構成管理データベースを使用した重みづけとアラートの通知件数の削減

平尾 真斗[1]　串田 高幸[1]

概要：障害発生時に原因の切り分け作業が遅くなると企業は金銭的リスクを負う．オンコールエンジニアは障害発生時に一時的な原因箇所の切り分けとエスカレーションを行う．課題は障害の発生箇所が 1 箇所であることに対して障害の発生箇所ではない監視対象からアラートが通知されることである．障害の発生箇所以外からアラートが通知されることで，エスカレーションにかかる時間が増加する．本稿ではシステム同士の構成を管理する構成管理データベースから依存関係を取得し重みを算出する．また通知されたアラートの中から重みの値が最大値の対象とそれ以外のアラートでまとめることでアラート通知件数を削減する．評価実験ではシナリオを元に対象に対して障害を発生させる．評価指標は，通知されたアラート件数である．その後，通知されたアラートから重要なアラートと関連するアラートの通知件数，エスカレーションにかかる作業量の 2 つを評価する．評価実験のシナリオは物理機器に接続する LAN ケーブルの接続不良と NFS サーバを配置する OS のファイルシステムの容量が 80%を超えた際に NFS サーバ側とクライアント側の両方でアラートが通知される 2 つである．通知されたアラートの件数で比較対象の AlertManager のみをもちいた場合のアラートの通知件数は LAN ケーブルの接続不良の障害シナリオでは 39 件となり，NFS サーバを配置している OS のファイルシステムの容量が 80%を超えたシナリオでは 4 件となった．AlertManager と提案ソフトウェアを組み合わせた場合のアラート通知件数は LAN ケーブルの接続不良の障害シナリオでは 2 件，NFS サーバを配置している OS のファイルシステムの容量が 80%を超えたシナリオでは 2 件となった．結果的にアラートの通知件数を約 91%削減できた．また通知されたアラートをまとめた際の LAN ケーブルの接続不良の障害シナリオでは，関連して通知されたアラートが AlertManager のみの場合が 33 件となり，AlertManager と提案ソフトウェアをもちいた場合が 1 件であった．障害発生箇所から通知された重要なアラートは AlertManager と提案ソフトウェアをもちいた場合が 1 件で AlertManager のみの場合が 6 件となった．エスカレーションにかかる作業量は AlertManager と提案ソフトウェアの場合が 0.5，AlertManager のみの場合が 0.84 となった．

1. はじめに

背景

インターネット上で公開されている企業のシステムは，数千のコンポーネントを用いて構築され，ユーザをサポートする [1]．それに伴い安定した動作を提供する必要がある [2]．障害が発生した場合，迅速に復旧しなければ企業は金銭的なリスクを負うことになる [3]．例えば，2017 年に世界最大の EC サイトである Amazon で発生した障害では，1 億 5,500 万ドル以上の損失が発生したと推定されている [4].

障害を迅速に解決することで, ビジネスへの影響が少なくなり, ユーザの信頼も高まる．システムの可用性を保証するために企業は監視を行う [5, 6]．監視システムは障害発生時にオンコールエンジニアに対しアラートを通知する [7]．オンコールエンジニアは監視システムが通知したアラートを受け取り，障害箇所の 1 次的な診断と手順書を元にした復旧作業を行う [8]．これらの対処ができない場合，2 次対応の担当者に対してエスカレーションを行う．

システムのアーキテクチャや監視ルールの設定によって，障害の発生箇所以外の箇所でアラートが通知される場合がある [9]．図 1に障害の発生箇所以外でアラートが通知される例を示す.

東京工科大学の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(以後:CDSL と呼ぶ) では，10 台の物理機器が配置されている．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おり，Virtual Machine(以後:VM と呼ぶ) が配置されている [10]．7 台の物理機器は学生が自由に実験や作業を行う目的で使用され，2 台は

[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

DNS と DHCP サーバを運用するために配置されている．残りの 1 台は外部にサイトやアプリケーションを配置し公開する目的で使用される．NAS は ESXi 上に配置された VM のストレージを提供する．NAS は学生が自由に VM を作成できる 7 台の物理機器に接続されている．SW1 はスイッチングハブである．監視システムには 3 つのソフトウェアが配置されている．Blackbox Exporter は外形監視を行うエージェントであり，物理機器上に配置している ESXi や VM 上に配置されている OS やアプリケーションを監視している．監視項目は ICMP と SSH のプロトコルを使用した VM 上に配置された OS への疎通確認，VM 上の OS に配置されたアプリケーションに対する HTTP のプロトコルをもちいた疎通確認，それぞれの疎通確認にかかった時間の 4 つである．物理機器上の NIC は VM 上の OS やアプリケーションに通信を行う際に経由する．図中の Mint は学生が自由に VM を作成できる物理マシンの一つである．2024 年 3 月 31 日に Mint の NIC と SW1 を繋ぐ LAN ケーブルが接続不良になった．LAN ケーブルでこの箇所の通信ができなくなると Mint の ESXi 上に配置されている VM1，VM2，VM3，VM4 の OS に対しても通信ができなくなる．その際に監視システムは物理機器である Mint と VM1 から VM4 の OS の 5 つの監視対象で疎通確認ができないことを示す 39 件のアラートを通知した．オンコールエンジニアは監視システムが通知したアラートを受け取る．障害箇所の 1 次的な診断と手順書を元にした復旧作業を行い，対処ができない場合，2 次対応の担当者に対してエスカレーションを行う [8]．図 1 の事例では 1 次対応として ICMP のプロトコルを使用した OS への疎通確認と Mint に対する HTTP リクエストを送る 1 次対応が行われた．その後，2 次対応の担当者にエスカレーションされた．

また，システム同士の依存関係が原因で障害発生箇所以外でアラートが通知される例もある．図 2に NFS サーバとクライアントの依存関係が原因で障害発生箇所以外でアラートが通知された例を示す．CDSL では 2024 年の 2 月から 6 月の間に論文要約サイト (以後:CPR と呼ぶ) を公開

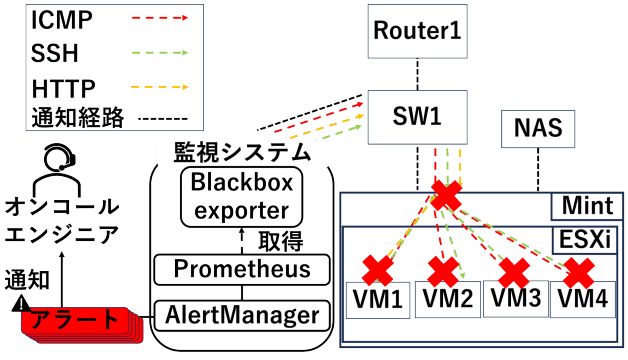

図 1: 障害の発生箇所以外でアラートが通知される例

していた．このサイトでは，CDSL の学生が論文に対する要約を行い外部のユーザが論文の要約を閲覧する．CRP には VM1 から VM4 の 4 つの VM が配置されており，VM1 の OS 上に NFS サーバが配置されており，VM2 から VM4 の OS 上には NFS クライアントが配置されている．

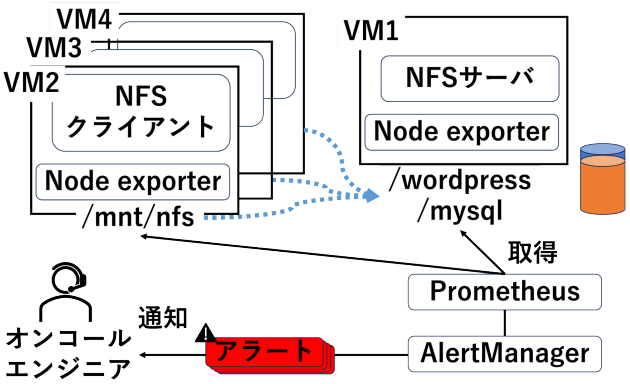

図 2: NFS サーバとクライアントの依存関係が原因で障害発生箇所以外でアラートが通知された例

VM1 上の OS には NFS サーバが配置されており，VM2 から VM4 上の OS に配置されたクライアントとファイルシステムを共有する．Node Exporter は VM1 から VM4 の OS 上に配置されており，ファイルシステムの容量を監視する．Prometheus は Node Exporter が監視した内容を取得し，ファイルシステムの容量が 80%を超えた際にアラートとして通知する．AlertManager は Prometheus の通知したアラートをオンコールエンジニアに通知する．2024 年の 6 月 3 日に NFS サーバのファイルシステムの容量が 80%を超えることを示すアラートが通知された．その際にファイルシステムを共有している VM1 だけでなく VM2 から VM4 までの対象でも同様のアラートが通知された．原因は NFS サーバとクライアントがファイルシステムを共有しており，クライアントのファイルシステムも同様にファイルシステムの使用率が 80%を超えていたことである．そのため VM1 のみでなく VM2 から VM4 でもアラートが通知された．

課題

課題は障害の発生箇所が 1 箇所であることに対して障害の発生箇所ではない監視対象からアラートが通知されることである．図 1の事例では，物理機器の NIC と SW1 を繋ぐ LAN ケーブルの接続不良が原因で ESXi 上に配置されている VM 上の OS にも通信ができなくなった．結果として 39 件のアラートが通知された．オンコールエンジニアはアラートの通知を受け取り障害箇所の 1 次的な診断と手順書を元にした復旧作業を行う．その際に，障害発生箇所が複数になることが原因でエスカレーションにかかる時間

が増加する [11–13].

またアラートの通知件数は 1 件にまとめる設定を行うことで 1 件にまとめることもできる．しかし，1 件にまとめてしまうと 1 件の中から障害の発生箇所を示す重要なアラートを示す箇所を探さなくてはならない．そのためエスカレーションに時間がかかってしまう．このような場合，障害の発生箇所とそれ以外の関連したアラートをそれぞれまとめて通知することで 1 次対応のエンジニアの対処がしやすくなる．

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 [14]．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．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 [15]．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注目するべき単語があるチケットを取得し，そこから重大なアラートを取得する．一方でチケットがあることが前提のため，チケットにない障害のアラート通知件数を削減することはできない．

アラートの重要度をランク付けするピアレビューメカニズムを用いて管理者に通知されるアラートの件数を削減する研究がある [16]．この論文では AlertRank というソフトウェアと機械学習をもちいたアラートの分類により，通知されたアラートの重要性を示している．CPU 使用率とネットワークパケット数を監視項目を入れているが，実際の監視ではネットワークやディスク容量，メモリの項目も入れるため，項目が不十分である．

クラウドシステムのアラートの削減に着目した研究がある [17]．この論文では，実際のアラートを含むデータセットにもとづくデータ駆動型のアプローチでアラートの数を減らしている．一方で評価実験でどのような障害や異常な動作を想定するかを明確に示していない．

アラートの検出とサマリする方式を用いてアラートストーム時の通知件数を削減する研究がある [18]．論文内では中国の銀行で使用された監視アラートやチケットをもとに手法の評価を行っているが，どのような障害のチケットが管理者に通知されたか一例をあげておらず，どのような障害に対して有効かどうか判断できない．

3. 提案方式

本稿ではシステム同士の構成を管理する構成管理データベースから依存関係を取得し重みを算出する．また通知されたアラートの中から重みが大きい対象のアラートをまとめ，それ以外をまとめることでアラートの通知件数を削減する．構成管理データベースには，ハイパーバイザの IP アドレス，VM の IP アドレス，VM がどこの対象に対して接続関係を持つかの情報が入力される．提案の概要を図 3に示す．

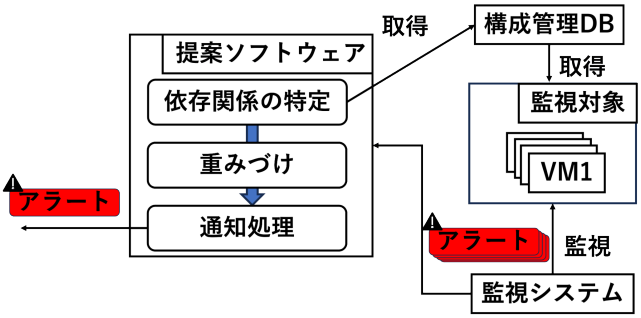

図 3: 提案の概要

監視システムは対象に対して監視を行う．事前に設定されていた閾値を超えた際にアラートを通知する．構成管理データベースは監視対象の構成情報を管理している．提案ソフトウェアには 3 つの機能がある．1 つ目が，依存関係の取得である．依存関係は構成管理データベース内に記述されている．2 つ目に依存関係をもちいた重みづけである．依存関係の取得後重みをそれぞれ算出する．3 つ目は通知処理である．監視対象ごとの依存関係から取得した重みからアラートの通知をまとめる対象とそれ以外の対象を決定する．

構成管理データベース

構成管理データベースは監視対象ごとの構成や接続先を記録したデータベースである．表 1に構成管理データベースの概要を示す．

構成管理データベースは HV_IP，IP，Connect IP Port の 3 つの要素から構成される．HV_IP はハイパーバイザの IP アドレスを示している．Connect IP Port は接続しに行く対象を示す．例えば NFS のサーバとクライアントをもちいたシステムで 192.168.100.140 が NFS サーバを OS 上に配置した VM でそれ以外の対象が NFS クライアントを OS 上に配置した VM であるとする．その際に 192.168.100.141 は 192.168.100.140:2049 と接続している．

表 1: 構成管理データベース

HV_IP	IP	Connect IP Port
192.168.100.21	192.168.100.140	192.168.100.141:789 192.168.100.142:910 192.168.100.143:820
192.168.100.21	192.168.100.141	192.168.100.140:2049
192.168.100.21	192.168.100.142	192.168.100.140:2049
192.168.100.21	192.168.100.143	192.168.100.140:2049

依存関係の取得

依存関係の取得は構成管理データベースの内容から行う．図 4に依存関係の取得方法を示す．

構成管理データベースには構成要素がそれぞれ書かれている．依存関係の取得では，HV_IP から IP に向けて矢印を向ける．このようにそれぞれの要素の中から右に向けて矢印を向けていき依存関係を取得する．

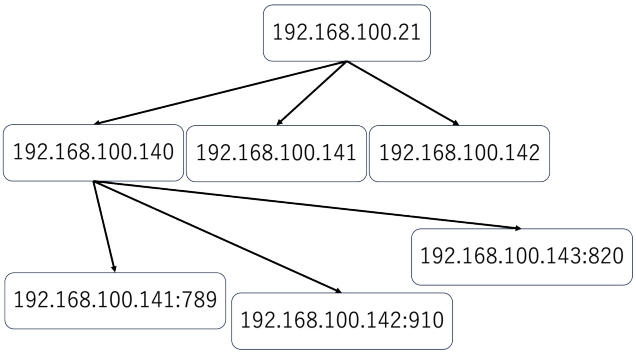

図 4: 依存関係の取得

重みづけ

次に依存関係から重みを算出する．重みは依存関係の取得で得られた依存関係から算出される．式 (1) の計算式で重みを算出する．

$$W = \frac{L}{A} \tag{1}$$

W は監視対象ごとの重みである．L は依存している対象の数を示している．A は総要素数である．図 5に依存関係の計算を行った結果を示す．例えば 192.168.100.21 は 6 つの対象から依存されている．そのため依存された数が L となる．また，総要素数は 7 となる．その下の 192.168.100.140 は総要素数が 6 で依存された数が，3 となる．

通知処理

通知処理では，重みづけで取得した監視対象ごとの重みの値の中から最大の値を取る箇所を算出する．その際に (2) の式を用いて最大値を取得する．

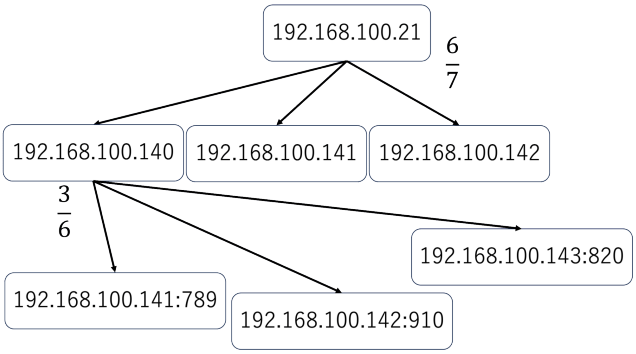

図 5: 重み算出の具体例

$$M = \max(W_1, W_2, \ldots, W_n) \tag{2}$$

W は重みづけで取得した監視対象ごとの値である．M は取得した W のそれぞれの値の中で最も値が大きい値である．重みが最も大きい値は，アラートが通知された際に使用される．図 6に通知処理の具体例を示す．障害発生時に複数のアラートが発生する場合がある．例えば Mint の LAN ケーブルの接続不良の事例では Mint の物理機器とその中に配置されている VM でもアラートが通知された．その際に提案のソフトウェアはアラートの通知内容の中で通知箇所の選出の際に取得した最大の値をもつ監視対象のアラートとそれ以外のアラートでまとめる．ハイパーバイザや通信の集中する箇所の影響で障害が発生する場合，重みの強い箇所以外の箇所もアラートとして通知される．そのため，重みの最も大きい箇所とそれ以外で分けることで障害の発生箇所とそれ以外の箇所でアラートをまとめることができる．

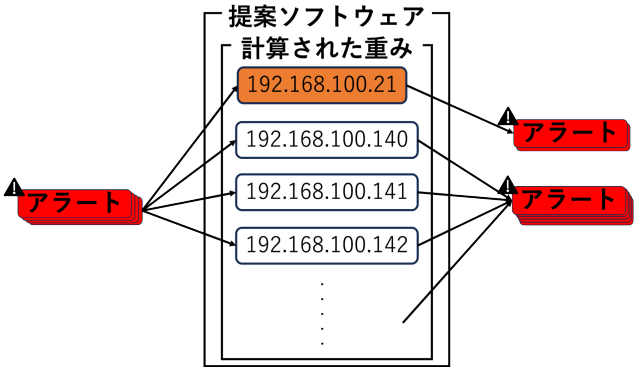

図 6: 通知処理の具体例

ユースケース・シナリオ

本ユースケースシナリオとして CDSL の環境内で障害発生時に通知されたアラートにおけるオンコールエンジニアの 1 次対応を想定する．図 7にユースケースシナリオを示す．

監視システムは物理機器と VM 上の OS に対して ICMP と SSH のプロトコルを使用した疎通確認，OS 上に配置さ

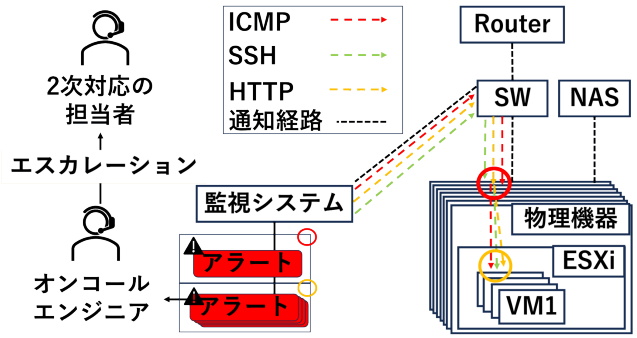

図 7: ユースケースシナリオ

れたアプリケーションに対する HTTP リクエストを行う．SW はスイッチングハブであり，Router や物理機器の通信を中継する．NAS は物理機器内のハイパーバイザ上に配置されている VM のストレージである．オンコールエンジニアは監視システムが通知されたアラートを受け取り，障害の発生箇所の切り分けを行う．その後，手順書を用いて対処を行う．手順書に沿った対処ができない場合，オンコールエンジニアは 2 次対応の担当者にエスカレーションを行う [19]．原因箇所を切り分ける際に障害の発生箇所から通知されるアラートと関連して通知されるアラートの中から発生箇所を示すアラートを探す．その際に提案ソフトウェアを用いることで発生箇所の切り分けをしやすいアラートの通知にすることができる．

4. 実装

本提案手法を実装したソフトウェアは Python3.10.12 で作成した．ライブラリは re，xmlrpc.client，xmlrpc.server，mysql-connector-python，subprocess である．実装したソフトウェアは 3 つある．1 つ目は ss コマンドの接続対象と所属するハイパーバイザの IP アドレスを提供する．CMDB-exporter である．2 つ目は構成管理データベースを構築する CMDB-Creater である．3 つ目は AlertManager が通知したアラートをまとめる設定を行う Autofiltering である．

CMDB-exporter

CMDB-exporter は構成管理データベースを構築する際に必要となる所属するハイパーバイザの IP アドレスとどの対象に対してアプリケーションが接続するのかを CMDB-Creater に提供する．ハイパーバイザの IP アドレスを取得には vmtoolsd のコマンドを使用した．また，接続しにいく IP アドレスの取得にはネットワークのソケットの情報を出力する ss コマンドをもちいた．ss コマンドにはどの対象の IP アドレスと Port に対して接続するかを示す peer address port の箇所がある．その箇所を ss コマンドから取得し，CMDB-creater に送信する．

CMDB-creater

CMDB-creater は CMDB-exporter が取得してきたハイパーバイザの IP アドレスと ss コマンドで取得した接続先の IP アドレスと Port をもとにデータベースに書き込む．データベースは MySQL を使用しており Python の mysql-connector-python モジュールを使用し書き込む．

AutoFiltering

AutoFiltering は監視システムが通知を行ったアラートをまとめる処理を行う．その際に重みが最も大きいアラートのみをまとめないでそれ以外のアラートをまとめる．

5. 評価実験

評価実験ではシナリオを元に対象に対して障害を発生させる．最初に通知されたアラート件数を比較する．その後，通知されたアラートから重要なアラートが含まれる割合，エスカレーションにかかる作業量を比較する．評価実験のシナリオは物理機器に対する LAN ケーブルの接続不良と NFS サーバとクライアントにおけるファイルシステムの容量が 80%を超えた際に NFS サーバ側とクライアント側の両方でアラートが通知される 2 つである．

実験環境

実験で使用する環境を図 8に示す．仮想マシンには Ubuntu24.04 をインストールした．仮想マシンは同じ性能の (vCPU: 4[core],RAM:4[GB],SSD:25[GB]) で作成した．

ユーザのアクセスを再現するクラスタは仮想マシン上に K3s による Kubernetes クラスタを用意しマスタ 1 台，ワーカ 2 台の構成とした．クラスタ上には Locust を配置した．監視クラスタは仮想マシン上に K3s による Kubernetes クラスタを用意しマスタ 1 台，ワーカ 1 台の構成とした．クラスタ上には Prometheus と AlertManager，提案のソフトウェアを配置する．Prometheus は監視対象の監視を行い，対象に異常が出た際，AlertManager にアラートの通知をする．AlertManager は Prometheus から受け取った内容を管理者に通知する．提案ソフトウェアは AlertManager

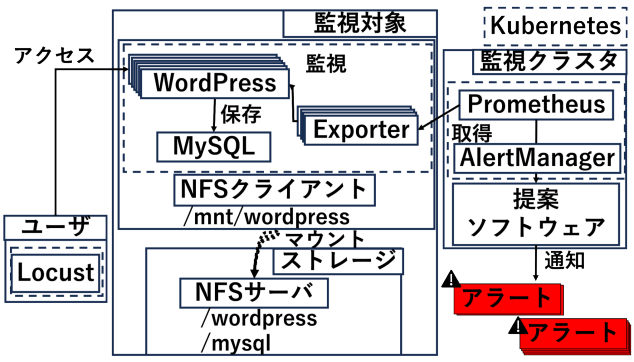

図 8: 実験環境

から通知されたアラートをまとめる．監視対象は仮想マシン上に K3s による Kubernetes クラスタを用意しマスタは 1 台のワーカ 2 台構成とした．NFS サーバ用の VM は 1 台とした. クラスタ上には WordPress と MySQL，NFS クライアントをそれぞれ配置する．WordPress のレプリカセットは 3，MySQL は 1 とした．この構成は論文要約サイトの構成を再現している．Prometheus に適応する監視条件は，26 項目ある．これは論文要約サイトを運用していた際の監視条件を再現している．

障害シナリオ

本実験で再現する障害事例は 2 つある．1 つ目が 2024 年 3 月 31 日に発生した物理機器である Mint の NIC と SW1 を繋ぐ LAN ケーブルが接続不良になった事例である．その際に Mint のハイパーバイザには VM1 から VM4 の 4 つの VM が監視対象として配置されていた．LAN ケーブルでこの箇所の通信ができなくなると Mint の ESXi 上に配置されている VM1，VM2，VM3，VM4 に対しても通信ができなくなる．その際に監視システムから物理機器である Mint と VM1 から VM4 までの 5 つの対象が疎通確認できないことを示すアラート通知された．

2 つ目の事例は NFS サーバとクライアント間で起きたストレージの容量不足の障害である．この事例では CDSL で運用されていた論文要約サイト上で画像データを WordPress 上に投稿した際に，NFS サーバを配置していた OS のファイルシステムの容量が 80%を超えた．その際に監視システムが NFS サーバ側とクライアント側のファイルシステムの容量を監視していた．またその際の閾値が 80%以上でアラートを通知する設定にしていたため，4 件のアラートが通知された．今回はディスク使用率が 80%を超えるように画像を送る実験を行う．画像を置く対象は画像プリントサイトを想定したサイトである．そのサイトを Kubernetes 上に Wordpreass と MySQL を配置して再現した．そのサイトには画像データを含む HTTP リクエストを送り，NFS のサーバが配置されている VM のファイルシステムの容量不足を発生させる障害を起こす. アクセスのシナリオはカメラのキタムラの画像データの事例を元にした. カメラのキタムラでは繁忙期の 11 月から 12 月の 60 日間で 120 万件の画像の処理依頼をうけ，それを 1 分間で換算すると約 13 件の画像の保存を行うこととなる[*1]. 画像枚数は，画像編集のサイトの無料枠の枚数が 20 枚であることから 20 枚ずつ送信する. また，画像 1 枚あたりの画像の容量は iPhone の場合，平均 2.5MB であるため，1 ユーザにつき 50MB の画像を送信する. 実験の時間は，NFS サーバ内の VM ストレージが 25GB であり，NFS サーバ内の VM のファイルシステムの使用率が 80%を超えた際にアラートを出すように設定している.25GB の 80%は 20GB である.NFS サーバが共有されている VM は最初の段階で 6.2GB を使用している状態であるため，ストレージ使用率が 80%を超えるためには 13.8GB 分の画像を送る必要がある. そのため，1 分間に 13 人のユーザが画像を 20 枚ずつ送ると 650MB となる. そのため 80%を超えるようにするためには約 52 分必要になる. そのため時間は 26 分とした.

[*1] https://chatbot.userlocal.jp/document/casestudy/kitamura/

実験結果と分析

評価実験では，通知されたアラート件数を比較する．その後，通知されたアラートから重要なアラートが含まれる割合，エスカレーションにかかる作業量を比較する．実際に実験で通知された項目は障害シナリオの Mint のケーブル障害では 9 項目ある．表 2に Mint の LAN ケーブルの接続不良の障害で通知された監視項目を示す．ICMP Connect

表 2: Mint の LAN ケーブルの接続不良の障害で通知された監視項目

監視項目	監視内容
ICMP Connect	ICMP のプロトコルをもちいた応答確認
ICMP Connect Time	応答確認にかかった時間 (1) 秒
ICMP Connect Time	応答確認にかかった時間 (3) 秒
SSH Connect	SSH のプロトコルをもちいた応答確認
SSH Connect Time	応答確認にかかった時間 (1) 秒
SSH Connect Time	応答確認にかかった時間 (3) 秒
HTTP Connect	SSH のプロトコルをもちいた応答確認
HTTP Connect Time	応答確認にかかった時間 (1) 秒
HTTP Connect Time	応答確認にかかった時間 (3) 秒

は ICMP のプロトコルをもちいた OS への応答確認の監視項目である．応答確認は SSH と HTTP のプロトコルに関しても監視している．応答確認はすべてのプロトコルの確認で実施した．

NFS サーバの OS 上のファイルシステムの容量不足の障害で通知されたアラートは 1 項目である．実際にアラートとして通知された監視項目を表 3に示す．

表 3: NFS サーバの OS 上のファイルシステムの容量不足の障害で通知されたアラートの項目

監視項目	監視内容
Disk usage	Disk 使用率の監視

通知されたアラートは Disk usage であり OS 上のファイルシステムの容量を監視する．

AlertManager のみをもちいた場合のアラートの通知件数は障害シナリオの Mint のケーブル障害では 39 件，ディスク使用率の障害では 4 件となった．また，AlertManager

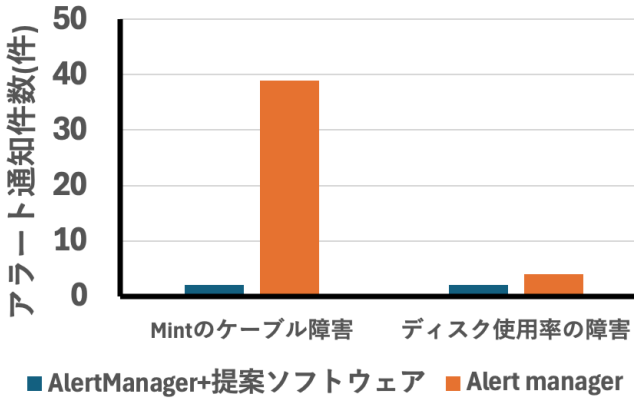

図 9: アラート通知件数

と提案ソフトウェアをもちいた場合のアラート通知件数は障害シナリオの Mint のケーブル障害では 2 件，ディスク使用率の障害では 2 件となった．結果的にアラートの通知件数を約 91%削減できた．

次に障害シナリオを再現した際に通知されたアラートの中から重要なアラートと関連して通知されたアラートの件数を比較する．比較は Mint のケーブル障害を再現した際と NFS サーバの OS 上のファイルシステムの容量不足の障害のそれぞれで比較する．図 10に Mint の障害シナリオ時における重要なアラートと関連するアラートの通知件数を示す．

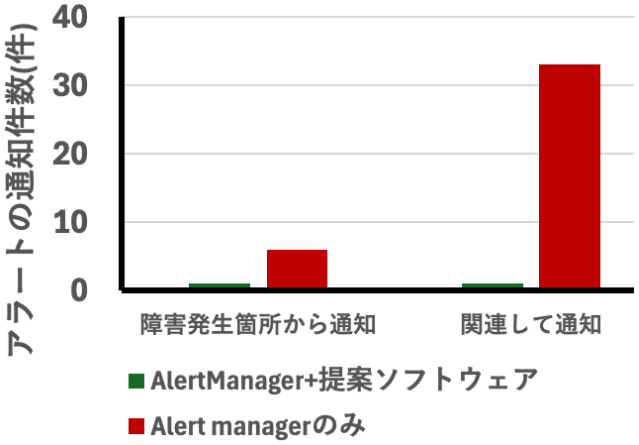

図 10: Mint の障害シナリオ時における重要なアラートと関連するアラートの通知件数

Mint のケーブルの接続不良の障害では，Mint とスイッチングハブの間を繋ぐケーブルが接続不良になったことで Mint に対してのアラートと Mint のハイパーバイザ上に配置された VM 上の OS に対しての 2 つのアラートが通知された．そのため Mint に接続しているケーブルの不良がハイパーバイザ上に配置されている VM の OS のアラートに影響を与えたと言える．そのため Mint の障害を示すアラートが重要なアラートとなり，それ以外アラートは関連するアラートとなる．関連して通知されたアラートが AlertManager と提案ソフトウェアをもちいた場合で 1 件であり，AlertManager のみの場合，38 件となった．また，障害発生箇所から通知された重要なアラートは AlertManager と提案ソフトウェアをもちいた場合，1 件で AlertManager のみをもちいた場合，6 件となった．

図 10に NFS サーバのファイルシステムの容量不足の障害における重要なアラートと関連するアラートの通知件数を示す．

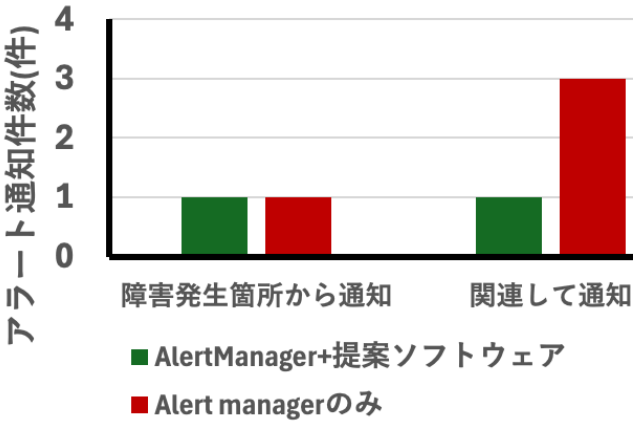

図 11: NFS サーバのファイルシステムの容量不足の障害における重要なアラートと関連するアラートの通知件数

NFS サーバのファイルシステムの容量不足の障害では，NFS サーバを配置した OS のファイルシステムをクライアントを配置した OS のファイルシステムが共有している．その際に NFS サーバ側の OS のファイルシステムの容量が 80%を超えたことでファイルシステムを共有するクライアント側でもアラートが通知された．その際に監視システムは NFS サーバ側と NFS クライアント側の 2 つのアラートを通知する．そのため NFS サーバ側の障害の通知アラートが重要なアラートとなり，それ以外のアラートが関連したアラートとなる．関連して通知されたアラートが AlertManager と提案ソフトウェアをもちいた場合，1 件であった．AlertManager のみの場合，3 件となった．障害発生箇所から通知されたアラートは AlertManager と提案ソフトウェアを用いる場合，1 件で AlertManager のみを用いる場合，1 件となった．この 2 つのシナリオでの結果から提案ソフトウェアが障害発生箇所のみのアラートとそれ以外のアラートで分けることができていると言える．

最後にエスカレーションにかかる作業量の評価である．エスカレーションにかかる作業量は通知されたアラートの中で重要なアラートがどれだけ通知されたかの割合で計算する．エスカレーションにかかる作業量の計算式を式 (3) に示す．本研究で実験した 2 つの障害シナリオごとに算出する．

$$Workload = \frac{\text{All Alerts} - \text{Important Alerts}}{\text{All Alerts}} \qquad (3)$$

Important Alerts は重要なアラートの件数である．All Alerts 通知されたアラートの全件数である．Mint の障害シナリオでは，障害発生箇所から通知されたアラートは AlertManager と提案ソフトウェアを用いる場合，1 件で AlertManager のみの場合，6 件となった．一方で関連して通知されたアラートが AlertManager と提案ソフトウェアが 1 件であり，AlertManager のみを用いる場合，38 件となった．

表 4: Mint の障害シナリオ時におけるエスカレーションにかかる作業量の比較

	作業量
AlertManager	0.84
AlertManager と提案ソフトウェア	0.5

AlertManager のみを用いる場合，エスカレーションにかかる作業量が 0.84 となった．一方で AlertManager と提案ソフトウェアを用いる場合，は 0.5 となった．NFS サーバの障害事例でのエスカレーションにかかる作業量では，障害発生箇所から通知されたアラートは提案ソフトウェアが 1 件で AlertManager が 1 件となった．一方で関連して通知されたアラートが提案ソフトウェアが 1 件であり，AlertManager が 3 件となった．

表 5: Mint の障害シナリオ時におけるエスカレーションにかかる作業量の比較

	作業量
AlertManager	0.75
AlertManager と提案ソフトウェア	0.75

エスカレーションにかかる作業量では AlertManager のみを用いる場合，0.75 となった．一方で AlertManager と提案ソフトウェアを用いる場合，0.5 となった．

6. 議論

本論文では 1 箇所の障害が原因で関連してアラートが通知されるシナリオを再現しアラートが通知される実験を行った．実験で扱う障害のシナリオは物理機器を繋ぐ LAN ケーブルの接続不良と NFS サーバにおけるディスクの容量不足の障害である．一方で原因箇所が 1 箇所でなくてもアラートが通知される事例は突発的なアクセス集中による CPU 使用率の増加もあげられる．実験では，WordPress と MySQL を使用し Locust を用いて画像を送信する実験を行った．その際に CPU 使用率は WordPress と MySQL の全ての Pod で増加した．本稿では障害発生箇所とそれ以外の箇所で通知されたアラートをそれぞれまとめて通知することでアラートの通知件数の削減を行った．WordPress はコンテンツの読み書きを MySQL に対して行うため MySQL がダウンしてしまうと，WordPress で読み書きができない．そのため WordPress と MySQL の Pod で通知されたアラートでそれぞれまとめて通知する必要がある．WordPress と MySQL の関係を取得する方法として ss コマンドをコンテナ上に配置し，どの対象に対して通信を行っているのかを取得する．また通信を行う対象を決定し依存関係を取得することで重みをつけることでコンテナ間の依存関係も特定できる．

本論文では構成管理データベースの内容から依存関係を特定し，重みづけを行なった．その際に構成管理データベースの中のハイパーバイザの IP アドレスと VM の IP アドレス，OS 上のアプリケーションがどの対象に対して接続するかの情報を用いて依存関係を特定した．一方で，これらの構成要素は依存関係を構築する際に対象ごとのレイヤの違いを考慮していない．レイヤの違う要素同士で依存関係を作る場合，それぞれの対象が共通する要素や正規化することが求められる．解決方法として対象それぞれに対して追加のスコアを掛け合わせてから正規化することで解決できる．例えばハイパーバイザのレイヤは VM のレイヤよりも低い位置に存在するためハイパーバイザのスコアをかける 1 する．一方でその下のレイヤにいる VM には 0.8 をかける．このように対象ごとにスコアをかけてから正規化することでレイヤの違いのない依存関係にできる．

7. おわりに

課題は障害の発生箇所が 1 箇所であることに対して障害の発生箇所ではない監視対象からアラートが通知されることである．障害の発生箇所以外からアラートが通知されることで，エスカレーションにかかる時間が増加する．本稿ではシステム同士の構成を管理する構成管理データベースから依存関係を取得し重みを算出する．また通知されたアラートの中から重みが大きい対象のアラート同士をまとめることでアラートの通知件数を削減する．評価実験ではシナリオを元に対象に対して障害を発生させる．評価指標は，通知されたアラート件数である．その後，通知されたアラートから重要なアラートと関連するアラートの通知件数，エスカレーションにかかる作業量の 2 つを評価する．評価実験のシナリオは物理機器に接続する LAN ケーブルの接続不良と NFS サーバを配置する OS のファイルシステムの容量が 80%を超えた際に NFS サーバ側とクライアント側の両方でアラートが通知される 2 つである．通知されたアラートの件数で比較対象の AlertManager のみをもちいた場合のアラートの通知件数は LAN ケーブルの接続不良の障害シナリオでは 39 件となり，NFS サーバを配置している OS のファイルシステムの容量が 80%を超えたシナリオでは 4 件となった．AlertManager と提案ソフトウェアを組み合わせた場合のアラート通知件数は LAN ケーブルの接続不良の障害シナリオでは 2 件，NFS サーバを配置

している OS のファイルシステムの容量が 80%を超えたシナリオでは 2 件となった．結果的にアラートの通知件数を約 91%削減できた．また通知されたアラートをまとめた際の LAN ケーブルの接続不良の障害シナリオでは，関連して通知されたアラートが AlertManager のみの場合が 33 件となり，AlertManager と提案ソフトウェアをもちいた場合が 1 件であった．障害発生箇所から通知された重要なアラートは AlertManager と提案ソフトウェアをもちいた場合が 1 件で AlertManager のみの場合が 6 件となった．エスカレーションにかかる作業量は AlertManager と提案ソフトウェアの場合が 0.5，AlertManager のみの場合が 0.84 となった．

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] Oostenbrink, J.: Financial impact of downtime decrease and performance increase of IT services (2015).

[4] 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).

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

[6] 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).

[7] SPADACCINI, A. and GULIANI, K.: Being an On-Call Engineer, *Operating Systems and Sysadmin*, p. 43.

[8] Huang, H., III, R., Ruan, Y., Sahoo, R., Sahu, S. and Shaikh, A.: PDA: A Tool for Automated Problem Determination., pp. 153–166 (2007).

[9] Srinivas, P., Husain, F., Parayil, A., Choure, A., Bansal, C. and Rajmohan, S.: Intelligent Monitoring Framework for Cloud Services: A Data-Driven Approach, *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '24, New York, NY, USA, Association for Computing Machinery, p. 381–391 (online), DOI: 10.1145/3639477.3639753 (2024).

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

[11] 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 (2020).

[12] Ahmed, T., Shah, A., Kolla, M. and Yellasiri, R.: Reduction of Alert Fatigue using Extended Isolation Forest, *2021 International Conference on Forensics, Analytics, Big Data, Security (FABS)*, Vol. 1, pp. 1–5 (online), DOI: 10.1109/FABS52071.2021.9702617 (2021).

[13] Saputra, M., Noprianto, N., Arief, S., Wijayaningrum, V. N. and Syaifudin, Y.: Real-Time Server Monitoring and Notification System with Prometheus, Grafana, and Telegram Integration, pp. 1808–1813 (2024).

[14] 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).

[15] 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).

[16] Jiang, G., Chen, H., Yoshihira, K. and Saxena, A.: Ranking the importance of alerts for problem determination in large computer systems, *Cluster Computing*, Vol. 14, pp. 213–227 (2009).

[17] 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 (2023).

[18] 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, *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 162–171 (2020).

[19] 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).