物理サーバとNodeとPodの配置関係の一致判定をもとにしたグルーピングによるチケット件数の削減

平尾 真斗[1]　串田 高幸[1]

概要：Cloud and Distributed Systems Laboratory では，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node や Pod を Prometheus をもちいて監視している．監視ルールで設定した監視対象のメトリクスの値が閾値を超えた際，Alertmanager がアラートを通知し，Redmine にチケットとして登録する．課題は，通知されたアラートを Redmine に登録する際に，同一の障害が原因で通知されたアラートをまとめてチケットとして登録しなければ，チケット件数が増加することである．提案では，アラートが通知された際に，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係をチケットに記録する．例えば，Kubernetes クラスタ上の Node の Filesystem が 40GB のうち 36GB 使用され閾値である 90%を超えたアラートが通知され，チケットが登録された際，Node 上に配置されている Pod 名をそのチケットに記録する．その後，最初に登録されたチケット内に記録された Pod 名のアラートが通知された際は，最初に登録されたチケットに関連したチケットとして登録する．評価では，Kubernetes クラスタ上の Node のメモリ使用量が 8GB 全て使い切り Pod がダウンした事例，Node で使用される Filesystem の容量が 40GB のうち 36GB 使用され使用率が 90%を超えた事例，物理サーバに対して Blackbox exporter による疎通確認ができなくなった事例の 3 つを再現する．評価指標は，監視対象ごと，アラート名ごと，Job ごと，配置関係ごと，提案手法の 5 種類で登録されたチケットの件数を比較した．評価の結果，Kubernetes クラスタ上の Node のメモリ使用量が 8GB 全て使い切り Pod がダウンした事例では，提案手法は，チケット件数が 9 件となりチケットをまとめて登録することができなかった．Node で使用される Filesystem の容量が 40GB のうち 36GB 使用され使用率が 90%を超えた事例では，提案手法は，チケット件数が 1 件となり登録されるチケットをまとめることができた．物理サーバに対して Blackbox exporter による疎通確認ができなくなった事例では，提案手法は，チケット件数が 14 件となりまとめることができなかった．また，2025 年 12 月 4 日から 12 月 8 日までで登録されたチケットの件数を提案手法ありとなしで比較した．2025 年 12 月 4 日で登録されたチケットは，提案手法なしで 124 件，提案手法ありで 55 件となりチケットをまとめることができた．一方で，2025 年 12 月 5 日から 12 月 8 日は登録されたチケット件数が，提案手法ありとなしで変わらなかった．

1. はじめに

背景

企業で運用されているシステムは，数百から数千のコンポーネントで構成され，使用するユーザをサポートする [1]．このようなシステムでは，ユーザに対して 24 時間 365 日安定した動作を提供することが求められる [2]．障害発生時に迅速に対処と復旧ができなければ，企業は金銭的なコストを負う可能性がある．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]．

システム障害の迅速な解決は，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につながる．監視は，システムの運用運用を円滑に行うためにもちいられる [4,5]．監視システムはエージェントに対してメトリクスを要求する．監視エージェントは対象を監視し，メトリクスを監視システムに提供する [6–8]．対象に障害が発生した際や事前に障害の兆候が見られた場合，監視システムはアラートをオンコールエンジニアに対して通知する [9,10]．アラートが通知されるルールは管理者によって設定され，閾値を超えた場合に異常の判定を行う [11]．アラート内には，タイムスタンプ，ソース，内容の 3 つの要素が含まれている [12,13]．通知されたアラートはチケットシステムに登録される．また，この内容に加えオンコールエンジニアは Runbook と呼ばれる手順書をもとに調査を行う [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

企業では，システム障害を迅速に解決するために，障害対応を段階的に進める体制を整えている [15]．オンコールエンジニアがアラートを受信し，初期調査と 1 次対応を行う．所定時間内に原因の特定や復旧が困難な場合には，より専門的な知識を持つ 2 次調査担当者へとエスカレーションされる．また，問題が複雑でシステム全体への影響が大きい場合には，3 次対応としてアプリケーションの開発者や管理者が調査と対応を引き継ぐ．このように，障害の性質や難易度に応じて段階的に対応者が切り替わる体制により，迅速かつ的確な問題解決が図られている．

監視システムを使用している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) がある．図 1に，CDSL で稼働しているシステムと障害発生時にアラートが通知されてから 1 次調査者が調査を行うまでの流れを示す．

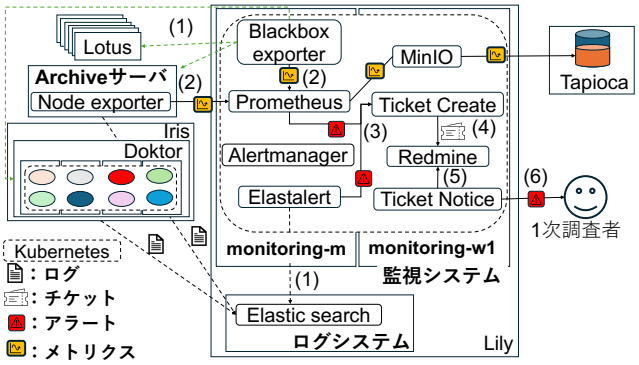

図 1: CDSL で稼働しているシステムと障害発生時にアラートが通知されてから 1 次調査者が調査を行うまでの流れ

CDSL には，米ブロードコム社が提供する仮想化ソフトウェア VMware ESXi がインストールされた 10 台の物理サーバが存在する．これらの物理サーバは，開発環境と本番環境の 2 つの環境に分けて運用されている．図中の Lotus と Lily は開発環境上に配置された物理サーバであり，合計 9 台の物理サーバが開発環境に属している．Iris は本番環境上に配置された物理サーバである．Tapioca は，ESXi 物理サーバ上の仮想マシンがもちいるストレージ領域を提供する NAS である．これらのサーバ群は，CDSL 内で行われる各種実験，検証，および実運用サービスを支える基盤として使用されている．Iris 上には，Doktor と呼ばれるマイクロサービスが稼働しており，4 つの Node 上に構成されている．Archive サーバは開発環境側で運用されており，研究室の学生が使用を終了した仮想マシンのデータを長期的に保存する目的で使用されている．これにより，不要になった仮想マシンのデータを安全に保存し，必要に応じて再使用できるようになっている．監視システムは CDSL 環境全体の監視するための統合的な基盤であり，Kubernetes クラスタを使用した 2 台の Node で構成されている．monitoring-m は Kubernetes クラスタのマスター Node である．monitoring-w1 はワーカー Node である．その構成要素の 1 つである Blackbox exporter は監視エージェントとして動作し，ESXi，Doktor の各 Node，および Archive サーバに対して疎通確認を行う．監視対象のプロトコルは ICMP，SSH，HTTP の 3 種類であり，これによりネットワーク層からアプリケーション層までの幅広い監視を実現している．また，Node exporter は Archive サーバ上で稼働し，各 Node の CPU 使用率，メモリ使用量，および Filesystem の空き容量の使用状況を収集している．Exporter によって取得されたメトリクスは，Prometheus によって定期的に収集される．Prometheus は監視サーバとして，取得したメトリクスを時系列データベースに保存し，閾値にもとづいた異常判定を行う．異常値が検出された場合，Prometheus は Alertmanager に対してアラートの通知を指示する．MinIO は Prometheus のメトリクスのデータをアーカイブするデータベースであり，Tapioca のボリュームに保存する．Alertmanager は Prometheus から受信したアラートを TicketCreate 通知する．TicketCreate は，Alertmanager から受信したアラートをもとに Redmine 上にチケットを登録する役割を担っている．TicketCreate によって登録されるチケットには，1 次調査者の ID が含まれている．Redmine はチケット管理システムとして機能し，TicketCreate によって登録されたチケットをデータベース上に保存する．Ticket Notice は Redmine 上で新規チケットが登録された際，チケットの更新の有無を確認した上で，1 次調査者にアラート通知を行う．ログシステムは，CDSL 上で運用されているサーバやアプリケーションのログを収集するための基盤である．Elasticsearch は ESXi や，Doktor，Archive サーバから送信されたログを保存する．Elastalert は Elasticsearch のログを元にアラートを TicketCreate に通知する．このように，監視エージェントによる監視の開始から，1 次調査者へのアラート通知に至るまでの処理の流れは，図中の (1) から (6) で示される．(1) は Blackbox exporter および Node exporter が各対象のメトリクスを取得する過程を示す．また，Elastalert は Elasticsearch に保存されたログの検索を行う．(2) は Prometheus が Exporter からメトリクスを収集する動作と Elastalert が Elasticsearch に検索を行いクエリにあったログの有無を確認する様子を示す．(3) は Prometheus の異常判定にもとづき，Alertmanager が TicketCreate にアラートを通知する処理を示す．また，Elastalert も同様に TicketCreate にアラートを通知する．(4) は TicketCreate が Redmine 上に新しいチケットを登録する過程を表している．(5) は Ticket Notice が Redmine 上のチケットを監視し，新しいチケットが登録されたかどうかを判断する工程を示す．(6) は Ticket Notice が 1 次調査者にアラート

を通知することを示す．この一連のプロセスにより，システム全体での障害検知からチケットの登録，1 次調査者への通知までが行われる．CDSL の監視作業は，4 交代制で行っており，9：00 から 10：30，10：30 から 12：30，12：30 から 15：00，15：00 から 17：30 となっている．また，監視作業中に障害が発生し，チケットが登録された際に該当の時間に監視作業を行っている担当者が調査を行う．調査開始時に最初に行うこととして監視作業を行っていない時間に登録されたチケットや，障害が発生した際のチケットをまとめる作業である．

障害が発生してから調査が行われた事例として 2025 年 9 月 26 日 (金)9 時 51 分に通知された本番環境上に配置された物理サーバである Iris に対して疎通確認ができなくなった事例があげられる．Iris で起きた障害の事例を図 2に示す．

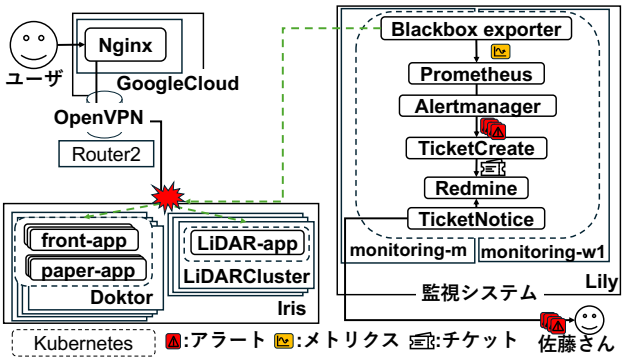

図 2: Iris で起きた障害の事例

ユーザは Iris 上に配置された Doktor のエンドポイントにアクセスを行う．Nginx は GoogleCloud 上に配置されており通信の経由を行う．Router2 は本番環境で使用されるルータである．それぞれの経路の間は OpenVPN を経由している．Iris は研究室の本番環境として使用される物理サーバである．Doktor は本番環境上で動作するマイクロサービスである．Doktor を構成するクラスタは 4 つの Node で構成されている．Node 上には K3s をもちいたクラスタが配置されている．front-app は Doktor の Web の UI を表示するための Pod である．paper-app は論文を保存するための Pod である．LiDARCluster は研究室上で運用されている LiDAR のデータを保存するためのクラスタであり，3 つの Node で構成されている．LiDAR-app は LiDAR が送ったデータを保存する Pod である．Firewall は開発環境と本番環境を通信の制御をする．SW-Local1 は Firewall と Lily を繋ぐスイッチングハブである．Lily は研究室の開発環境で運用される物理サーバである．監視システムは Lily 上に配置された K3s を使用したクラスタであり，monitoring-m と monitoring-w1 の 2 台の Node で構成される．Blackbox exporter は Iris，LiDARCluster を構成する 3 つの Node，Doktor を構成するクラスタは 4 つの Node に対して疎通確認を行う．Blackbox exporter が行う疎通確認は，ICMP，SSH，HTTP のプロトコルを使用する．Prometheus は監視サーバであり，Blackbox exporter の監視したメトリクスを収集する．また，事前に設定された閾値が異常値となった際にアラートとして通知する．Alertmanager は Prometheus が異常値としてアラートを発行した際に通知する役目を持つ．TicketCreate は，Alertmanager から受信したアラート情報をもとに Redmine 上にチケットを生成する役割を担っている．Redmine はチケット管理システムとして機能し，TicketCreate によって登録されたチケットをデータベース上に記録し保存する．Ticket Notice は Redmine 上で新規チケットが登録されたことを監視し，更新の有無を確認した上で，1 次調査者にアラート通知を行う．佐藤さんは CDSL に所属する研究室生であり，2025 年 9 月 26 日 (金) にアラートの対応を行っていた．2025 年 9 月 26 日 (金) の事例では，ESXi の Iris 自体に対する疎通ができなくなったため，LiDARCluster を構成する 3 つの Node，Doktor を構成するクラスタは 4 つの Node も同様に疎通確認ができなくなった．その結果，14 件のアラートが Alertmanager から通知され，Redmine 上にチケットが登録された．また，Redmine 上に登録されたチケットは，Alertmanager のアラート 1 件につき 1 件となるため，14 件登録された．

アラートをまとめてチケットに登録する場合

Iris の事例の場合，障害の発生箇所である Iris のアラートをチケットとして，それ以外の Node から登録されたチケットは関連するチケットにすることで，障害の発生箇所が明確になる．図 3に Iris で起きた障害のアラートをチケットにする際のまとめ方を示す．

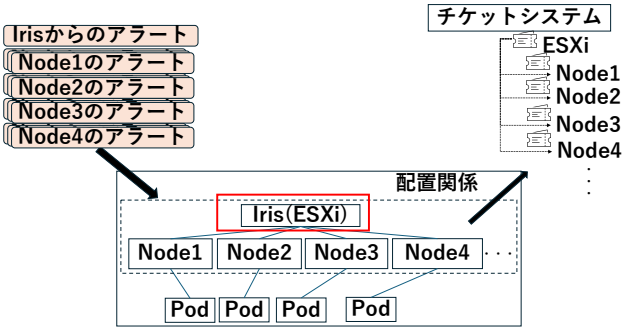

図 3: Iris で起きた障害のアラートをチケットにする際のまとめ方

Iris からのアラートは物理サーバである Iris から通知された ICMP の疎通ができなかったことを示すアラートである．Iris が疎通できなくなった事例では，Iris 以外にも Node のアラートも通知された．配置関係は Node や Pod がどの対象に配置されているかを示す．例えば，Node1 は，

ESXi である Iris に配置されている．アラートが通知された際にチケットのまとめ方として，アラートが通知された対象と完全にマッチする配置関係を選択し，木構造の根になる部分をチケットとすれば，チケットをまとめることができる．例えば．Iris の事例では，配置関係上で根の位置にあたる Iris をチケットとして Iris 上に配置されている Node 全てを関連するチケットにすれば，障害発生時に対処すべきチケットが明確になる．

アラートをまとめてチケットとして登録できない事例

障害発生時にチケットをまとめられない事例として，障害のアラートが通知された後，時間が経過してからアラートが通知される事例がある [16]．図 4に障害発生時にチケットをまとめられない事例を示す．monitoring-m

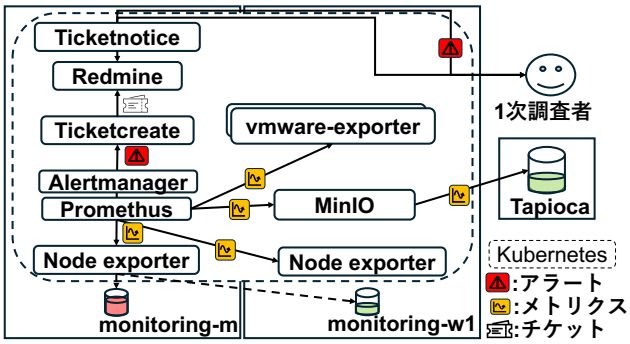

図 4: 障害発生時にチケットをまとめきれない事例

と monitoring-w1 は CDSL の監視を行うサーバを構成する Node である．monitoring-m 上には Ticketnotice，Ticketcreate，Redmine，Alertmanager，Prometheus，Node exporter が配置されていた．monitoring-w1 上には，vmware exporter と Prometheus のデータをアーカイブする MinIO，Node exporter が配置されていた．Tapioca 上には MinIO の上に保存された Prometheus のメトリクスのデータを保存するためのボリュームがある．11 月 7 日 (金)23 時 48 分に monitoring-m の Filesystem が 40GB 中 36GB 使用していたことで，Prometheus 上で設定されている閾値の 90%を超過した．その際に通知されたアラートは，Redmine 上にチケットとして登録された．このチケットは CDSL の監視作業の時間外だったため，対処されなかった．その後，11 月 8 日 (土)15 時 21 分に monitoring-m 上の Pod が DiskPressure でダウンし，6 件のチケットが Redmine 上に登録された．その結果，monitoring-m の Filesystem が 90%を超えたアラートと Pod がダウンしたアラートが別々に分けて通知され，チケットとして登録された．(1) は monitoring-m の Filesystem の容量が 90%を超えた際にアラートが通知されたことを示す．(2) は (1) のアラートが通知され，アラートがチケットとして登録されたことを示す．(3) は Filesystem の 90%を超えた影響で monitoring-m 上に配置された Pod がダウンしたことを示すアラートが通知されたことを示す．(4) は (3) で通知されたアラートがチケットとして登録されたことを示す．

課題

課題は，通知されたアラートを Redmine に登録する際に，同一の障害が原因で通知されたアラートをまとめてチケットとして登録しなければ，チケット件数が増加することである．図 4で解説した事例では，File-system の容量が 40GB 中 36GB を使用したことで，Pod が DiskPressure となった．Iris の事例のように障害が発生した時間にアラートがに一斉に通知される場合，障害箇所を監視対象である ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係ごとにチケットをまとめることができる．一方で，障害発生時に Kubernetes クラスタ上の Node や Pod のアラートが別々の時間帯で通知される場合，それぞれのアラートは独立した障害としてチケットシステムに登録される．その結果，同一の障害に起因するアラートであっても別々のチケットとして扱われ，チケットの件数が増加する．チケットの件数が増加すると，運用者はチケットを手動でまとめるために追加の作業時間を要する．また，本来は同一の障害として扱うべきアラートであっても，個別のチケットとして登録されてしまうことで，調査者が障害の全体像を把握しにくくなり，チケット統合の判断や作業を妨げる要因となる [7]．

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

障害発生時にクリックされたアラートから重要なチケットかどうかを分析し，まとめる手法がある．[10]．この手法では，過去に通知されたアラートの中で対処されたアラートに焦点を当て，10 個の特徴量 (アラート名，アラートの属するカテゴリー，アラートの分類，ステータス，継続時間，非クリア継続時間，Node 全体の Warning 数，Node 全体の Critical 数，値，単位) を元にランダムフォレストを使用して分析をしている．この手法をもちいることで経験年数が高いエンジニアがどのようにチケットをまとめるのかを再現する．一方，この手法では，過去のアラート履歴を元に分析を行うためチケットのまとめかたが人に依存してしまうデメリットがある．

チケット間の内容の近似性をもとに，関連するチケットを自動的に統合する手法が提案されている [17]．この研究では，各チケットのタイトルや本文をテキストデータとし

て解析し，TF-IDF による特徴抽出やベクトル化された文書表現，さらに類似度スコアをもちいてチケット同士の関連度を算出している．その結果，内容的に重複している，あるいは類似した障害に関するチケットをグループ化し，対応作業の効率向上を実現している．一方で，障害がシステム内の Node や Pod，アプリケーションのように 1 つの障害が原因で別の障害が発生する場合，チケット内の記述内容の違いにより別の事象として扱われる．そのため，同一の障害に起因しているにもかかわらず，テキスト上の表現差によって自動統合の対象とならない問題がある．

管理者が登録したインシデントチケットの情報を活用し，見落とされた重大なアラートを自動的に検出する研究が提案されている [18]．この研究では，チケットに記載された内容の中でも「ドメイン語」と呼ばれる専門的な用語に注目し，それらの語を手がかりにチケットに関連する障害へ対応するアラートを削減している．これにより，既にチケット化されている既知の障害に由来する重複的なアラートの通知を抑制することが可能となる．一方で，過去にチケットとして登録されておらず，ドメイン語を抽出できない新規の障害に対しては，アラートを削減できない課題が残されており，さらなる改善の余地がある．アラートが削減できない場合，登録されるチケット件数も削減できない．

3. 提案

本提案の目的は，障害発生時に通知されたアラートをチケットとして登録する際に同一の障害で発生したアラートをチケットとしてまとめて登録することである．提案では，アラートが通知された際に，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係をチケットに記録する．例えば，Kubernetes 上の Node の Filesystem が 40GB のうち 36GB 使用され閾値である 90%を超えたアラートが通知され，チケットが登録された際，Node 上に配置されている Pod 名をそのチケットに記録する．その後，最初に登録されたチケット内に記録された Pod 名のアラートが通知された際は，最初に登録されたチケットに関連したチケットとして登録する．

提案方式

提案方式の概要を図 5に示す．

監視対象は ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node1，Node2，Node 上の Pod である．ESXi は ESXi をインストールした物理サーバを示す．監視システムは，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node1，Node2，Node 上の Pod からメトリクスを取得し，監視システムで設定されている，閾値を超えた場合にアラートを通知する．それぞれの Node 上には，FilAgent が配置されており，監視対象の

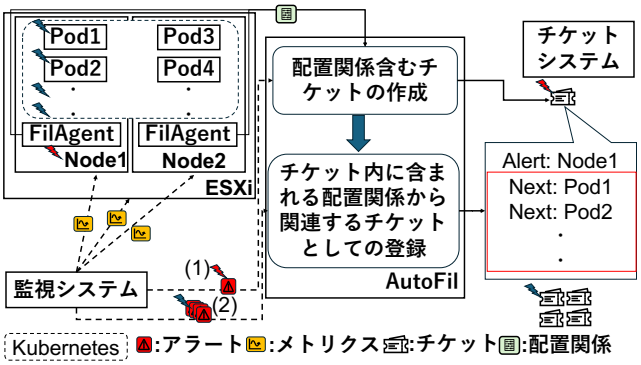

図 5: 提案方式の概要

ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod がどのように配置されているかを示す配置関係を取得する．例えば，図 5上の ESXi をインストールした物理サーバ上には Node1 と Node2 が配置されている．また，Node1 と Node2 にはそれぞれ Pod が配置されている．チケットシステムはアラートを元に登録されたチケットを保存する．AutoFil は 2 つの機能をもちいてチケットシステムにチケットの登録を行う．1 つ目は，配置関係を含むチケットの登録であり，アラートが通知された際に，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係をチケットに記録する．2 つ目はチケット内に含まれる配置関係から関連するチケットとしての登録であり，最初に登録されたチケット内に記録された対象からアラートが通知された際は，最初に登録されたチケットに関連したチケットとして登録する．

配置関係を含むチケットの登録

配置関係を含むチケットの登録では，アラートが通知された際に，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係をチケットに記録する．例えば，Node1 からアラートが通知された際に Node1 上に配置されている，Pod1 と Pod2 の Pod 名がチケット内に記録される．図中の Alert は，どの対象からアラートが通知されたのかを示す．Alert: Node1 は Node1 からアラートが通知されたことを示す．Next には，アラートが通知された対象に配置されている対象の名前が記録される．例えば，Node1 からアラートが通知された際に Pod1，Pod2 が Node1 上に配置されている対象である．

チケット内に含まれる配置関係から関連するチケットとしての登録

チケット内に含まれる配置関係から関連するチケットとしての登録では，1 回目のアラートが通知された後に通知されたアラートの対象と登録されたチケットの中に含まれる配置関係を比較する．その際に配置関係と通知されたアラート内に含まれる監視対象がマッチしている場合，関

連したチケットとして登録する．例えば，1 回目のアラートが Node1 から通知され，2 回目に Node1 上の Pod1 と Pod2 でアラートが通知された際に，1 回目に登録されたチケット内に記録された Pod1 と Pod2 の Pod 名が記録されているため，Pod1 と Pod2 のアラートを関連したチケットとして登録する．

ユースケース・シナリオ

ユースケースとして CDSL での監視作業中の 1 次調査を想定する．図 6にユースケースを示す．監視システムは，

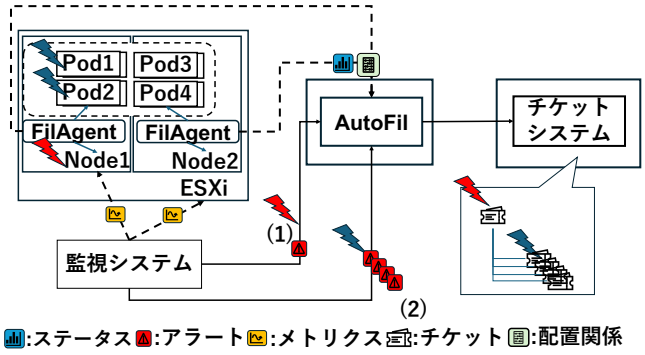

図 6: ユースケース

監視対象のメトリクスを取得する．監視対象は ESXi を配置した物理サーバと物理サーバ上に配置された Kubernetes クラスタ上の Node，Pod である．監視対象上には FilAgent が配置されており，配置関係を AutoFil に送信する．AutoFil は配置関係を元にチケットの登録を行う．チケットシステムには登録されたチケットが保存される．(1) は，Node1 からアラートが通知されたことを示す．(2) は，Node1 の障害が原因で Pod がダウンし，アラートが通知されたことを示す．監視作業中の 1 次調査者は最初に登録されたチケットをまとめる作業を行う．

4. 実装

実装したソフトウェアは Python3.1.0 で作成した．提案のシステムを実装するため FilAgent と AutoFil の 2 つのソフトウェアを作成した．図 7に実装したソフトウェアを示す．

監視対象は ESXi のインストールされた物理サーバ，Kubernetes クラスタ上の Node，Node 上に配置された Pod である．監視システムは対象の監視を行い，メトリクスを収集する．FilAgent は監視対象の Node 上に配置する．FilAgent は監視対象の物理サーバ，Node，Pod の配置関係を取得し，AutoFil に送信する．AutoFil は FilAgent からの配置関係の受け取り，配置関係の中からフィールドに登録する対象の決定，チケットの登録の 3 つの機能を持つ．チケットシステムにはチケットを保存される．

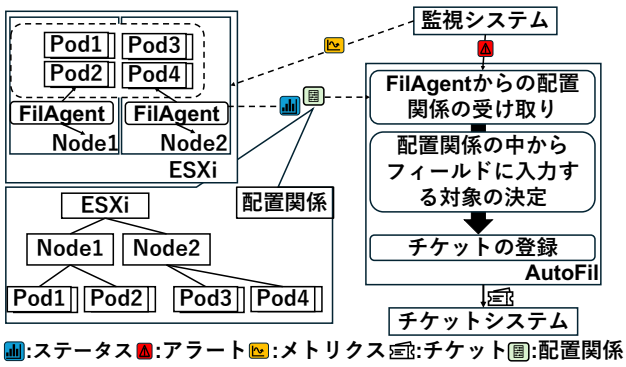

図 7: 実装したソフトウェア

AutoFil

AutoFil は，3 つの機能をもちいて通知されたアラートを元にチケットをまとめる．1 つ目が，FilAgent からの配置関係の受け取りであり，FilAgent が取得してきた監視対象の配置関係を取得する．2 つ目は，配置関係の中からフィールドに入力する対象の決定であり，最初に通知されたアラートに配置関係を登録する．例えば，1 回目のアラートで Node1 のアラートが通知された場合，Node1 がチケットとして登録され，Node1 上にいる Pod1，Pod2 の名前がフィールド内に登録される．そして，チケットのフィールド内に登録されている対象を示すアラートが通知された際に，そのチケットを関連するチケットとして登録する．

5. 評価実験

評価では，Kubernetes クラスタ上の Node のメモリ使用量が 8GB 全て使い切り Pod がダウンした事例，Node で使用される Filesystem の容量が 40GB のうち 36GB 使用され使用率が 90%を超えた事例，物理サーバに対して Blackbox exporter による疎通確認ができなくなった事例の 3 つを再現する．評価指標は，監視対象ごと，アラート名ごと，Job ごと，配置関係ごと，提案手法の 5 種類で登録されたチケットの件数を比較した．

また，実際に CDSL で運用されている環境内でアラートが通知された際に，通知件数をもちいた場合と，提案手法なしの場合でチケットの登録件数を比較した．比較を行った期間は，2025 年 12 月 4 日の 0 時からから 12 月 8 日の 23 時 59 分までである．

実験環境

図 8に実験環境を示す．Node1 と Node2 は監視サーバである．監視サーバは，Ubuntu24.04.2LTS を配置した 2 台の仮想マシンで構成されている．仮想マシンの性能は vCPU: 4[core],RAM:4[GB],SSD:40[GB] である．それぞれの Ubnutu 上には K3s をインストールしてあり，2 台でクラスタを構成している．監視対象は Monitoring-target1 から Monitoring-target7 の 7 つの Node である．Blackbox

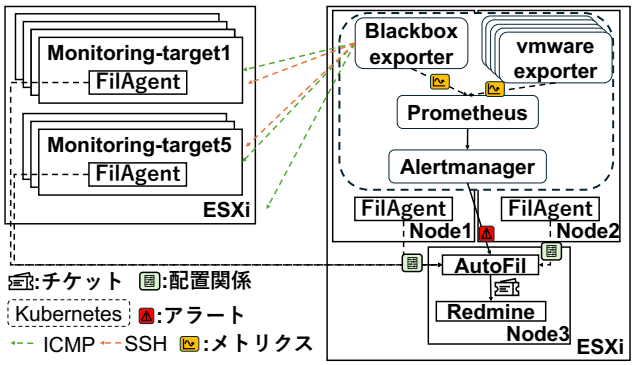

図 8: 実験環境

exporter は監視対象に対して ICMP と SSH の疎通確認を行う．vmware exporter は ESXi のメトリクスを監視する．Prometheus はメトリクスをもちいた異常判定を行う．Alertmanager は Promethus が異常判定したアラートを通知する，FilAgent は，Monitoring-target1 から Monitoring-target7 の Node と監視システムである Node1 と Node2 上に配置している．FilAgent は配置関係を AutoFil に通知する．AutoFil は Redmine 上にチケットを登録する提案ソフトウェアである．Redmine はチケットを保存する．

障害の再現

再現した障害は 3 つある．1 つ目が，2025 年 4 月 26 日 7 時 24 分に発生した，Kubernetes クラスタの Node 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ある．その障害を再現するために図 8の Node1 と Node2 を使用する．障害が起きた際に Node 上の vmware exporter の Pod が対象の ESXi のエンドポイントに接続できなくなったことでメモリを過剰に使用した．Node2 上に配置された vmware exporter の Pod は 8 台分である．エンドポイントに接続できなかった理由は CDSL 上で動いている ESXi の物理サーバを止めるシステムが働いたためである．CDSL では，電力の削減の観点から夜 8 時以降に物理サーバを止めるようになっている．その際に CDSL 上で運用されている開発環境の 6 台の物理サーバの電源を落とす．そのため，6 台の vmware exporter がエンドポイントに接続できなくなり，Node のメモリを枯渇させた．そのため，6 台の物理サーバの電源を事前に落とし，アラートを通知させる．

2 つ目が，11 月 7 日 (金)23 時 48 分に Node の Filesystem が 90%の閾値を超過した事例である．監視サーバである Node の内 Node1 の Filesystem の容量が 90%を超えた際に Node1 と Node1 上で動作する Pod のアラートが通知された．Filesystem の容量は 2 日かけて約 80%から約 93%まで増加していた．そのため Filesystem の容量を約 80%にした上で 2 日で約 93%を超えるようにファイルを作成し，容量を増加させた．

3 つ目が，2025 年 9 月 26 日 (金)9 時 51 分に通知された本番環境上に配置された物理サーバである Iris に対して疎通確認ができなくなった事例である．この事例では，物理サーバの疎通ができなくなったことで，物理サーバ上の Node である LiDARCluster を構成する 3 つの Node，Doktor を構成するクラスタは 4 つの Node も疎通ができなくなった．障害を再現するために，監視対象の ESXi 上に 7 台の物理サーバを配置し，ESXi の電源を落とした上で疎通ができない事例を再現する．

実験結果と分析

最初の評価では，3 つの障害事例を再現した際のチケットの件数を比較する．図 9に Node 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した際に登録されたチケット件数を示す．1 つ目の障害である 2025 年 4 月 26 日 7 時 24 分に発生した,Kubernetes クラスタの Node 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は，vmware exporter を示す 8 台の Pod のアラートと Node のアラートがチケットとして登録された．監視対象ごとの場

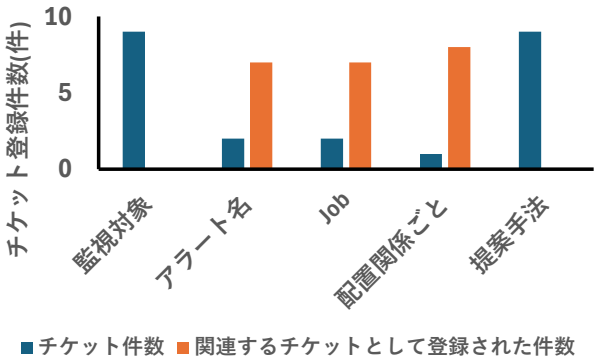

図 9: Node のメモリ使用量が枯渇したことで Pod がダウンした事例を再現した際に登録されたチケット件数

合，チケットが 9 件，関連するチケットとして登録された件数が 0 件となった．アラート名ごとに登録した場合，チケットが 2 件，関連するチケットとして登録された件数が 7 件となった．Job ごとに登録した場合，チケットが 2 件，関連するチケットとして登録された件数が 7 件となった．ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係ごとに登録した場合，チケットが 1 件，関連するチケットとして登録された件数が 8 件となった．提案手法では，チケットが 9 件，関連するチケットとして登録された件数が 0 件となった．

図 10に Node の Filesystem が 90%の閾値を超過した事例を再現した際の登録されたチケット件数を示す．2 つ目の事例である 11 月 7 日 (金)23 時 48 分に Node の Filesystem が 90%の閾値を超過した事例では，Node の Filesystem のアラートが 1 件，Pod のステータスがダウンしたアラー

トが 4 件，Diskpresser が起きたアラートが 1 件であり，6 件のチケットが登録された．監視対象ごとに登録した場

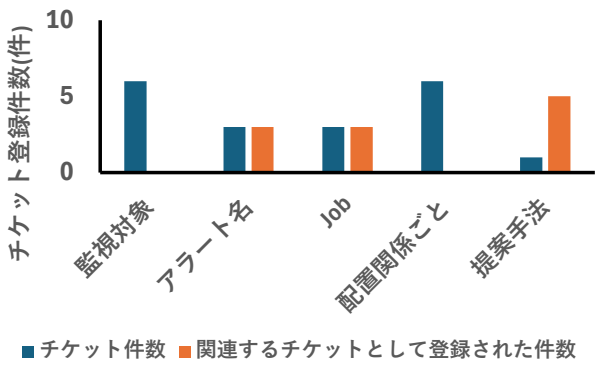

図 10: Node の Filesystem が 90%の閾値を超過した事例を再現した際の登録されたチケット件数

合，チケットが 6 件，関連するチケットとして登録された件数が 0 件となった．アラート名ごとに登録した場合，チケットが 3 件，関連するチケットとして登録された件数が 3 件となった．Job ごとに登録した場合，チケットが 3 件，関連するチケットとして登録された件数が 3 件となった．ESXi のインストールされた物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係ごとに登録した場合，チケットが 6 件，関連するチケットとして登録された件数が 0 件となった．提案手法では，チケットが 1 件，関連するチケットとして登録された件数が 5 件となった．

図 11に Iris に対して疎通確認ができなくなった事例を再現した際のチケット件数を示す．3 つ目の事例は，2025 年 9 月 26 日 (金)9 時 51 分に通知された本番環境上に配置された物理サーバである Iris に対して疎通確認ができなくなった事例である．アラートは監視対象の Doktor を構成するクラスタは 4 つの Node から 8 件，LiDARCluster を構成する 3 つの Node から 5 件，Iris の ESXi から 1 件通知された．そのため 14 件のチケットが登録された．監視

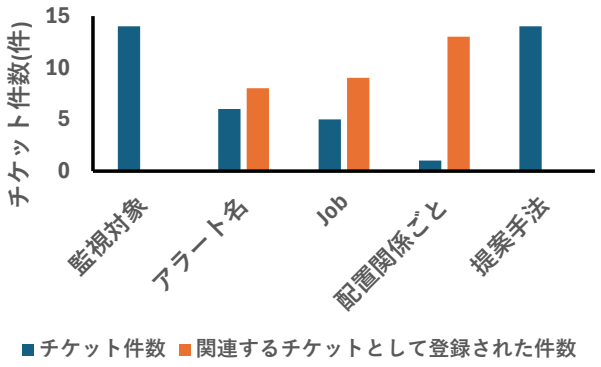

図 11: Iris に対して疎通確認ができなくなった事例を再現した際のチケット件数

対象ごとに登録した場合，チケットが 14 件，関連するチケットとして登録された件数が 0 件となった．アラート名ごとに登録した場合，チケットが 6 件，関連するチケットとして登録された件数が 8 件となった．Job ごとに登録した場合，チケットが 5 件，関連するチケットとして登録された件数が 9 件となった．ESXi をインストールした物理サーバ，Kubernetes 上の Node，Pod の配置関係ごとに登録した場合，チケットが 1 件，関連するチケットとして登録された件数が 13 件となった．提案手法では，チケットが 14 件，関連するチケットとして登録された件数が 0 件となった．

1 つ目の障害では，Kubernetes クラスタ上の Node 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は，Node が原因で Pod に影響を及ぼした点から Node を先に再起動しなければ，Pod の復旧や調査ができない．つまり，Node のアラートをチケットとし，それ以外の Pod のアラートを関連したチケットとしてまとめるべきである．監視対象ごとにアラートをまとめる場合と提案手法では，通知されているアラートの件数と，登録されているチケットの数が同じため，まとめられていない．一方で，アラート名ごとにまとめる場合と Job ごとにまとめる場合は，それぞれ，2 件ずつにチケットをまとめられている．配置関係ごとは，Node のステータスが Running 以外になったアラートをチケットとし，残りの Pod のステータスが Running 以外になったアラートを関連するチケットにまとめることができた．

2 つ目の障害では，Node の Filesystem が 90%の閾値を超過したことで Pod に影響を与えステータスが異常になった．そのため，Node の Filesystem が 90%を超過したアラートを先に解決しなければ，Pod がダウンする障害の復旧や調査ができない．そのため，Node の Filesystem が 90%を超過したアラートをチケットとし，それ以外の Pod のアラートを関連したチケットとしてまとめるべきである．監視対象と配置関係ごとは，通知されたアラートの件数と，登録されるチケットの件数が同じため，チケットをまとめられていない．アラート名と Job ごとにまとめた場合，チケットは 3 件ずつにまとまった．提案手法は，Node の Filesystem の容量が 90%を超えたアラートをチケットにし，残りの Pod のステータスが，Runing 以外になったアラートを関連するチケットにできていた．

3 つ目の障害では，本番環境上で動作する Iris に疎通確認ができなくなったことで，Iris 上で動作する 7 つの Node にも疎通確認ができなくなった．そのため，先に Iris 自体が疎通できないアラートを解決しなければ，復旧や調査ができない．そのため Iris 自体が疎通できないアラートをチケットとし，それ以外の Node に疎通ができなくなったアラートを関連したチケットにするべきである．監視対象ごとにアラートをまとめる場合と提案手法では，通知されて

いるアラートの件数と，登録されているチケットの数が同じため，まとめられていない．一方で，アラート名ごとにまとめる場合では，通知されたアラートがの種類が，6種類だったため，6 件のチケットが登録された．Job ごとの場合，チケット件数が 5 件となった．配置関係ごとは，ESXi をインストールした物理サーバである，Iris に対して ICMP の疎通確認ができなくなったことを示すアラートをチケットにし，それ以外を関連するチケットにまとめられている．

12 月 4 日から 12 月 8 日に登録されたチケット件数

図 12に 12 月 4 日から 12 月 8 日に登録されたチケット件数を示す．チケットは提案手法をもちいてチケットをまとめた場合と，提案手法をもちいない場合に登録されるチケットの件数を比較した．比較を行った期間は，2025 年 12 月 4 日の 0 時からから 12 月 8 日の 23 時 59 分までである．グラフの縦軸は，チケットの件数を示している．グラフの横軸は期間を示している．2025/12/4 は 2025 年 12 月 4 日 0:00 分から 12 月 4 日 23:59 分までを示す．2025/12/5 は 2025 年 12 月 5 日 0:00 分から 12 月 5 日 23:59 分までを示す．2025/12/6 は 2025 年 12 月 6 日 0:00 分から 12 月 6 日 23:59 分までを示す．2025/12/7 は 2025 年 12 月 7 日 0:00 分から 12 月 7 日 23:59 分までを示す．2025/12/8 は 2025 年 12 月 8 日 0:00 分から 12 月 8 日 23:59 分までを示す．2025 年 12 月 4 日のチケットの登録件数は，提案手法なしの場合，124 件となり，提案手法ありが，55 件となった．2025 年 12 月 5 日のチケットの登録件数は，提案手法なしと提案手法ありの場合で 51 件となった．2025 年 12 月 6 日のチケットの登録件数は，提案手法なしと提案手法ありの場合で 51 件となった．2025 年 12 月 7 日のチケットの登録件数は，提案手法なしと提案手法ありの場合で 48 件となった．2025 年 12 月 8 日のチケットの登録件数は，提案手法なしと提案手法ありの場合で 46 件となった．

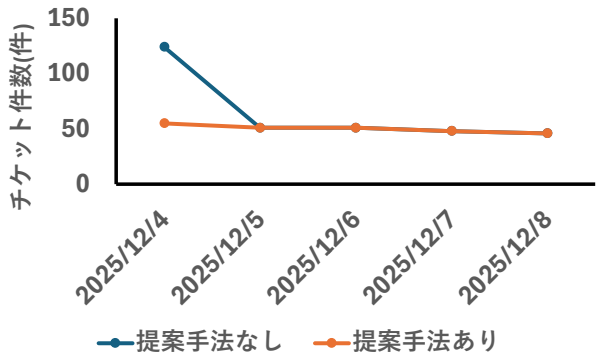

図 12: 12 月 4 日から 12 月 8 日に登録されたチケット件数

次に 2025 年 12 月 4 日の 0 時からから 12 月 8 日の 23 時 59 分までで登録されたチケットの件数を 1 日ごとに分けて登録されたチケットの比較を行う．表 1に 2025 年 12 月 4 日 0:00 分から 12 月 4 日 23:59 分までに登録されたチケットの件数を示す．Monitoring Cluster Abnormal pod status で提案手法なしの場合，68 件となり，提案手法ありの場合，0 件となった．Doktor Front Page HTTP Status Code Err paper で提案手法なしと提案手法なしの場合，21 件となった．Doktor Front Page HTTP Status Code Err author で提案手法なしと提案手法なしの場合，17 件となった．Doktor Front Page HTTP Status Code Err で提案手法なしと提案手法なしの場合，4 件となった．output Lidar data check port 8001，output Lidar data check port 8002，Clematis Node Memory Usage High 90%，DataStore DiskUsageHigh-Usage 80%，DataStore DiskUsageHigh-Usage 90%，Host CPU UsageHigh-Usage100%はそれぞれ，2 件ずつとなった．Monitoring Cluster Node Filesystem Usage High 90%は，提案手法なしと提案手法ありで場合，それぞれ，1 件となった．Monitoring Cluster Node DiskPressure は提案手法なしの場合，1 件となり，提案手法ありの場合，0 件となった．提案手法で Monitoring Cluster Abnormal pod status と，Monitoring Cluster Node DiskPressure はそれぞれ 0 件となっている．理由は，Monitoring Cluster Node Filesystem Usage High

表 1: 2025 年 12 月 4 日 0:00 分から 12 月 4 日 23:59 分までに登録されたチケットの件数

チケット名	提案手法なし	提案手法あり
Monitoring Cluster Abnormal pod status	68	0
Doktor Front Page HTTP Status Code Err paper	21	21
Doktor Front Page HTTP Status Code Err author	17	17
Doktor Front Page HTTP Status Code Err	4	4
output Lidar data check port 8001	2	2
output Lidar data check port 8002	2	2
Clematis Node Memory Usage High 90%	2	2
DataStore DiskUsageHigh-Usage 80%	2	2
DataStore DiskUsageHigh-Usage 90%	2	2
Host CPU UsageHigh-Usage100%	2	2
Monitoring Cluster Node Filesystem Usage High 90%	1	1
Monitoring Cluster Node DiskPressure	1	0

90%のチケット 1 件が，チケットとなり，関連したチケットとしてまとめられたためである．そのため，提案手法ありの方が登録されたチケットの件数が少なかった．

表 2に 2025 年 12 月 5 日 0:00 分から 12 月 5 日 23:59 分までに登録されたチケットの件数を示す．Doktor Front Page

表 2: 2025 年 12 月 5 日 0:00 分から 12 月 5 日 23:59 分までに登録されたチケットの件数

チケット名	提案手法なし	提案手法あり
Doktor Front Page HTTP Status Code Err author	18	18
Doktor Front Page HTTP Status Code Err paper	18	18
Doktor Front Page HTTP Status Code Err	3	3
output Lidar data check port 8001	2	2
output Lidar data check port 8002	2	2
Clematis Node Memory Usage High 90%	2	2
DataStore DiskUsageHigh-Usage 80%	2	2
DataStore DiskUsageHigh-Usage 90%	2	2
Host CPU UsageHigh-Usage100%	2	2

HTTP Status Code Err author, Doktor Front Page HTTP Status Code Err paper は，提案手法なしと提案手法ありで場合，それぞれ，18 件となった．Doktor Front Page HTTP Status Code Err は，提案手法なしと提案手法ありで場合，それぞれ，3 件となった．output Lidar data check port 8001，output Lidar data check port 8002，Clematis Node Memory Usage High 90%，DataStore DiskUsageHigh-Usage 80%，DataStore DiskUsageHigh-Usage 90%，Host CPU UsageHigh-Usage100%では，提案手法なしと提案手法ありで場合，それぞれ，2 件となった．提案手法ありと提案手法なしでは，チケットの登録された件数が変わらなかった．

表 3に 2025 年 12 月 6 日 0:00 分から 12 月 6 日 23:59 分までに登録されたチケットの件数を示す．

Doktor Front Page HTTP Status Code Err author は，提案手法なしと提案手法ありで場合，それぞれ，21 件となった．Doktor Front Page HTTP Status Code Err paper は，提案手法なしと提案手法ありで場合，それぞれ，14 件となった．Doktor Front Page HTTP Status Code Err，Host CPU Usage High Usage 100%は，提案手法なしと提案手法ありで場合，それぞれ，3 件となった．output Lidar data check port 8001，output Lidar data check port 8002，Clematis Node Memory Usage

表 3: 2025 年 12 月 6 日 0:00 分から 12 月 6 日 23:59 分までに登録されたチケットの件数

チケット名	提案手法なし	提案手法あり
Doktor Front Page HTTP Status Code Err author	21	21
Doktor Front Page HTTP Status Code Err paper	14	14
Doktor Front Page HTTP Status Code Err	3	3
Host CPU Usage High Usage 100%	3	3
output Lidar data check port 8001	2	2
output Lidar data check port 8002	2	2
Clematis Node Memory Usage High 90%	2	2
DataStore DiskUsageHigh-Usage 80%	2	2
DataStore DiskUsageHigh-Usage 90%	2	2

High 90%，DataStore DiskUsageHigh-Usage 80%，DataStore DiskUsageHigh-Usage 90%では，提案手法なしと提案手法ありで場合，それぞれ，2 件となった．提案手法ありと提案手法なしでは，チケットの登録された件数が変わらなかった．

表 4に 2025 年 12 月 7 日 0:00 分から 12 月 7 日 23:59 分までに登録されたチケットの件数を示す．

表 4: 2025 年 12 月 7 日 0:00 分から 12 月 7 日 23:59 分までに登録されたチケットの件数

チケット名	提案手法なし	提案手法あり
Doktor Front Page HTTP Status Code Err paper	17	17
Doktor Front Page HTTP Status Code Err author	15	15
Doktor Front Page HTTP Status Code Err	4	4
output Lidar data check port 8001	2	2
output Lidar data check port 8002	2	2
Clematis Node Memory Usage High 90%	2	2
DataStore DiskUsageHigh-Usage 80%	2	2
DataStore DiskUsageHigh-Usage 90%	2	2
Host CPU UsageHigh-Usage100%	2	2

Doktor Front Page HTTP Status Code Err paper は，提案手法なしと提案手法ありで場合，それぞれ，17 件となった．Doktor Front Page HTTP Status Code Err author は，提案手法なしと提案手法ありで場合，それぞれ，15 件となった．Doktor Front Page HTTP Status Code Err は，提案手法なしと提案手法ありで場合，それぞれ，4 件となった．output Lidar data check port 8001，output Lidar data check port 8002，Clematis Node Memory Usage High 90%，DataStore DiskUsageHigh-Usage 80%，DataStore DiskUsageHigh-Usage 90%，Host CPU UsageHigh-Usage100%では，提案手法なしと提案手法ありで場合，それぞれ，2 件となった．提案手法ありと提案手法なしでは，チケットの登録された件数が変わらなかった．

表 5に 2025 年 12 月 8 日 0:00 分から 12 月 8 日 23:59 分までに登録されたチケットの件数を示す．Doktor Front Page HTTP Status Code Err paper は，提案手法なしと提案手法ありで場合，それぞれ，9 件となった．Doktor Front Page HTTP Status Code Err author は，提案手法なしと提案手法ありで場合，それぞれ，6 件となった．output Lidar data check port 8002，output Lidar data check port 8001 は，提案手法なしと提案手法ありで場合，それぞれ，5 件となった．Host CPU Usage High Usage 100%，DataStore DiskUsageHigh Usage 90%，DataStore DiskUsageHigh Usage 80%，Clematis Node Memory Usage High 90%は，提案手法なしと提案手法ありで場合，それぞれ，4 件となった．Clematis Node Memory Average Usage High 80%は，提案手法なしと提案手法ありで場合，それぞれ，3 件となった．Monitoring Cluster PersistentVolumeFullSoon，Doktor Front Page HTTP Status Code Err は，提案手法なしと提案手法ありで場合，それぞれ，1 件となった．提案手法ありと提案手法なしでは，チケットの登録された件数が変わらなかった．

表 5: 2025 年 12 月 8 日 0:00 分から 12 月 8 日 23:59 分までに登録されたチケットの件数

チケット名	提案手法なし	提案手法あり
Doktor Front Page HTTP Status Code Err paper	9	9
Doktor Front Page HTTP Status Code Err author	6	6
output Lidar data check port 8002	5	5
output Lidar data check port 8001	5	5
Host CPU Usage High Usage 100%	4	4
DataStore DiskUsageHigh Usage 90%	4	4
DataStore DiskUsageHigh Usage 80%	4	4
Clematis Node Memory Usage High 90%	4	4
Clematis Node Memory Average Usage High 80%	3	3
Monitoring Cluster PersistentVolumeFullSoon	1	1
Doktor Front Page HTTP Status Code Err	1	1

6. 議論

今回の実験から，提案手法では，時間差で発生する Filesystem の障害に対しては有効である一方，Kubernetes クラスタ上の Node のメモリ枯渇によって Pod がダウンした事例や，Iris に対して疎通確認ができなくなった事例には有効ではないことが分かった．その理由は，AutoFil が最初に通知されるアラートに対しては配置関係の登録のみを行い，アラート同士をまとめる機能を持たないためである．これら 2 つの事例に共通する特徴は，ESXi や Node のダウンによって，それらに配置する Node や Pod が連鎖的に影響を受けた点にある．これらの障害に対応するためには，過去のアラート通知結果と配置関係に因果関係を持たせる．具体的には，ESXi をインストールした物理サーバである Iris が疎通不能となった事例では，Iris の疎通確認ができなくなったことで Node に対して疎通確認ができなくなった．また，Node のメモリ使用量が枯渇した事例では，Node の停止が原因で Pod がダウンしている．これらの点から，Node で発生した障害は ESXi をインストールした物理サーバには影響を与えない関係性が読み取れる．このような関係を事前に AutoFil に認識させ，同様のアラートが発生した際にどの対象から通知されたアラートをチケットとし，どのチケットを関連したチケットとして扱うべきかを明確化になる．

障害が発生した際に対象のアプリケーションがダウンしたことでエンドポイントに疎通ができなくなる場合がある．その場合，配置関係やアプリケーションごとの接続関係が取れたとしてもチケットをまとめることができない．対象のアプリケーションがダウンする場合，比較的短い時間にアラートが集中することが起こる．そのため特定の時間アラートをまとめてチケットとして登録すれば，チケット件数の削減ができる．

7. おわりに

Cloud and Distributed Systems Laboratory では，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node や Pod を Prometheus をもちいて監視している．監視ルールで設定した監視対象のメトリクスの値が閾値を超えた際，Alertmanager がアラートを通知し，Redmine

にチケットとして登録する．課題は，通知されたアラートを Redmine に登録する際に，同一の障害が原因で通知されたアラートをまとめてチケットとして登録しなければ，チケット件数が増加することである．提案では，アラートが通知された際に，ESXi をインストールした物理サーバ，Kubernetes クラスタ上の Node，Pod の配置関係をチケットに記録する．例えば，Kubernetes クラスタ上の Node の Filesystem が 40GB のうち 36GB 使用され閾値である 90%を超えたアラートが通知され，チケットが登録された際，Node 上に配置されている Pod 名をそのチケットに記録する．その後，最初に登録されたチケット内に記録された Pod 名のアラートが通知された際は，最初に登録されたチケットに関連したチケットとして登録する．評価では，Kubernetes クラスタ上の Node のメモリ使用量が 8GB 全て使い切り Pod がダウンした事例，Node で使用される Filesystem の容量が 40GB のうち 36GB 使用され使用率が 90%を超えた事例，物理サーバに対して Blackbox exporter による疎通確認ができなくなった事例の 3 つを再現する．評価指標は，監視対象ごと，アラート名ごと，Job ごと，配置関係ごと，提案手法の 5 種類で登録されたチケットの件数を比較した．評価の結果，Kubernetes クラスタ上の Node のメモリ使用量が 8GB 全て使い切り Pod がダウンした事例では，提案手法は，チケット件数が 9 件となりチケットをまとめて登録することができなかった．Node で使用される Filesystem の容量が 40GB のうち 36GB 使用され使用率が 90%を超えた事例では，提案手法は，チケット件数が 1 件となり登録されるチケットをまとめることができた．物理サーバに対して Blackbox exporter による疎通確認ができなくなった事例では，提案手法は，チケット件数が 14 件となりまとめることができなかった．また，2025 年 12 月 4 日から 12 月 8 日までで登録されたチケットの件数を提案手法ありとなしで比較した．2025 年 12 月 4 日で登録されたチケットは，提案手法なしで 124 件，提案手法ありで 55 件となりチケットをまとめることができた．一方で，2025 年 12 月 5 日から 12 月 8 日は登録されたチケット件数が，提案手法ありとなしで変わらなかった．

提案手法ありで，チケットの件数をまとめられたのは，12 月 4 日のみだった．12 月 4 日のチケットがまとめられた要因は，通知されたアラートの種類にある．Monitoring Cluster Node Filesystem Usage High 90%はが通知された時間は，2025 年 12 月 4 日 01:56 分であった．その後，2025 年 12 月 4 日 07:21 分に monitoring-master-ml 上の Pod が DiskPressure を起こしたことで，ステータスが Running 以外になった．そのため，monitoring-master-ml に Node 上の Pod の名前が追加され，その Pod からアラートが出たことでまとめることができた．

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] Voutsas, F., Violos, J. and Leivadeas, A.: Filtering Alerts on Cloud Monitoring Systems, *2023 IEEE International Conference on Joint Cloud Computing (JCC)*, pp. 34–37 (online), DOI: 10.1109/JCC59055.2023.00010 (2023).

[11] 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, *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 79–90 (online), DOI: 10.1109/ASE56229.2023.00177 (2023).

[12] Chen, J., Chen, X., Shi, J., Wang, P. and Wang, W.: VOCE: A Virtual On-Call Engineer for Automated Alert Incident Analysis Using a Large Language Model, *Fundamental Approaches to Software Engineering* (Boronat, A. and Fraser, G., eds.), Cham, Springer Nature Switzerland, pp. 65–88 (2025).

[13] Voutsas, F., Violos, J. and Leivadeas, A.: Mitigating Alert Fatigue in Cloud Monitoring Systems: A Machine Learning Perspective, *Computer Networks*, Vol. 250, p. 110543 (online), DOI: https://doi.org/10.1016/j.comnet.2024.110543 (2024).

[14] 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).

[15] 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).

[16] Yang, B., run Wang, H., guang Li, H. and dong He, Y.: A novel detection of correlated alarms with delays based on improved block matching similarities, *ISA Transactions*, Vol. 98, pp. 393–402 (online), DOI: https://doi.org/10.1016/j.isatra.2019.07.011 (2020).

[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).

[18] 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).

