015).

[10] Zhu, X., Zhou, W. and Li, T.: A visual tool for ticket monitoring and management, *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 1–7 (online), DOI: 10.1109/ISKE.2017.8258805 (2017).

[11] Yamada, D., Kuwata, Y., Kasai, R. and Nakamura, T.: Automation of Message Handling in Cloud-based Managed Service, *Procedia Computer Science*, Vol. 22 (online), DOI: 10.1016/j.procs.2013.09.145 (2013).

[12] Savola, A.: Server virtualization with VMware (2021).

[13] Chen, L., Xian, M. and Liu, J.: Monitoring System of OpenStack Cloud Platform Based on Prometheus, *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pp. 206–209 (online), DOI: 10.1109/CVIDL51233.2020.0-100 (2020).

[14] Rabenstein, B. and Volz, J.: Prometheus: A {Next-Generation} Monitoring System (Talk) (2015).

[15] Meng, S. and Liu, L.: Enhanced Monitoring-as-a-Service for Effective Cloud Management, *IEEE Transactions on Computers*, Vol. 62, No. 9, pp. 1705–1720 (2013).

[16] Tang, L., Li, T., Shwartz, L. and Grabarnik, G. Y.: Identifying missed monitoring alerts based on unstructured incident tickets, *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pp. 143–146 (2013).

[17] Maksai, A., Bogojeska, J. and Wiesmann, D.: Hierarchical Incident Ticket Classification with Minimal Supervision, *2014 IEEE International Conference on Data Mining*, pp. 923–928 (online), DOI: 10.1109/ICDM.2014.81 (2014).

