障害発生時における解決ステータスの確認までのアラート抑制によるチケットの集約

山崎 拓海[1]　平尾 真斗[2]　串田 高幸[1]

概要：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 のシステム監視では, Prometheus と Alertmanager から通知されたアラートをチケット作成用のシステムでチケットにする. 課題は, 障害の発生時に冗長なチケットが作成されていることである. 同一のホストからの同じアラートに対して個別にチケットが作成されると, 運用者の対応負荷や視認性の低下を招き, 障害対応の遅延要因となる. 本研究では, 監視システムにおいて, 同じアラート名の過去のアラートのステータスが Resolved になっているかを確認し, チケット数を抑制する手法を提案する. アラート発生時に alertname と instance を取得し，過去に同名アラートが Resolved 状態になっているかを確認することで, チケットを発行するか, 発行しないかを制御する. 土曜日から翌週の金曜日までは, 状態を維持し, Resolved にならなかった場合はクローズし, 新しくチケットを作成する. 重複したチケットについては, 既存チケットのコメントに追記していくことで, 履歴を残していく. 同一アラートでホストが異なる場合は, 2 件目が登録されたところで, 共通のルートチケットを作成し, そこの子チケットに登録して集約していく. 重複したチケットについては, コメントに追記していく. 基礎実験では，2025 年の 6 月 16 日から 7 月 8 日に CDSL で通知されたアラートの 3191 件の内, Redmine にチケットが登録され, 1 次対応を行った 33 件のチケットを使用した. そこで重複しているチケットを調べ, 何件を集約できるか確認した. その結果, 33 件中 27 件が重複しているチケットとなり 6 件まで集約することができた.

1. はじめに

背景

EC サイトシステムは, 24 時間 365 日, 絶え間なく安定して稼働し続けることが求められており, その運用には高度な監視体制が不可欠となっている. これを実現するために, サーバやネットワーク機器, 各種アプリケーションの状態を常時監視し, 異常を早期に検知と対応するために, 監視システムと連携したアラート通知機能を導入している [1].

アラートはチケットに記録や追跡できる形式に変換されることで, 運用対応の可視化と体系化を実現している. この運用方法を採用することで, 運用担当者が誰でも対応履歴を参照できるようにし, 属人化の回避, 対応漏れの防止, 障害対応のスピード向上の効果をもたらしてくれる [2, 3]. さらに, アラートチケットはシステム障害のパターン分析や, インシデントの傾向把握にも活用できる. 特に, 同一ホストや同一サービスに関して繰り返し発生する問題を可視化し, 根本原因の分析につなげるための基礎データとしても機能する [4]. 運用現場においては, 発生したアラートを単発で処理するのではなく, チケットとして管理し体系的に扱うことが, 長期的な安定運用の鍵となる. 特に, 大規模なシステム環境ではアラートの頻度が高く, 同一障害に起因する多数のアラートが同時に発生することもある. 体系的に管理することで, 対応の重複や漏れを防ぎ, 根本原因に対する集中的な対応を実現する. また, インシデント対応の標準化した基盤でも, アラートチケットは運用の信頼性と効率性を支える基盤要素となる [5, 6].

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）では, 物理マシンが合計で 10 台稼働している. 物理マシンには, VMware が提供しているハイパーバイザーである VMware ESXi を導入している. VMware ESXi は, 物理マシンにインストールして仮想マシンを実行できるソフトウェアで, 1 つの物理マシン上で複数の仮想マシンを運用することができる. VMware ESXi により, 他の仮想マシンを実行するためのソフトウェアに比べて少ないリソースで運用でき, 物理マシンのリソースを効率的に使用できる. CDSL 内では, 所属している学生が実験に使用するマシンと, システム監視用, 外部公開サイト用の 3 つの目的で

[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
〒 192–0982 東京都八王子市片倉町 1404-1

運用されている．その他に，データを保存する NAS サーバが稼働している．

図 1 は CDSL 内のシステム構成図で，各マシンやシステムからのメトリクスを収集し，チケットが作成されるまでの流れを示している．

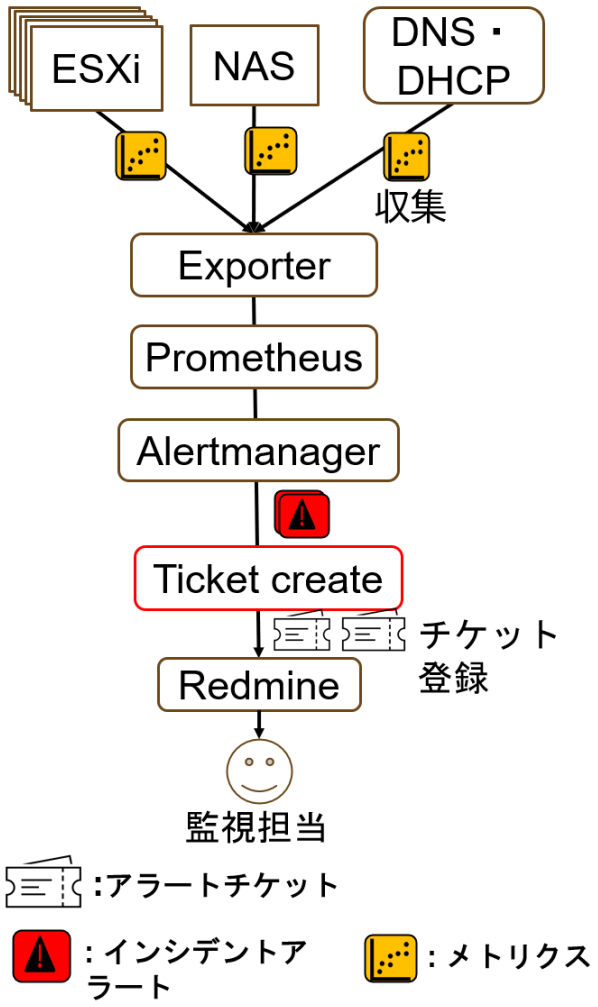

図 1: CDSL 内のシステム構成

CDSL で稼働しているマシンについては，オープンソースのシステム監視ソフトウェアである Prometheus を利用している [7]．それと組み合わせて，監視対象のシステムやアプリケーションからメトリクスを収集する Exporter を利用している．Exporter は各コミュニティが，対象のソフトウェアに合わせたものを公開しており，CDSL では VMware Exporter, Blackbox Exporter, Node Exporter を組み合わせている．各 Exporter は VMware ESXi や物理マシンからメモリやストレージの使用率，疎通確認を主に監視している．Exporter が収集したデータは Prometheus が定期的に取得し，Grafana で可視化されている [8,9]．システム運用においては，複雑化するインフラ環境に対応するために，Prometheus による時系列モニタリングと Alertmanager によるアラート通知が導入されている．また，アラート発生後の対応状況を可視化と記録する手段として，チケット管理ツールと連携し，アラートごとにチケットを登録する運用がある．CDSL でも Alertmanager からのアラートを受信し，Flask アプリケーション経由で Redmine にチケットを作成する仕組みが構築されている．

図 2 ではアラートチケット対応の流れを示している．CDSL では，原因調査や対応のためにエスカレーションし，2 次チケット，3 次チケットまで作成される仕組みをとっている．Redmine に登録されたチケットは，最初に 1 次調査チケットとして CDSL 内で監視業務を担当しているメンバーに割りてられる．アラートチケットが発行されると，定められている手順と Runbook に沿ってアラートについて調査を実施する．原因の特定や解決に至らなかった場合は，2 次調査としてそのシステムの管理を担当しているメンバーが調査する仕組みになっている．

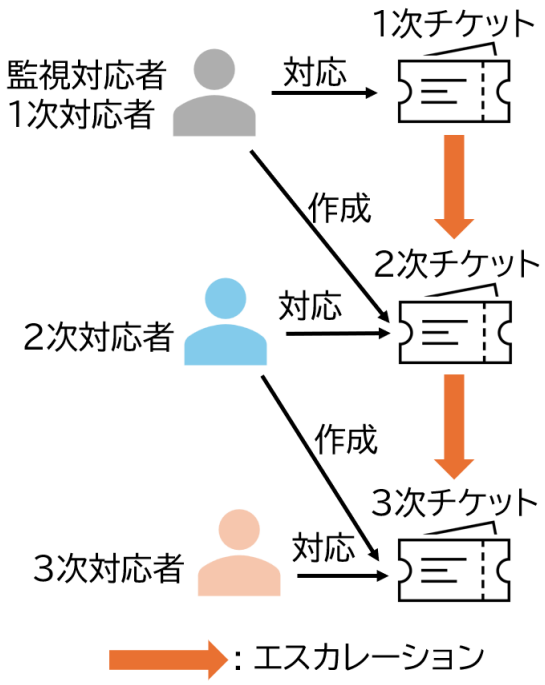

図 2: アラートチケット対応

課題

課題はアラートが通知された際に通知されたすべてのアラートをチケットとして登録してしまっていることである．図 3 は，同アラート，ホストのチケットが重複して作成されている様子を示す．

表 1 は，2025 年の 6 月 16 日から 6 月 27 日の期間にアラート Internal Archive Server ICMP-Check のチケットが作成された履歴である．CDSL 上で運用している Alertmanager は，ICMP-Check（疎通確認）が 10 分間できないとアラートとして通知する設定になっている．表 1 では，同一アラート，ホストによるアラートチケットの履歴を

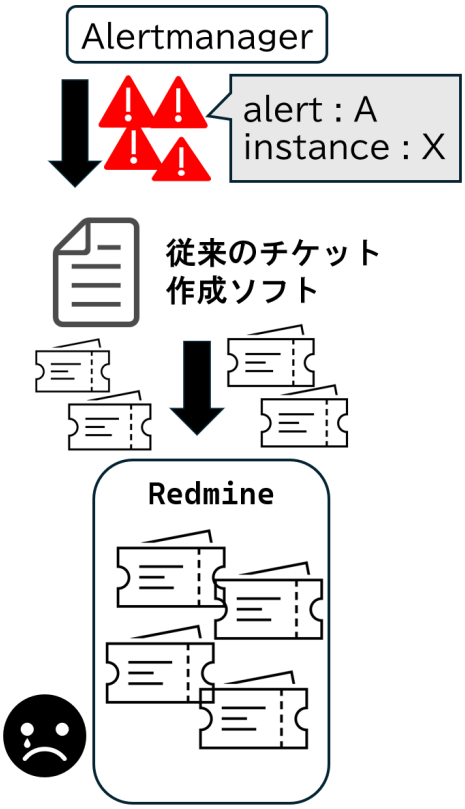

図 3: 同アラート・ホストのチケットが重複して作成される

示している．例を挙げると，Redmine に「[Alert] Internal Archive Server ICMP-Check」というチケットが 2025 年の 6 月 17 日の 11:23 に登録された．これは CDSL のアーカイブサーバマシンへの疎通確認が，監視対象に対してできなかったことによるアラートによるものである．このアラートチケットが登録された後，12:27, 14:08, 15:13 にも同名のアラートチケットが登録された．同一のアラートが短時間に複数回通知された背景に，Alertmanager 側の誤検知が要因として有力である．結果として，Redmine には同名のアラートチケットが 4 件登録された．

表 1: 同一アラート、ホストによるアラートチケット履歴

アラート名	ホスト	発行日
[Alert] Internal Archive Server ICMP-Check	achive-s	6 月 16 日 11:23
[Alert] Internal Archive Server ICMP-Check	achive-s	6 月 16 日 12:27
[Alert] Internal Archive Server ICMP-Check	achive-s	6 月 16 日 14:08
[Alert] Internal Archive Server ICMP-Check	achive-s	6 月 16 日 15:28
[Alert] Internal Archive Server ICMP-Check	achive-s	6 月 17 日 15:13
[Alert] Internal Archive Server ICMP-Check	achive-s	6 月 27 日 12:19

表 2 は，2025 年 6 月 16 日から 6 月 17 日の期間に，Internal HostMemoryUsageHigh-Usage90%のアラートによるチケットの登録履歴で，対象のホストも記載されている．異なるホストからの同一アラートの履歴を示している．アラートの基準の 1 つとして，ホストのメモリ使用率が高くなっていることを示すアラート「[Alert] Internal HostMemoryUsageHigh-Usage90%」が設定されている．これについて，2025 年の 6 月 16 日 11:46 を端に，11:51 に同時に 2 件，15:12, 15:13 と短期間で同じアラートチケットが登録された．このアラートについては，VM での原因ではなく，Alertmanager の誤検知によって通知された．実際には 1 つの障害原因が継続しているだけであっても，それぞれが独立したチケットとして登録されてしまっているのが問題となっている．

表 2: 異なるホストからの同一アラートの履歴

アラート名	ホスト	発行日
[Alert] Internal HostMemoryUsageHigh-Usage90%	canele	6 月 16 日 11:46
[Alert] Internal HostMemoryUsageHigh-Usage90%	mint	6 月 16 日 11:51
[Alert] Internal HostMemoryUsageHigh-Usage90%	makaron	6 月 16 日 11:51
[Alert] Internal HostMemoryUsageHigh-Usage90%	lily	6 月 16 日 15:12
[Alert] Internal HostMemoryUsageHigh-Usage90%	canele	6 月 17 日 15:13
[Alert] Internal HostMemoryUsageHigh-Usage90%	lily	6 月 17 日 17:27
[Alert] Internal HostMemoryUsageHigh-Usage90%	canele	6 月 17 日 17:27
[Alert] Internal HostMemoryUsageHigh-Usage90%	makaron	6 月 17 日 17:27

この結果，アラートに対応する担当者は，同一の原因にもとづく複数のチケットに対して，同じような対応作業を繰り返して行う必要があり，時間や人的リソースが無駄に消費されている．また，現行のチケット作成プログラムには，Redmine において元となるチケット，ルートチケットと子チケット関係を形成する機能が存在しないため，アラート間の関係性を構造的に表現することができず，チケットが一覧上に並列に羅列される状態となっている．これは，どのチケットが再発か，あるいは共通原因にもとづくものかを人手で判別しなければならず，確認や対応に余計な手間がかかっている．また，アラートが大量に発生した際にチケットの数が増え，視認性が著しく低下するとともに，対応優先度の判断が困難となり，結果的に障害対応の遅延や対応漏れが発生するリスクが発生する．

各章の概要

第 2 章の関連研究では, 関連する既存研究について述べる. 第 3 章の提案では, 課題を解決する提案とユースケース・シナリオについて述べる. 第 4 章の実装では, 提案をもとに作成したソフトウェアの実装方法について述べる. 第 5 章の評価実験では, 実験環境, 実験結果と分析について述べる. 第 6 章の議論では, 提案の議論を述べる. 第 7 章では, 全体のまとめを述べる.

2. 関連研究

通信業界におけるアラート管理の効率化を目的とし, アラートからトラブルチケット（TT）が起票されるかどうかを機械学習によって予測する手法を提案している論文がある [10]. 通信ネットワークでは, 機器障害や設定の不備により大量のアラートが生成されるが, そのすべてが実際の障害に直結するとは限らない, 従来の手動運用では, 不要なアラートに対する対応工数が増大し, 重大障害への対応が遅れるリスクがあった. 同研究では, 大手通信事業者から収集された実データを用いて, スライディングウィンドウを活用したアラートの時系列履歴を特徴量として抽出し, 分類器によりチケット起票の有無を予測するモデルを構築した. これにより, 誤検知文のと重要アラートの識別において有効であると確認しているが, アラートの本質的な原因究明には踏み込んでいない.

サポート部門における大量のチケット処理を対象に, マルチレベル分類（チケットの階層構造に応じた分類）の手法を体系的に整理している研究がある [11]. 20,000 件の顧客クレームと 35,000 件のバグレポートを含む 2 種類の実データセットを使用し, 事前学習済み BERT モデルによる文書埋め込みと階層情報の注入によって, F1 スコアおよび精度を最大で 28%以上, 向上させている. この階層構造を考慮した分類手法にすることで, チケット処理における高度な分類を実現している. 多くのシステムでは, チケットが単一のカテゴリに属するのではなく, 「大分類（カテゴリ）–中分類（サブカテゴリ）–小分類（詳細）」の多層的な分類体系で管理されている. これにより, 担当部署や処理手順の決定, ナレッジの検索効率化を実現する. 本研究では, BERT にもとづく事前学習済みモデルを利用した文書埋め込みを行い, チケットの本文情報から階層的なラベルを予測する分類モデルを構築している. これが 28%の F1 スコア向上に寄与しているが, シンプルスタックによる評価にとどまっており, オンコール管理システム, ログ収集による実用的な運用を想定した連携は検討されていない.

クラウドシステムにおけるチケット管理において, 内容が異なるにもかかわらず同一インシデントに起因する重複チケットが多数発生する課題に着目し, それらを集約する手法「iPACK（incident-aware duplicate ticket aggregation）」を提案している研究がある [12]. Microsoft Azure の大規模な実データを用い, アラート情報とチケット情報を二重リンク構造（alert–alert および alert–ticket）で接続することにより, テキストの表現が異なるチケット同士でもインシデントレベルでの集約ができるグラフベース手法を開発している. 具体的には, アラートの意味的, 時間的近接性から Graph-based Incident Profiling（GIP）を行い, さらに Attention ベースのニューラルネットワークにより各チケットがどのインシデントに属するかを分類している. その結果, 従来のテキストベース重複検出手法を大きく上回る F1 スコア（0.871〜0.935）を達成し, 最大で 31%の精度向上を実現している. これにより, アラートを軸にチケットを集約し分類する, 実運用に即したアプローチの有効性を示しているが, 全システムや多様なドメインに対する適用可能性の検証は行われていない.

自然言語処理技術の進展により, インシデントチケットの分類やラベリングに関する研究が進んでいる.Zhexiong Liu らによって提案された「Ticket-BERT」は, その代表的な研究の一つである [13]. Google が開発した文の意味や文脈を理解することができる自然言語処理モデル「BERT」をベースに使用し, インシデント管理チケットに対して高精度なカテゴリ付けを行う手法を提案している. Ticket-BERT は, 従来のルールベースや単純な機械学習モデルと比較し, 複雑な文脈や専門用語を含むチケット文書に対しても高い識別精度を示し, 実際の運用環境における分類作業の効率化に大きく貢献している. 特に, 複数部門が扱う異種のチケットを対象とする大規模 IT 運用環境において, 分類精度の向上がアラートのグルーピングや対応の迅速化にもつながるとされており, 本研究におけるチケットグルーピングの実現においても高い F1 スコアを示しているが, リアルタイム性能や実際の運用コストについては未評価となっている.

AI を活用し, パターン分析や依存関係の解析手法を提案する研究がある [14]. サーバ運用におけるアラートの多発や障害の複雑化する課題に対して, AI 技術を活用したリアルタイムのイベント相関分析および根本原因分析（Root Cause Analysis）を行う手法を提案している. 本論文では, 複数の層にまたがるログやアラート情報の膨大なデータを処理し, 運用者にとって重要なインシデントを即時に抽出する必要性を指摘している. 具体的には, アラートの洪水（alert fatigue）によって対応が追いつかない状況を解決するため, 機械学習やディープラーニングを用いた AI モデルを活用し, イベント同士の因果関係やパターンの検出を行うアーキテクチャとなっている. 異常な振る舞いや相関イベントを識別し, それらをもとに根本的な障害のを特定すると同時に Explainable AI（XAI）の観点から, AI による推論結果に対して人間が理解可能な説明を付加する設計が取り入れられており, ブラックボックス性の課題にも配慮している. 実験結果として, 平均復旧時間（MTTR）の短

縮や誤検知の集約する効果が確認されているが, 因果関係にもとづく根本原因の特定技術は取り入れておらず, 相関推定に近い評価にとどまっている.

3. 提案

提案方式

本稿では, Alertmanager の Resolved ステータスを活用してアラートチケットを集約することで, する手法を提案する. この提案で既存のチケット発行システムにおいて, アラートチケットが重複して作成されている問題や, 管理負荷の増大している問題を解決を図る. 土曜日から翌週の金曜日までは, 状態を維持し, Resolved にならなかった場合はクローズし, 新しくチケットを作成する. これは金曜日までに Resolved されなかったチケットが, 翌週に持ち越しされていることを明示的に把握できるためである. Resolved 通知を受信した際に, 関連するルートチケットと子チケットを完了ステータスに変更する機能を備えている. これにより, 担当者による手動のステータス管理作業を省略し, 対応完了の漏れを防ぐことを実現する.

同一アラートかつ同一ホストからのアラートチケットの場合

図 4 は, 同一アラートかつ同一ホストのパターンにおいて, アラートが通知されると, API がチケット管理ソフトウェアに問い合わせ, 条件に合わせてチケットを作成するパターンの流れを示している. 初回に作成されたアラートチケットは, 通常通りチケットとして登録される. それ以降に, 同一アラートかつ同一ホストのチケットが作成された際には, プログラムがチケットを探索し, 該当するチケットを発見した場合, 新たにチケットは発行しない. アラートが継続していることの記録や, 頻度の確認のために既存チケットのコメントに重複分はコメントに追記していく. ルートチケットが「Resolved（解決済み）」になるとチケットが Close（完了）となり, 後に新たなアラートが発生した場合は, 再びチケットを登録する.

同一アラート名で異なるホストの場合

図 5 は, 同一アラート名であるが異なるホストからのアラートの場合の流れである. 同一のアラート名であっても異なるホストからアラートが発生する場合がある. このパターンでは, そのマシンまたは VM より下位層である, スーパーバイザーや物理マシンで問題が発生しているか Alertmanager の誤検知により, 根本原因は同一である. この場合では, 共通ルートチケット「[Alert-Root] alertname」を作成してまとめていく. 1 件目のチケットについては独立したチケットとして登録する. その後同名のチケットが作成され, ホストが異なっていた場合, 共通ルートチケットを作成し, 1 件目のチケットと同時に子チケットとして登録していく. もし, 同じホストからのアラートがでた場合は, チケットを登録しないことでチケットを集約していく. ルートチケット内のアラートが Resolved になった場合は完了にする. その後に同じホストからアラートがでた場合は再びチケットを登録する. 共通ルートチケット内の子チケットが全て Resolved になり, チケットが完了したらルートチケットも完了とする.

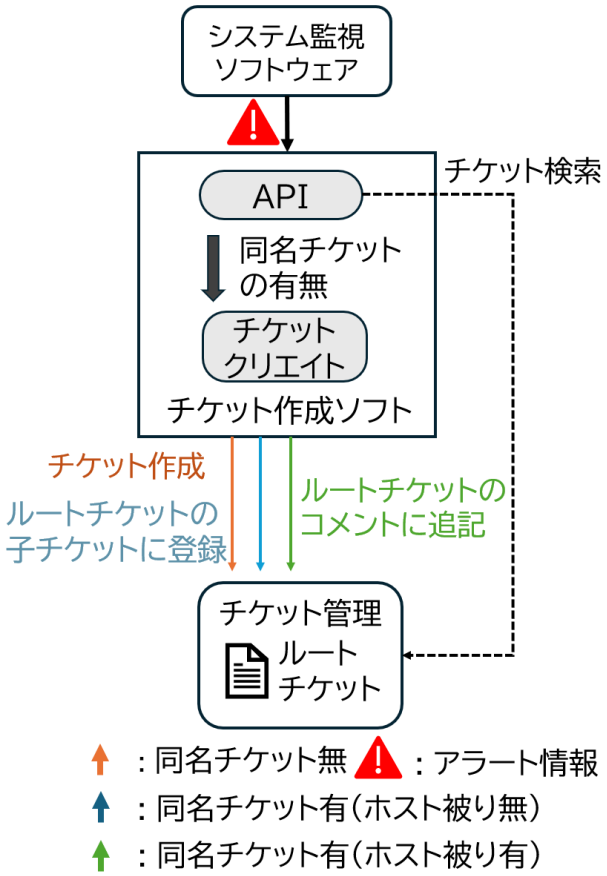

図 4: 提案ソフトウェア概要

ユースケース・シナリオ

本稿では, CDSL で運用されている監視システムにおけるチケット登録の仕組みをユースケースとする. CDSL では, Prometheus に事前に設定されたルールにのっとり, しきい値アラートをこえるとアラートを発報し, Alertmanager が通知を担当する. Alertmanager からチケット作成プログラムにアラート通知が送られるとその都度チケットが Redmine に登録され, 監視担当者がそのアラートチケットの調査を定められた Runbook に沿って実施する. 従来のシステムでは, 同一原因に起因する複数のアラートが発生しても全てがチケットとして登録されるため, 同じ調査を複数回行う必要があり, 人的リソース, 時間的リソースどちらも非効率的で, 対応工数が増加していた.

図 5 では, ユースケースシナリオとして, 重複するチケットを作成せずに, チケットの総数を集約していることを表

している．提案するシステムを導入することにより，同一の原因により発行されたと考えられる同名のチケットを Redmine の親子関係システムを利用し，集約したうえで，チケットの総数も減らすことができる．これにより，従来では同一の原因によるアラートでも，個別に調査を行っていたが，それを 1 回で完了できる．また，最初の調査で原因の特定や障害の解消ができなかった場合は，そのシステムの管理者がさらなる調査を行っているため，これらの工数も減らすことができ，迅速かつ効率的な障害対応を実現できる．

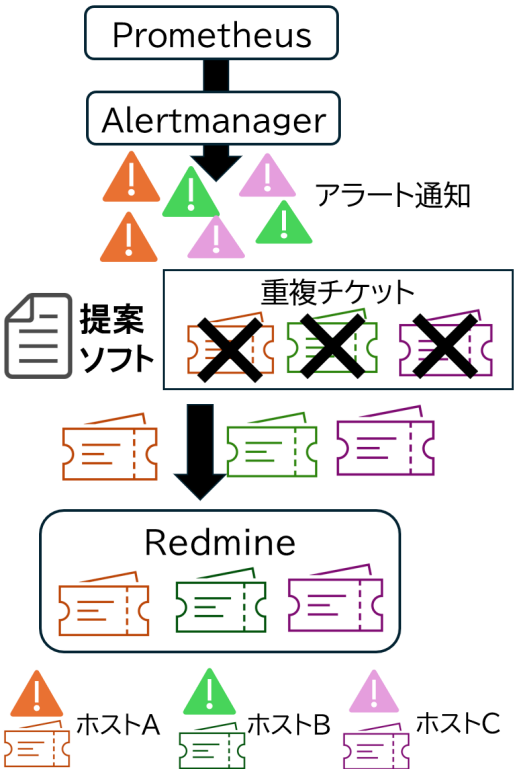

図 5: ユースケースシナリオ

4. 実装

本稿では，提案方式をもとに，Python3 (ver. 3.12.3) で作成したチケット作成プログラムについて説明する．図 6 では，プログラムとソフトウェアとのやり取りの流れを表す，実装プログラムの概要を表している．本システムでのプログラム構成は，単一の FastAPI アプリケーションファイルと，必要に応じた起動スクリプト，環境変数ファイル（.env）で構成されている．Prometheus および Alertmanager により通知された，alertname，instance を，Python アプリケーション（ticket_create.py）が受け取る．Redmine に対して REST API でチケットの重複を確認し，チケット登録の可否を判断する．

アラートを受信すると，プログラムは JSON 形式の情報から alertname および instance（ホスト名）を抽出し，Redmine 上に既存の未解決チケットが存在するかを API を介して照会する．存在しない場合は新規に独立したチケットとして登録する．既存チケットが存在する場合は，当該のチケットをルートチケット（Redmine 上では親チケット）とし，新たに作成されたものを子チケットとして登録する．同一アラート名で異なるホストからのアラートが通知された場合は，共通のルートチケットを作成し，既存チケットおよび新規チケットをその子チケットとして登録していく．アラートの status が resolved であることを受信した際は，alertname および instance の組み合わせに該当する Redmine チケットを検索し，それらを完了ステータスに更新する処理を行う．

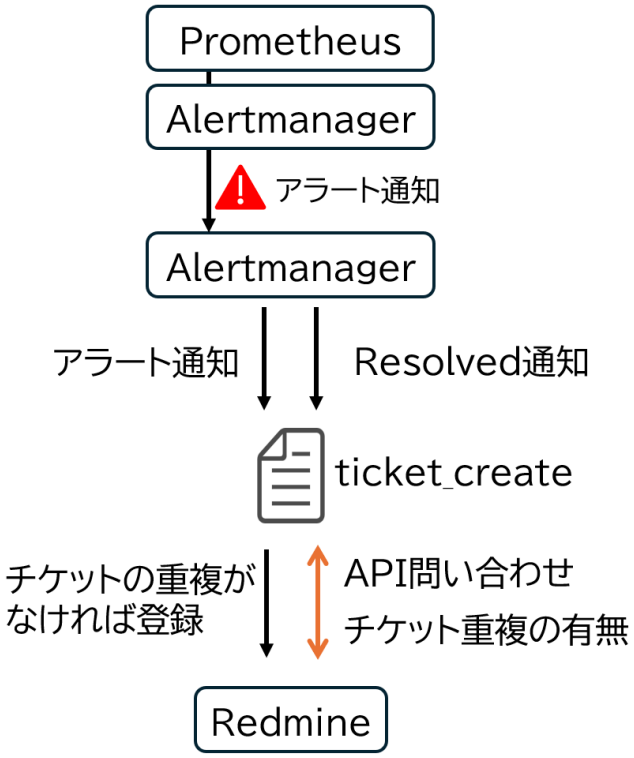

図 6: 実装プログラムの概要

5. 評価実験

評価実験では，2025 年の 6 月 16 日から 7 月 8 日に，CDSL で対応が行われたチケットについて，提案手法のチケット作成システムを使用すると，どれだけ集約できるかを評価する．

実験環境

図 7 に実験を行う環境の構成図を示す．

実験を行う環境として，監視システム，チケット作成システム，チケットシステムを用意する．提案ソフトウェアを動かす VM のスペックは，vCPU が 2 コア，メモリが 4GB，ストレージが 30GB となっている．監視対象は，物理マシ

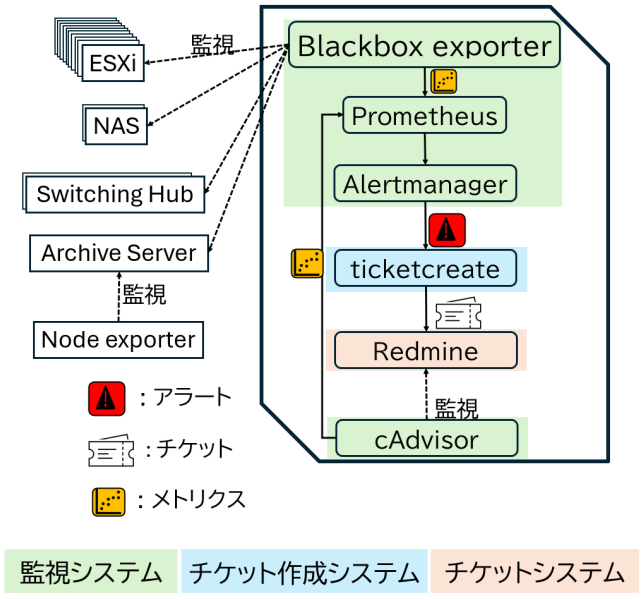

図 7: 実験環境の構成図

ンのほかに, ネットワーク機器と物理マシンにインストールされた VMware ESXi に配置された VM がある. 監視システムは, 監視ソフトウェアに Prometheus, エージェントは Blackbox exporter, Node exporter, cAdvisor, Metrics Server を使用する. アラートの通知には Alertmanager を使用する. Alertmanager から通知されたアラートは, ticketcreate がアラートチケットを作成し, Redmine に登録していく. 監視システム，チケット作成システム，チケットシステムは K3s を使用した Kubernetes クラスタ上に配置している.

VMware ESXi をインストールした物理マシンは 10 台ある. その内訳は, 研究室で実験を行う用途の物理マシンが 7 台，DNS や DHCP，Jump サーバを運用する基幹用の物理マシンが 2 台，外部にサイトを公開するための用途の物理マシンが 1 台となっている. NAS は 2 台あり，実験用の VMware ESXi の VM データを保存するストレージとして使用する. VMware ESXi の電源管理のために SwitchBot を導入している. 1GB の Swichinghub は VMware ESXi に対して通信を行うためのネットワークを経由する目的で使用する. 10GB の Swichinghub は VMware ESXi と NAS を用いたストレージ間のネットワークを経由する目的で使用する. 実験と本番環境間のファイアウォールは実験用の環境と本番環境で疎通を行う際に経由する. Rider 送信用 Raspberry Pi は Rider が取得したデータを本番環境上の VMwareESXi である Iris 上の VM に送信するために使用される. アーカイブ用の Ubuntu は実験環境で使用している VM のデータの中で登録が解除されたデータをアーカイブする目的で使用している.

実験結果と分析

表 3 は, 2025 年 6 月 16 から 7 月 8 日の期間に, CDSL の Redmine に同じ原因によるアラートチケットで, 対応を行ったチケットの数である.

表 3: チケットの集計結果

アラート名	期間	件数
[Alert] Internal HostMemory UsageHigh-Usage90%	6 月 16 日 11:46〜6 月 16 日 15:13	5
[Alert] Internal MonitoringCluster MemoryUsageHigh	6 月 16 日 16:02〜6 月 17 日 9:50	7
[Alert] Internal Monitoring Cluster Abnormal pod status	6 月 17 日 11:27〜6 月 17 日 12:38	4
[Alert] Internal Archive Server ICMP-Check	6 月 17 日 14:26〜6 月 17 日 15:13	2
[Alert] Doktor Dev Nod Filesystem Average Usage High 80%	6 月 25 日 12:35〜6 月 25 日 12:35	3
[Alert] External Clematis Node Memory Usage High 90	6 月 25 日 13:19〜7 月 8 日 10:36	12

本研究では, Alertmanager の通知処理において, 同一アラートについては Resolved ステータスを受信するまでは新たなチケットを発行しないことで, チケットを集約することを目標としている. 実際に登録されたからどれだけのチケット数を集約できるかを定量的に評価した.

CDSL では 6 月 16 日から 7 月 8 日の間に 3191 件のアラートが通知された. この評価では, Resolved ステータスによる通知を除外した, 期間中に Redmine にチケットとして登録されたアラートの中で, 1 次対応を行った 33 件のチケットをもちいる. Internal HostMemoryUsageHigh-Usage90% については, 6 月 16 日 11:46 から 6 月 16 日 15:13 の間に 5 件発生した. Internal MonitoringCluster MemoryUsageHigh については, 6 月 16 日 16:02 から 6 月 17 日 9:50 の間に 7 件発生した. Internal Monitoring Cluster Abnormal pod status については, 6 月 17 日 11:27 から 6 月 17 日 12:38 の間に 4 件発生した. Internal Archive Server ICMP-Check については, 6 月 17 日 14:26 から 6 月 17 日 15:13 の間に 2 件発生した. Doktor Dev Node Filesystem Average Usage High 80% については, 6 月 25 日 12:35 から 6 月 25 日 12:35 の間に 3 件発生した. External Clematis Node Memory Usage High 90 については, 6 月 25 日 13:19 から 7 月 8 日 10:36 の間に 12 件発生した.

それぞれのアラートにおいて重複しているチケット数を確認すると合計で 27 件あり, 約 82%集約できることが確認できた. これにより, 監視担当者および 2 次対応者以降

の負担を軽減できる.

6. 議論

アラートのグルーピングは主に alertname および instance をキーとした単純な一致判定を基準としている. 一方で, 異なるアラート名であっても, 同一原因による事象である可能性も存在する. たとえば「CPU 使用率異常」と「メモリ使用率異常」が同一ホスト上で同時発生した場合, それらは同一障害の一部と解釈できる. 過去の障害パターンにもとづく類似度スコアや時系列分析により, アラートの自動判定を行うことができる.

アラートの定義が異なるにもかかわらず, 実質的に同じ障害事象を指すケースが存在する.「Disk usage over 90%」や「ファイルシステム容量が閾値を超過」は, 異なる表現ながら同一の障害内容を意味している. このような場合, 現行システムでは異なる alertname として処理され, 別々の親チケットとして登録されてしまう. そこで, アラートの annotations.description や labels.message を使用した自由文を対象に, BERT や Sentence BERT の事前学習済み言語モデルを用いた埋め込み処理を行い, 類似度の高いアラート同士をクラスタリングする手法が有効となる. これにより, 言語表現に依存せずに意味的に類似したアラートを 1 つの障害グループとして扱うことができ, さらなるチケット統合と対応効率の向上をできる. 本手法は, 既存の Redmine 側の構造や Prometheus のアラート定義を大きく変更する必要がなく, 前処理としてクラスタ ID を付加するだけで実装できる.

7. おわりに

CDSL では, システムで発生したアラートを Prometheus, Alertmanager を通じてアラートチケットを Redmine に登録している. その一方で, 同じ原因から通知され, 作成されたアラートチケット Redmine に重複して登録され, アラートへの対応の効率低下や, チケット乱立による視認性の低下を招いている. 本研究では, アラートチケット管理の効率化を目的として, 同一アラートからのチケットの集約をするアラートチケット作成プログラムを提案する. プログラムは, Alertmanager からアラートが通知されると API を介して Redmine に, 同名のアラートがないかを問い合わせる. 同名のアラートがなければ, なければチケット作成し登録する. 同名のアラートであるがホストが異なる場合は, 2 つめのチケットが作成されたところからルートチケットを作成し, グルーピングしていく. チケットが存在する場合は, 作成を見送る. 評価実験では, 2025 年の 6 月 16 日から 7 月 8 日に CDSL の Redmine に登録された, 同じ原因から作成されたチケットを確認した. その中で 1 次対応を行った 33 件のチケットをもちいた. その結果, 33 件の約 82% にあたる 27 件のチケットを集約できることを確認した.

謝辞

本稿の執筆にあたり, アラート件数のカウントにご協力いただきました, 東京工科大学コンピュータサイエンス学部の佐藤健斗さん, 坂井萌桜さんに御礼申し上げます.

参考文献

[1] Ling, C., Xiaoming, W., Ping, X., Yong, Z., Xuezhi, Z., Xianzhou, C., Mengyao, F., Yueqi, G. and Shuang, X.: Based on Zabbix-Prometheus Group Classification Alarm System, *Journal of Physics: Conference Series*, Vol. 2665, p. 012010 (online), DOI: 10.1088/1742-6596/2665/1/012010 (2023).

[2] Aglibar, K., Alegre, G., del Mundo, G., Duro, K. and Rodelas, N.: Ticketing System: A Descriptive Research on the Use of Ticketing System for Project Management and Issue Tracking in IT Companies (2022).

[3] Aglibar, K. and Rodelas, N.: Impact of Critical and Auto Ticket: Analysis for Management and Workers Productivity in using a Ticketing System, *International Journal of Computing Sciences Research*, Vol. 6, pp. 988–1004 (online), DOI: 10.25147/ijcsr.2017.001.1.84 (2022).

[4] Montgomery, L. and Damian, D.: What do Support Analysts Know About Their Customers? On the Study and Prediction of Support Ticket Escalations in Large Software Organizations, *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 362–371 (online), DOI: 10.1109/RE.2017.61 (2017).

[5] Salah, S., Maciá-Fernández, G. and Díaz-Verdejo, J. E.: Fusing information from tickets and alerts to improve the incident resolution process, *Information Fusion*, Vol. 45, pp. 38–52 (online), DOI: https://doi.org/10.1016/j.inffus.2018.01.011 (2019).

[6] Montgomery, L. and Damian, D.: What do support analysts know about their customers? on the study and prediction of support ticket escalations in large software organizations, *2017 IEEE 25th international requirements engineering conference (RE)*, IEEE, pp. 362–371 (2017).

[7] Jani, Y.: Unified Monitoring for Microservices: Implementing Prometheus and Grafana for Scalable Solutions, *Journal of Artificial Intelligence, Machine Learning and Data Science*, Vol. 2, pp. 848–852 (online), DOI: 10.51219/JAIMLD/yash-jani/206 (2024).

[8] B.C., P., Maddirala, H. and M., S.: Implementing an effective Infrastructure Monitoring Solution with Prometheus and Grafana, *International Journal of Computer Applications*, Vol. 186, pp. 7–15 (online), DOI: 10.5120/ijca2024923873 (2024).

[9] Jani, Y.: Unified monitoring for microservices: Implementing prometheus and grafana for scalable solutions, *J Artif Intell Mach Learn & Data Sci*, Vol. 2, No. 1, pp. 848–852 (2024).

[10] Asres, M. W., Mengistu, M. A., Castrogiovanni, P., Bottaccioli, L., Macii, E., Patti, E. and Acquaviva, A.: Supporting Telecommunication Alarm Management System With Trouble Ticket Prediction, *IEEE Transactions on Industrial Informatics*, Vol. 17, No. 2, pp. 1459–1469 (online), DOI: 10.1109/TII.2020.2996942 (2021).

[11] Yudha Erian Saputra, M., Noprianto, Noor Arief, S., Nur Wijayaningrum, V. and Syaifudin, Y. W.: Real-Time Server Monitoring and Notification System with Prometheus, Grafana, and Telegram Integration, *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent*

Systems (ICETSIS), pp. 1808–1813 (online), DOI: 10.1109/ICETSIS61505.2024.10459488 (2024).

[12] Liu, Z., Benge, C. and Jiang, S.: Ticket-BERT: Labeling Incident Management Tickets with Language Models (2023).

[13] Liu, J., He, S., Chen, Z., Li, L., Kang, Y., Zhang, X., He, P., Zhang, H., Lin, Q., Xu, Z., Rajmohan, S., Zhang, D. and Lyu, M. R.: Incident-aware Duplicate Ticket Aggregation for Cloud Systems (2023).

[14] Liang, W.: Real-Time Event Correlation and Root Cause Analysis in AI-Powered Server Monitoring (2025).

