サーバへのSSH接続時間にもとづく順位付けによるアラートの並べ替え

加藤 健吾[1]　平尾 真斗[2]　串田 高幸[1]

概要：Cloud and Distributed Systems Laboratory(以下 CDSL と呼ぶ) では，7 台の物理サーバの死活監視を行っている．監視サーバとして，Prometheus と Alertmanager をもちいている．CDSL の学生は，物理サーバに対して ICMP による通信エラーのアラートを受け取ると，物理サーバの再起動や電源の起動を行うことで対処する．Alertmanager はアラートを severity ごとにグループ化する設定を行っているため，すべてのアラートの severity が同じになることがある．課題は，グループ化されたアラートの severity が全て同じ値で設定されているため，オンコール担当の学生がどのアラートを解決するべきなのか判断ができないことである．そのため，解決されるべきアラートのエスカレーションが後回しになり，MTTR が増加する．本稿では，物理サーバごとに，障害が発生する前に学生がサーバに SSH 接続した時間の割合を表す直前接続度を設定し，直前接続度の値が大きいものから順番に，順位をつける方式を提案する．直前接続度は，サーバへの SSH 接続時間が，障害発生時刻から遡る時間内のどのくらいの割合を占めるのかという VM ごとの値であり，VM ごとの直前接続度を平均したものが物理サーバごとの直前接続度となる．また，順位の高いものから並べたアラートを通知する．評価実験では，各物理サーバにおいて VM を所有している CDSL の 4 年生の人数をもとにアラートの通知順の指標を作成した．そして作成したアラートの通知順の指標に対して，提案ソフトウェアから通知されるアラートの通知順が一致するかどうかを比較した．その後，いくつのアラートが通知順の指標と一致しているかを指標との一致率として表し，指標との一致率を評価する．指標との一致率は，グループ内の全アラートの件数のうち，いくつのアラートが指標と同じ通知順であったのかを，百分率で表す．実験内容は，物理サーバとスイッチングハブを接続する LAN ケーブルを引き抜くことで，ICMP による通信エラーのアラートを通知することである．結果は，6 つのアラートのうち，2 つのアラートの通知順が指標の通知順と一致したため，指標との一致率が約 33%であった．

1. はじめに

背景

IT サービスは，ユーザに利用を継続させるために信頼性を保つ必要がある．IT サービスへのアクセスができなくなると，ユーザは不便を感じ，サービスへの信頼性が低下するとともに，大きな損失になる．実際の例として，2008 年 3 月に，ヒースロー空港の手荷物システムでシステム障害が発生した [1]．損失として，3200 万ドルの損害が発生し，14 万人が影響を受けたと推定されている．システム管理者は，こうした IT サービスの障害や，障害に伴う損失を最小限に抑えるために，システムの監視を行う [2]．

監視ツールの 1 つに Prometheus がある．Prometheus は Cloud Native Computing Foundation の Graduated プロジェクトであり，オープンソースのツールとして使用されている [3–5]．Prometheus のエコシステムは主に以下のコンポーネントから構成されている [6]．

- Prometheus サーバ：ジョブからメトリクスをスクレイピングし，時系列データとして収集，保持する
- Alertmanager：主に Prometheus サーバが生成したアラートを，指定された方法で送信する
- Exporter：監視対象における時系列データの収集を行う
- Grafana：Prometheus サーバが保持する時系列データの視覚化を行う

Alertmanager はアラートのグループ化や，サイレンス，抑制をする機能を持つ [7–9]．アラートのグループ化機能では，アラートを受信してから送信を行うまでの待ち時間を指定することができる．グループ化機能により，待ち時間中に生成されたアラートを 1 つのグループとすることが

[1] 東京工科大学コンピュータサイエンス学部
クラウド・分散システム研究室
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻
〒 192–0982 東京都八王子市片倉町 1404-1

できる．また，時間だけでなく，アラートの重大度を表す severity やアラート名，job ごとにグループ化することもできる．そのため，severity でグループ化した場合，すべてのアラートの severity が同じになる．

システム障害によってアラートが通知されると，1 次対応を担当するオンコールエンジニアがアラートの対応を行う．1 次対応では，Runbook に沿った手順化された対処や原因調査を行う [10,11]．具体的には，サーバの再起動やコマンドの実行と実行結果の記録があげられる．1 次対応でシステム障害を復旧できなかった場合，1 次対応を担当するオンコールエンジニアはエスカレーションを行い，2 次対応担当のオンコールエンジニアに引き継ぐ [12]．

1 次対応を担当するオンコールエンジニアは，アラートを最初に受け取る．受け取ったアラートを確認し，それぞれのアラートが対応すべきかどうかの判断をする．この時，確認するアラートの順番として，オンコールエンジニアは severity が高いものから順番に確認する [13]．例えば，severity が critical，warning の 2 種類あり，severity が critical である ICMP による通信エラーのアラートと，severity が warning である ICMP による通信の応答時間が長いことを示すアラートが通知されているとする．この時，オンコールエンジニアは，2 つのアラートの並び順に関わらず，severity が critical のアラートから 1 次対応を行う．

CDSL では，7 台の物理サーバにブロードコム社の VMware 製品である ESXi が導入されている．VMware ESXi は，VMware vSphere のコンポーネントの 1 つで，ハイパーバイザである [14–16]．学生は ESXi 上に Virtual Machine(以下 VM と呼ぶ) を作成して研究に使用している．物理サーバは以下の通りである．

- Plum
- Lotus
- Rose
- Jasmine
- Violet
- Mint
- Lily

CDSL では，監視対象として上記の 7 つの対象を設定している．監視項目は表 1 のようになっている．

表 1: 監視項目

アラート名	PromQL 式	アラートを通知させるために PromQL 式が成立した状態が継続する期間	重大度
icmp-check	probe_success == 0	3 分	critical

アラート名は icmp-check，PromQL 式は probe_success の値が 0 と等しいこと，アラートを通知させるために PromQL 式が成立した状態が継続する期間は 3 分，重大度は critical になっている．このような環境において，図 1 のような障害が発生した．

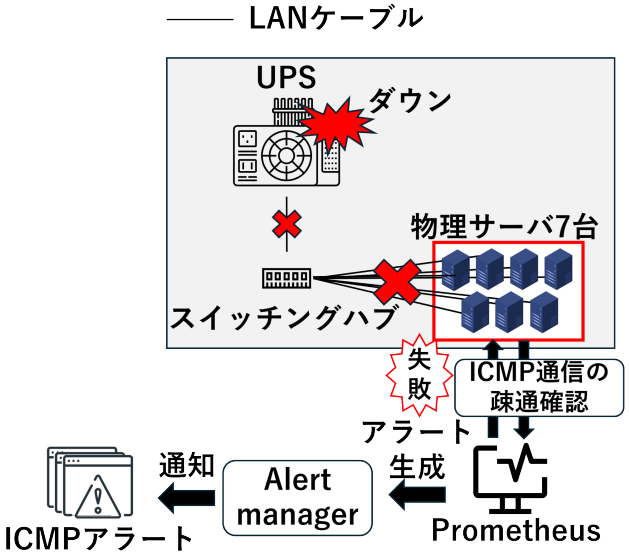

図 1: アラートが通知された経緯

停電時に一定時間, 電力の提供を行う Uninterruptible Power Supply(以下 UPS と呼ぶ) がダウンした．これによってスイッチングハブへの電力供給が行われなくなり，各物理サーバに対しての通信経路が断たれた．Prometheus は各物理サーバに ICMP による通信の疎通確認を行うが，ICMP による通信に失敗した．そのため，Alertmanager から ICMP による通信エラーのアラートが通知された．また，severity は critical，warning，attention の 3 段階あり，critical が最も高く，attention が最も低くなっている．通知されたアラートは，各物理サーバに対して ICMP による通信ができないことを示す，severity が critical のアラートである．

課題

課題は，グループ化されたアラートの severity が全て同じ値で設定されているため，オンコール担当の学生がどのアラートを解決するべきなのか判断ができないことである．そのため，解決されるべきアラートのエスカレーションが後回しになることがある．解決されるべきアラートのエスカレーションが後回しになることにより，Mean Time To Repair(以下 MTTR と呼ぶ) が増加し，学生がサーバに対して SSH 接続できない時間が長くなる．解決されるべきアラートは，障害が発生する前にサーバに SSH 接続されていた時間の割合が大きい物理サーバのアラートとする．

図 2 はオンコール担当の学生がどのアラートを解決するべきなのか判断ができない状況を示している．これらのアラートはすべて重大度を示す severity が critical である．

そのため，オンコール担当の学生はどのアラートを解決するべきなのか判断がついていない．

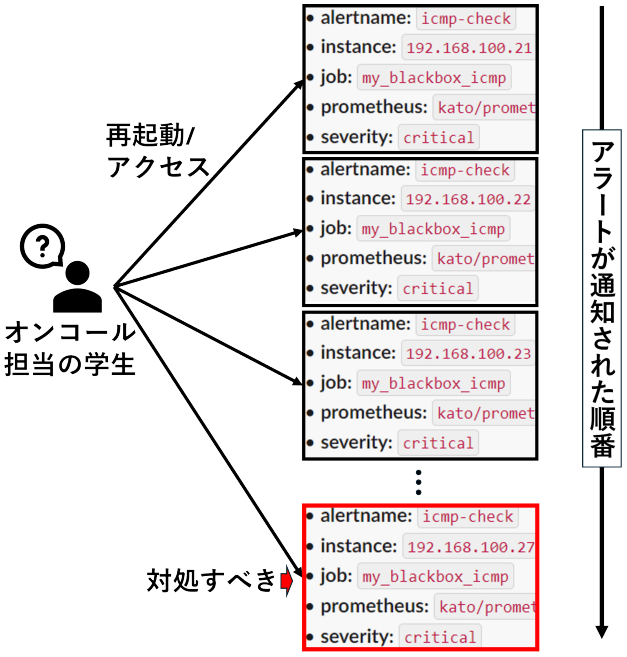

図 2: オンコール担当の学生がどのアラートを解決するべきなのか判断ができない状況

ここで，IP アドレスが 192.168.100.27 の物理サーバの，障害が発生する前にサーバに SSH 接続されていた時間の割合が最も大きいとする．この時，最も解決されるべきアラートは，IP アドレスが 192.168.100.27 である物理サーバに対して ICMP による通信ができないことを示すアラートである．図 2 の状況では，最も解決されるべきアラートの解決が遅くなってしまい，MTTR の増加につながる．解決されるべきアラートの解決が遅くなることで，可用性が低下する．

各章の概要

本稿は以下のように構成される．第 2 章では，本稿の関連研究について述べる．第 3 章では，本稿であげた課題を解決するための提案について述べる．第 4 章では，提案した手法の実装について述べる．第 5 章では，提案した手法に対しての実験内容と，その評価について述べる．第 6 章では，提案した手法の議論を述べる．第 7 章は，本稿のまとめである．

2. 関連研究

セキュリティ監視を実現する SIEM から通知された大量のアラートに対して，以前のアラートの重大度，アラートエンティティ，及び重要度評価を使用して優先度の値を計算することで，セキュリティイベントに優先順位を付けるスライディングウィンドウ手法を提案している研究がある [17]．この提案により，セキュリティアラートの優先順位付けが改善され，重要なシステムに影響する重大及び中程度のアラートが，重要でないシステムに影響する低，高，及び重大なアラートよりも優先されることがわかる．この研究はセキュリティアラートに関しての優先順位付けを行っている一方，ICMP による通信のアラートのような死活監視についての優先順位付けを行うことが改善の余地である．

セキュリティ管理者が，アラートの評価及び管理をする時間と労力を削減するために，IDS によって生成された大量のアラートをスコアリングして優先順位を付けるファジーロジックベースの手法を提案し，アラートの数をさらに削減できるアラートの再スコアリング手法を紹介している研究がある [18]．この提案は，セキュリティ管理者に最も重要なアラートを提供できることを示した．一方で，この研究ではネットワーク IDS によって生成されたアラートの評価に重点を置いているため，Prometheus によって生成されたアラートを，Alertmanager が優先順位付けを行うことが改善の余地である．

最も重要なアラートをキューの先頭に配置することで、アラートの感度低下を軽減することを目標に，サイバーセキュリティアラートをランク付けして、アナリストの時間と労力が最も重要なアラートに集中するようにするための教師あり機械学習の使用について説明している研究がある [19]．この研究において，特徴評価の実験では，LDA または時間にもとづく特徴のみがランダム特徴よりも重要であることが示されている．この研究では，アラートのランク付けをするための教師あり機械学習の使用についての説明はしているが，実際にランク付けを行うことが改善の余地である．

アナリストがアラートに関する出所分析タスクを効率的かつ協調的に実行できるように支援するリアルタイムアラート調査システムである RAPID を提案している研究がある [20]．RAPID は，スペース効率を最大 3 桁向上させ，すべての主要な攻撃の痕跡を発見するためのアラート出所分析の時間を最大 99%短縮できることを示した．この研究では，攻撃の出所を分析するために使用する優先順位は設定しているが，優先して対処すべきアラートの判定まで行うことが改善の余地である．

3. 提案方式

本稿では，一定時間ごとにグループ化されたアラートにおいて，グループ内での相対的な順位を設定することで，オンコール担当の学生がどのアラートから解決するべきなのか判断ができるようにすることを目的とした．

目的を達成するため，各物理サーバに，アラートが通知される前に学生がサーバに SSH 接続した時間の割合を表す直前接続度を設定する．そして，直前接続度の値が大きい

ものから順番に高い順位を割り当てる方式を提案する．この時，順位は 1 が最も高く，順位が 1 増えるごとに下がっていくものとする．

提案方式

図 3 に提案方式の手順を示す．

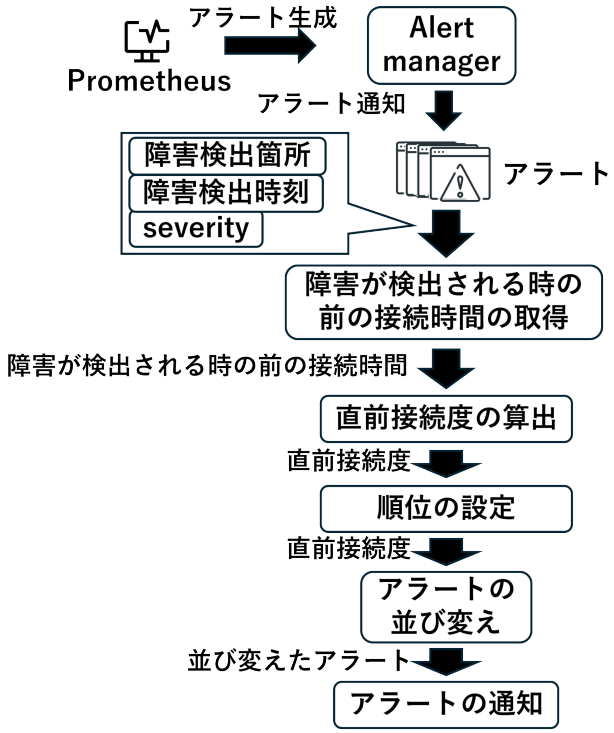

図 3: 提案方式の手順

Prometheus がアラートを生成し，Alertmanager がアラートを通知すると，まず各物理サーバにおける障害が検出される前に学生がサーバに SSH 接続した時間を取得する．この時，Alertmanager から受け取ったアラートから，障害検出箇所，障害検出時刻，severity を使用する．次に，各物理サーバの直前接続度の算出を行う．障害が検出される時刻の前の時間のうち，学生がサーバに SSH 接続した時間の割合を直前接続度とする．直前接続度の算出には，各物理サーバの，障害が検出される時刻の前に学生がサーバに SSH 接続した時間を使用する．次に，算出された直前接続度の値が最も大きいものに，1 を最大の順位として割り当て，直前接続度が小さくなるにつれて順位を 1 ずつ増加させる．次に，アラートを順位の昇順に並び替える．最後に，順位が高い順番に並び変えられたアラートを出力する．

入力として以下を使用する．

- アラートに含まれる，障害の検出箇所と検出時刻，severity
- 障害が検出される時刻の前に学生がサーバに SSH 接続した時間

アラートは，severity でグループ化されているものとする．

障害発生箇所における障害検出時刻より前に学生がサーバに SSH 接続した時間の取得

各障害発生箇所の各サーバに学生が SSH 接続した時間から，障害検出時刻から遡る時間 (以下 upstream[分] と呼ぶ) 内に学生がサーバに SSH 接続した時間を取得する．ここで，upstream[分] の時間内の SSH 接続を，取得する SSH 接続時間の対象とする．図 4 は学生からのサーバへの SSH 接続時間を取得する時間を示している．

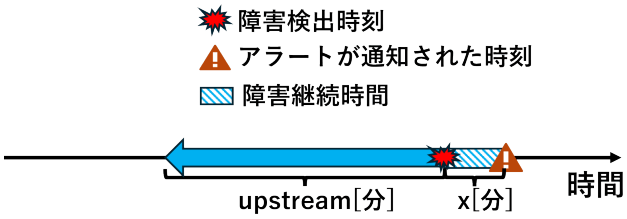

図 4: 学生からのサーバへの SSH 接続時間を取得する時間

x[分] は，Prometheus のアラートルールで管理者が設定する，アラートを通知させるために PromQL 式が成立した状態が継続する期間の値と，Alertmanager がグループ化をするためにアラートを待機する時間である group_wait の値の和とする．

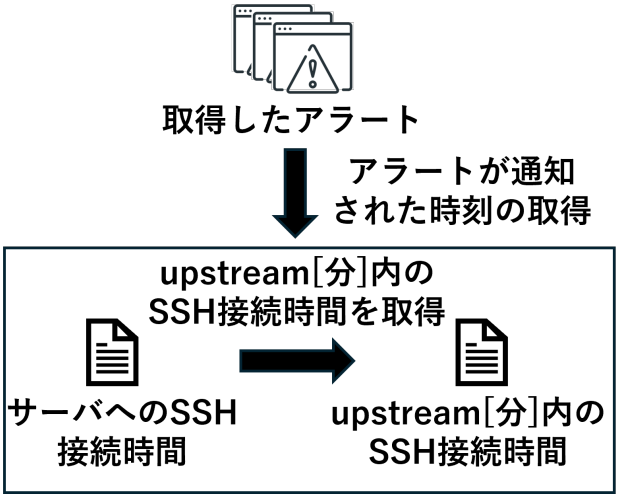

図 5: upstream[分] の時間内に学生がサーバに SSH 接続した時間の取得

アラートが通知された時刻が一番右に位置しており，アラートが通知された時刻の x 分前に障害が検出されている．この間の x 分間は，障害継続時間であり，アラートはこの x 分間障害が継続することで通知される．また，アラートが通知された時刻から x 分前の時刻を，障害検出時刻とする．障害検出時刻を起点とする upstream[分] は，学生からのサーバへの SSH 接続を取得する時間範囲である．次に，図 5 は upstream[分] の時間内に学生がサーバに SSH 接続した時間の取得方法を示す．まず，取得したアラートから，

アラートが通知された時刻を取得する．次に，各サーバへの SSH 接続時間から，upstream[分] の時間内の接続時間を取得する．

直前接続度の算出

upstream[分] の時間内のうち，学生がサーバに SSH 接続した時間の割合を直前接続度と定義し，小数の値として算出する．図 6 に直前接続度の算出時の手順を示す．

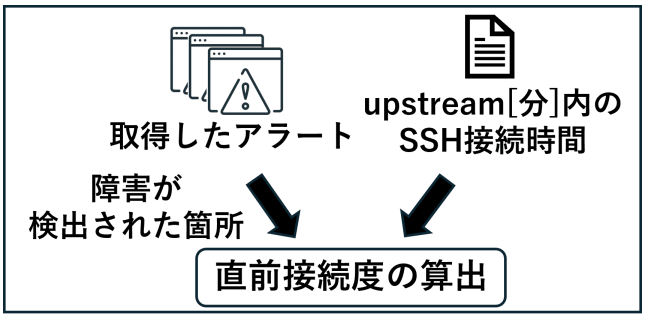

図 6: 直前接続度の算出

直前接続度の算出には，Alertmanager から取得したアラートの障害が検出された箇所と，取得した upstream[分] の時間内の SSH 接続時間を使用する．次に，直前接続度の算出方式を数式 (1) に示す．

$$B = U[分] \div upstream[分] \qquad (1)$$

B は VM の直前接続度を表し，U[分] は upstream[分] の時間内に学生がサーバに SSH 接続した時間を表している．VM の直前接続度は，upstream[分] の時間内に学生がサーバに SSH 接続した時間を，upstream[分] で割ることで算出する．この時，小数第 3 位を四捨五入して小数第 2 位まで求める．次に物理サーバの直前接続度を算出する．物理サーバの直前接続度は，物理サーバ内の VM の直前接続度の平均を使用する．

次に，直前接続度の算出例をあげる．図 7 は，算出例を示している．まず，物理サーバである Plum，Rose，Lotus，Jasmine において障害が発生したとする．図 7 の例では，x[分] の値を 8 分，upstream[分] の値を 50 分とし，Plum と Lotus の算出例を説明する．Plum では，紫の学生と黒の学生がサーバに SSH 接続して作業している．紫の学生は，50 分間のうち 14 分間作業した．そのため，この VM の直前接続度は，14 を 50 で割り，0.28 となる．黒の学生は，50 分間のうち，作業をしていない．そのため，この VM の直前接続度は，0 を 50 で割り，0.00 となる．よって，Plum の直前接続度は，0.28 と 0.00 の平均を取り，0.14 となる．Lotus では，緑の学生と紫の学生がサーバに SSH 接続して作業している．緑の学生は，50 分間のうち 15 分と 14 分の合計 29 分間作業した．そのため，この VM の直前接続度は，29 を 50 で割り，0.58 となる．紫の学生は，50 分間のうち，32 分間作業した．そのため，この VM の直前接続度は，32 を 50 で割り，0.64 となる．よって，Lotus の直前接続度は，0.58 と 0.64 の平均を取り，0.61 となる．

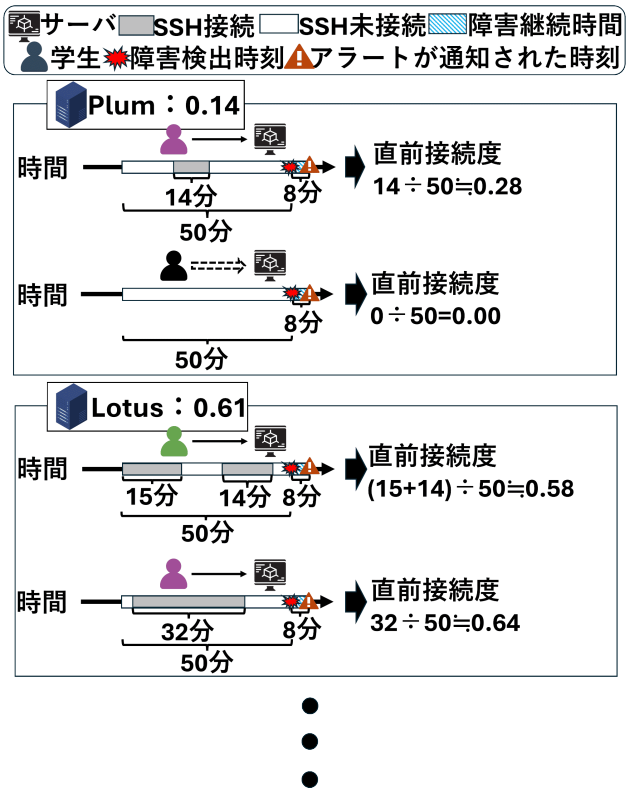

図 7: 直前接続度の算出例

順位の設定

順位は，算出された障害発生箇所の直前接続度が大きいほど，並び順が上になるように設定する．直前接続度の値が大きいものから順番に，自然数の昇順を順位として設定する．そのため，順位が 1 のものが最も並び順が上になり，順位が 1 増えるごとに並び順が下がっていく．表 2 に各物理サーバに設定される順位の例を示す．

表 2: 順位の設定

物理サーバ名	直前接続度	順位
Jasmine	0.94	1
Lotus	0.61	2
Rose	0.54	3
Plum	0.14	4

直前接続度が Jasmine，Rose，Lotus，Plum の順番で大きい．そのため，Jasmine，Rose，Lotus，Plum の順番で，順位は 1，2，3，4 と設定される．

出力

出力は，順位が高い順番に並び変えられたアラートを通知する．図 8 は出力されるアラートを示す．

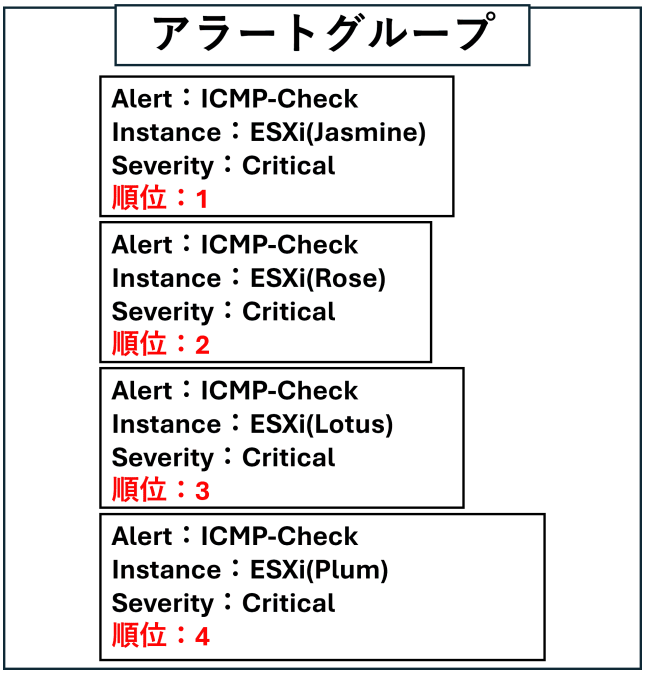

図 8: 出力されるアラート

各アラートには，順位が設定されている．アラートは順位の高い順番に並べ替えられている．アラートはグループとしてまとめられている．この時，順位が 1 のアラートが一番上に並ぶようにする．

ユースケース・シナリオ

CDSL の環境を想定する．物理サーバは 7 台あり，各物理サーバに VMware ESXi を導入している．CDSL に所属する学生は ESXi 上に VM を作成し，サーバに SSH 接続して作業を行う．7 台の物理サーバの IP アドレスは以下のようになっている．

- 192.168.100.21
- 192.168.100.22
- 192.168.100.23
- 192.168.100.24
- 192.168.100.25
- 192.168.100.26
- 192.168.100.27

図 9 にユースケース・シナリオを示す．7 台の物理サーバが同一のスイッチングハブに接続されている．7 台が接続しているスイッチングハブは，電力を UPS 経由で受け取っている．7 台の物理サーバの ESXi は NAS に対してマウントしている．NAS は電力を UPS 経由で受け取っている．また，基幹サーバには Prometheus が配置されており，Prometheus は 7 台の物理サーバを監視している．

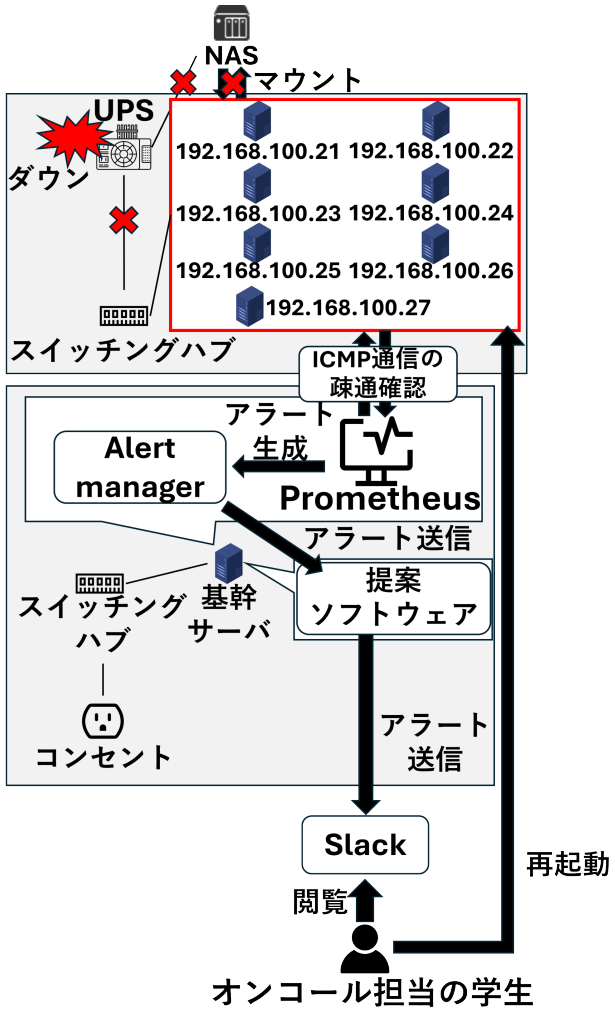

図 9: ユースケース・シナリオ

基幹サーバは 7 台の物理サーバが接続しているスイッチングハブとは別のスイッチングハブに接続されており，このスイッチングハブは UPS には接続されていない．この状況下で UPS が落ちた時，UPS に接続されているスイッチングハブと NAS が電力の供給を断たれ，機能を停止する．スイッチングハブが機能を停止したことにより，7 台の物理サーバへの ICMP による通信ができなくなる．Prometheus はこの 7 台の物理サーバへの ICMP による通信が x 分間できなかった場合に，7 台の物理サーバのアラートを生成する．Alertmanager は，Prometheus から受け取ったアラートをグループ化して提案ソフトウェアへ送信する．提案ソフトウェアは，7 台の物理サーバの障害が検出される前に学生がサーバに SSH 接続した時間をもとに，各アラートに順位を設定し，順位が高い順番に並び替える．また，並び替えたアラートを通知する．その後，オンコール担当の学生によって UPS が復旧された．これにより，スイッチングハブ及び NAS が復旧する．そして，オンコール担当の学生は各物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．この時，UPS

が落ちたときに通知されたアラートは，順位の高い順番に並んでいるため，オンコール担当の学生は，このアラートの順番に確認する．通知されたアラートを図 10 に示す．

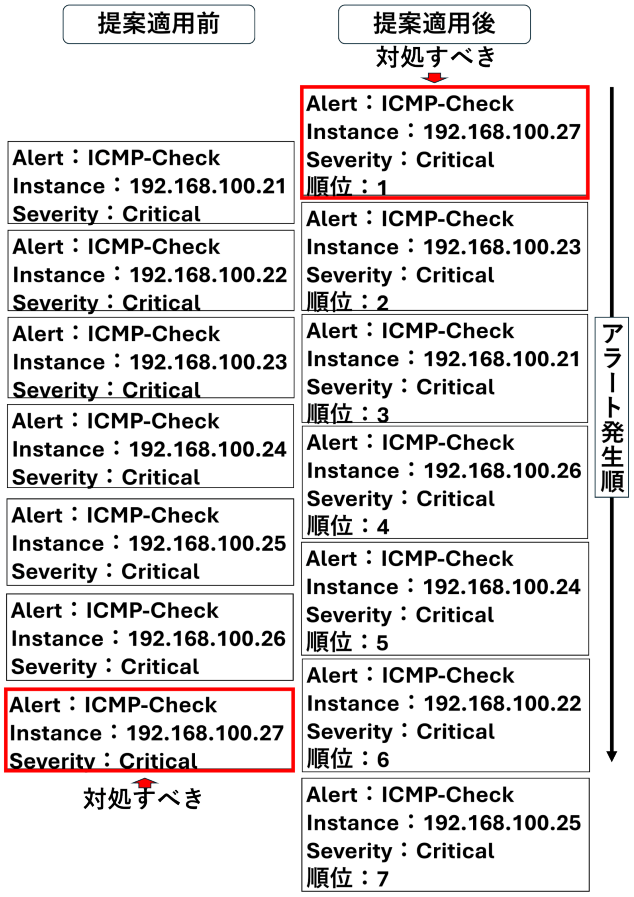

図 10: 通知されたアラート

オンコール担当の学生はまず, IP アドレスが 192.168.100.27 である物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．次に，IP アドレスが 192.168.100.23 である物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．次に，IP アドレスが 192.168.100.21 である物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．次に，IP アドレスが 192.168.100.26 である物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．次に，IP アドレスが 192.168.100.24 である物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．次に，IP アドレスが 192.168.100.22 である物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．次に，IP アドレスが 192.168.100.25 である物理サーバの ESXi 上にある VM が正常に NAS に対してマウントしているかどうか確認する．

4. 実装

Python を使用して，提案ソフトウェア CutIner を実装した．CutIner では，get_alert.py が動作する．図 11 は，ソフトウェアの全体の構成を示している．

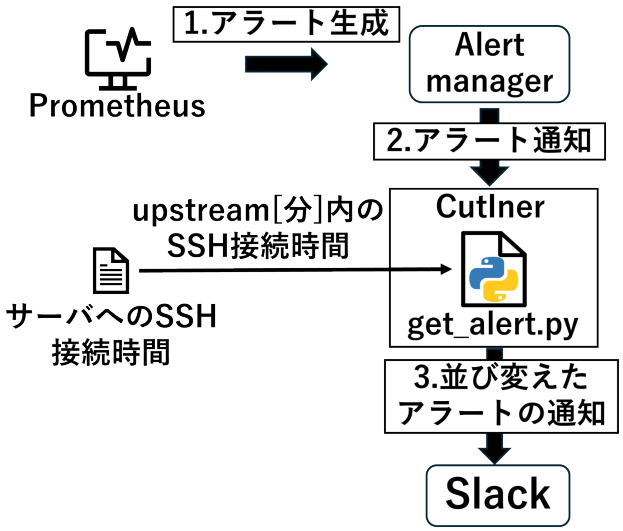

図 11: 実装の全体図

まず, Prometheus がアラートを生成し, Alertmanager に送信する．Alertmanager は取得したアラートを get_alert.py に送信する．get_alert.py では，以下のことを行う．

- アラートの取得
- 各サーバへの SSH 接続時間から，upstream[分] の時間内に学生がサーバに SSH 接続した時間を取得
- 直前接続度の算出
- 順位の設定
- 順位の高い順番で並び変えたアラートの通知

アラートの取得には，Python の HTTPServer ライブラリを使用した．get_alert.py に送信されるアラートは，Alertmanager の設定で送信方法を webhook にすることで，HTTPServer で作成した web サーバにアラートが送信されるようにした．

各サーバへの SSH 接続時間から，upstream[分] の時間内に学生がサーバに SSH 接続した時間を取得する関数では，ファイル名の後ろ 4 文字が LAST であるファイルを読み込み，1 行ずつ SSH 接続時間を取得する．そして，SSH 接続時間がアラートが通知された時刻から $x + upstream$ 分前〜x 分前の間に入っていれば，その SSH 接続時間を取得する．

直前接続度の算出では，障害発生箇所のホスト名と，障害発生箇所の各サーバへの SSH 接続時間を使用した．直前接続度は，小数第 3 位を四捨五入し小数第 2 位まで求めるため，四捨五入を行う関数を作成した．四捨五入を行う関数には，decimal モジュールを使用した．

順位の設定は，算出した直前接続度を降順に並べ替え，先頭から自然数の昇順を割り当てた．

順位で並び変えたアラートの通知では，まずホスト名ごとに割り当てられた順位を，Alertmanager から取得したアラートから，アラート名，障害発生箇所，severity，アラートの詳細，アラートの Runbook の URL を取得する．次に，通知するアラートの内容を，取得した 5 項目に順位を追加したものとして text 形式で作成する．最後に，作成したアラートを，Webhook URL で指定された Slack へと通知する．

5. 評価実験

アラートの通知順の指標を作成し，提案ソフトウェアから通知されるアラートの通知順が指標と一致しているかを表す指標との一致率を評価する．指標との一致率は，アラートの通知順の指標と，提案ソフトウェアによって通知されたアラートの通知順を比較して，通知順が一致していた割合を百分率で表す．ここでアラート通知順の指標は，ESXi 上に VM を所有している CDSL の 4 年生の人数が多い物理サーバについてのアラートほど通知順が先になるとする．4 年生の人数が多い物理サーバほど順位を高くする理由として，4 年生は卒業論文の提出期限が迫っているため，早急に 4 年生が使用する VM を使用可能にする必要があると判断したためである．ESXi 上に VM を所有している CDSL の 4 年生の人数が同数の物理サーバが存在している場合，3 年生の人数を比較し，人数が多い方の物理サーバについてのアラートの通知順を先にする．これは，3 年生はテクニカルレポートという技術論文の締め切りが迫っているため，できるだけ早急に 3 年生が使用する VM を使用可能にする必要があると判断したためである．

指標との一致率の算出式を数式 (2) に示す．

$$M[\%] = T \div N \times 100 \tag{2}$$

M[%] は指標との一致率を表す．T は通知されたアラートが，アラートの通知順の指標と一致していた数を表す．N はグループ内のアラートの総数である．T を N で割ることで，通知されたアラートが，アラートの通知順の指標と一致していた割合を表す．そして 100 を乗算することで，百分率で指標との一致率を表す．

表 3 に物理サーバ名，VM を所有している 4 年生の人数，VM を所有している 3 年生の人数，アラートの通知順の指標を示す．Lily についてのアラートは，VM を所有している 4 年生の人数が 8 人であり，1 番目に多いため，1 番目に通知される．Rose についてのアラートは，VM を所有している 4 年生の人数が 7 人であり，Mint の VM を所有している 4 年生の人数と同数である．そのため，VM を所有している 3 年生の人数を比較する．Rose の VM を所有している 3 年生の人数は 6 人であり，Mint の VM を所有している 3 年生の人数は 4 人である．

表 3: アラートの通知順の指標

物理サーバ名	VM を所有している 4 年生 [人]	VM を所有している 3 年生 [人]	アラートの通知順の指標
Lily	8	9	1
Rose	7	6	2
Mint	7	4	3
Jasmine	6	0	4
Lotus	5	4	5
Plum	4	4	6

そのため，VM を所有している 3 年生の人数が多い Rose についてのアラートの方が順位が高くなり，2 番目に通知される．また，Mint についてのアラートは 3 番目に通知される．Jasmine についてのアラートは，VM を所有している 4 年生の人数が 6 人であり，4 番目に多いため，4 番目に通知される．Lotus についてのアラートは，VM を所有している 4 年生の人数が 5 人であり，5 番目に多いため，5 番目に通知される．Plum についてのアラートは，VM を所有している 4 年生の人数が 4 人であり，6 番目に多いため，6 番目に通知される．

評価実験の手順を述べる．前提として，ユースケースシナリオにおいて基幹サーバに配置されていた提案ソフトウェアは，IP アドレスが 192.168.100.25 である Violet の ESXi 上にある VM に配置されている．そのため，評価実験において Violet についてのアラートは通知しないものとする．まず，提案ソフトウェアを使用しない場合で，6 台の物理サーバにおける ICMP による通信エラーのアラートを通知させる．アラートを通知させる物理サーバは以下の 6 台である．

- Plum(IP アドレス：192.168.100.21)
- Jasmine(IP アドレス：192.168.100.22)
- Rose(IP アドレス：192.168.100.23)
- Lotus(IP アドレス：192.168.100.24)
- Mint(IP アドレス：192.168.100.26)
- Lily(IP アドレス：192.168.100.27)

ユースケース・シナリオを再現するため，物理サーバ 6 台において，スイッチングハブに接続している LAN ケーブルを引き抜くことで ICMP による通信エラーのアラートを通知させる．次に，提案ソフトウェアを使用した場合で ICMP による通信エラーのアラートを通知させる．方法は提案ソフトウェアを使用しない場合と同じである．

実験環境

実験環境を図 12 に示す．まず監視クラスタがあり，監視クラスタ内では Prometheus サーバ，Blackbox exporter，Alertmanager が動作している．Blackbox exporter は，監

視対象の物理サーバ 6 台に対して ICMP による通信の疎通確認を行い，メトリクスデータを収集する．

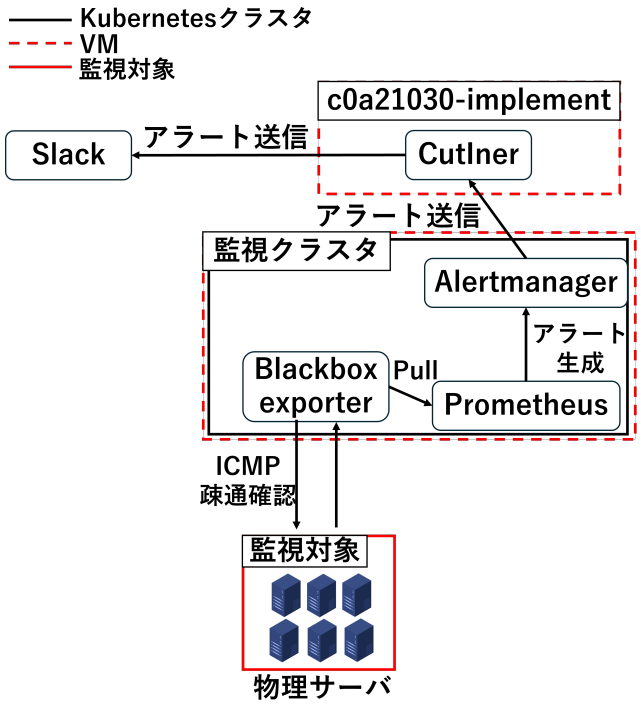

図 12: 実験環境

Prometheus サーバは Blackbox exporter からメトリクスデータを収集する．Blackbox exporter から収集した ICMP による通信の疎通確認のメトリクスデータである probe_success の値が 0 であった場合，Prometheus サーバはアラートを生成する．Alertmanager は，Prometheus サーバから受け取ったアラートを，提案ソフトウェアである CutIner へ送信する．CutIner は，出力結果である順位の高い順番に並び変えられたアラートを，Slack へと送信する．以下に監視クラスタの VM の構成を示す．

- 監視クラスタの VM の構成
  - OS：Ubuntu Linux 22.04(64 ビット)
  - vCPU：4[core]
  - RAM：4[GB]
  - SSD：40[GB]

次に，監視クラスタのクラスタ構成を示す．

- master のノード数
  - 3 台
- worker のノード数
  - 6 台

次に，提案ソフトウェアが入っている VM である c0a21030-implement の VM 構成を以下に示す．

- c0a21030-implement の VM 構成
  - OS：Ubuntu Linux 22.04(64 ビット)
  - vCPU：2[core]
  - RAM：2[GB]
  - SSD：25[GB]

実験に使用する値

変数 x[分]，upstream[分] の値は，以下のように設定した．

- x[分]：8
- upstream[分]：50

本稿の実験において，Prometheus のアラートルールのアラートを通知させるために PromQL 式が成立した状態が継続する期間を 3 分，Alertmanager の group_wait を 5 分としたため，x[分] に 8 を使用した．また，upstream[分] は 50 分とした．

実験結果と分析

通知されたアラート

図 13 に提案適用前の通知されたアラートを示す．192.168.100.27 は Lily の IP アドレスを示しており，この物理サーバのアラートは 6 番目に通知されている．次に，図 14 に提案適用後の通知されたアラートを示す．提案適用前の通知されたアラートと比較すると，アラートの通知順が変化している．提案適用前の通知されたアラートにおいて 6 番目に通知されていた Lily のアラートは，提案適用後の通知されたアラートにおいて 1 番目に通知されている．提案適用前の通知されたアラートにおいて 1 番目に通知されていた Plum のアラートは，提案適用後の通知されたアラートにおいて 2 番目に通知されている．提案適用前の通知されたアラートにおいて 2 番目に通知されていた Jasmine のアラートは，提案適用後の通知されたアラートにおいて 3 番目に通知されている．提案適用前の通知されたアラートにおいて 3 番目に通知されていた Rose のアラートは，提案適用後の通知されたアラートにおいて 4 番目に通知されている．提案適用前の通知されたアラートにおいて 4 番目に通知されていた Lotus のアラートは，提案適用後の通知されたアラートにおいて 5 番目に通知されている．提案適用前の通知されたアラートにおいて 5 番目に通知されていた Mint のアラートは，提案適用後の通知されたアラートにおいて 6 番目に通知されている．

指標との一致率

図 14 及び表 3 より，実際に通知されたアラートを，作成したアラートの通知順の指標と比較する．図 14 の 1 番目に通知されたアラートは Lily についてのアラートであり，表 3 で 1 番目に通知されるべきとしたアラートは Lily についてのアラートであるため，一致している．図 14 の 2 番目に通知されたアラートは Plum についてのアラートであり，表 3 で 2 番目に通知されるべきとしたアラートは Rose についてのアラートであるため，一致していない．図 14 の 3 番目に通知されたアラートは Jasmine についてのアラートであり，表 3 で 3 番目に通知されるべきとしたア

ラートは Mint についてのアラートであるため，一致していない．

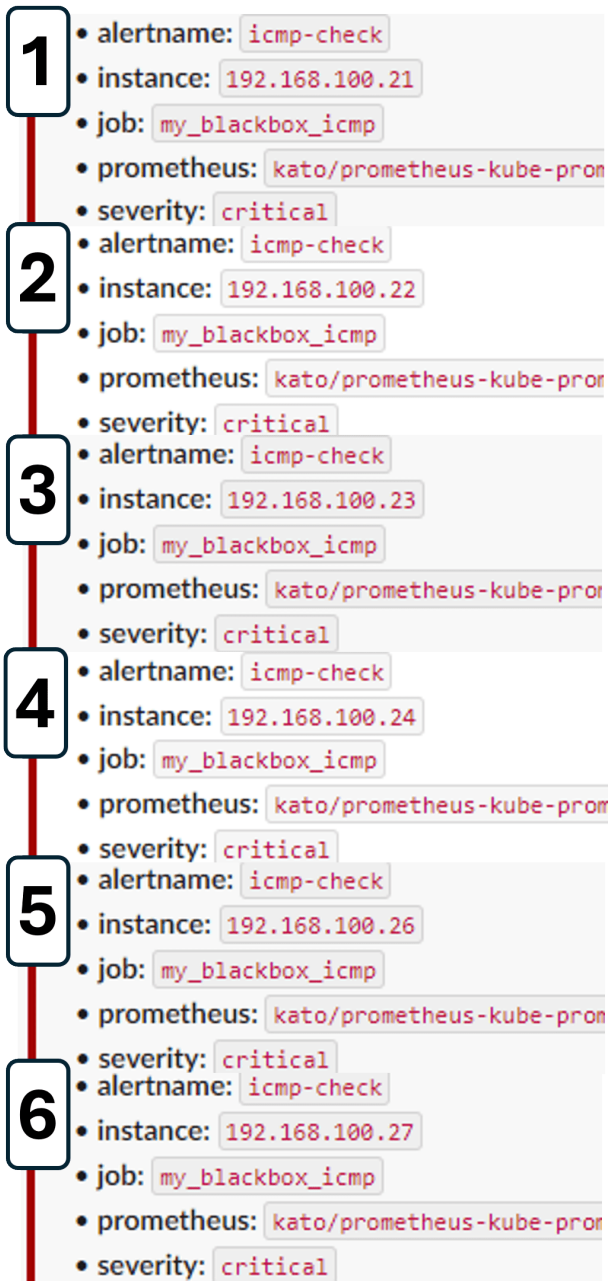

図 13: 提案適用前の通知されたアラート

図 14 の 4 番目に通知されたアラートは Rose についてのアラートであり，表 3 で 4 番目に通知されるべきとしたアラートは Jasmine についてのアラートであるため，一致していない．図 14 の 5 番目に通知されたアラートは Lotus についてのアラートであり，表 3 で 5 番目に通知されるべきとしたアラートは Lotus についてのアラートであるため，一致している．図 14 の 6 番目に通知されたアラートは Mint についてのアラートであり，表 3 で 6 番目に通知されるべきとしたアラートは Plum についてのアラートであるため，一致していない．アラートの通知順は，2 つ一致し，4 つ一致しなかった．そのため指標との一致率は，$2 \div 6 \times 100$ より約 33%となる．

図 14: 提案適用後の通知されたアラート

以上の結果から，提案ソフトウェアは 2 つのアラートを，作成したアラートの通知順の指標と同じ通知順にすること

ができたが，4 つのアラートを，作成したアラートの通知順の指標と同じ通知順にできなかったため，改善の余地がある．

6. 議論

提案では，アラートが通知された箇所において，upstream[分] の時間内に学生がサーバに SSH 接続した時間の割合から各物理サーバの直前接続度を算出することで，アラートに順位を設定した．upstream[分] の時間内にいずれのサーバにも学生からの SSH 接続が無かった場合，アラートの通知順は提案を使用していない場合と同じになる．これは，各サーバにおける学生からの SSH 接続のうち，障害検出時刻から，障害検出時刻から最も近い SSH 接続の開始時刻までの時間 (以下 next_upstream[分] と呼ぶ) を使用することで解決可能である．図 15 は next_upstream[分] の時間範囲を示している．

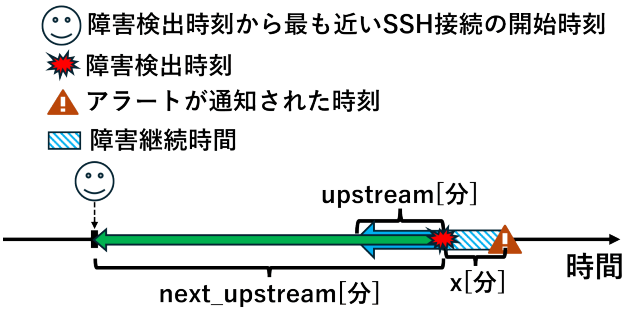

図 15: next_upstream[分] の時間範囲

障害検出時刻から upstream[分] の時間の範囲外に，サーバへの学生からの SSH 接続が開始された時刻が記録されている．そのため，障害検出時刻からその時刻までの時間範囲を next_upstream[分] の時間範囲としている．next_upstream[分] の値は，各物理サーバでそれぞれ算出する．そして，各物理サーバの next_upstream[分] の値を比較し，最小値のものを全物理サーバで使用することで upstream よりも以前の SSH 接続を直前接続度の算出に使用できる．そのため，直前接続度が 0.00 でなくなり，アラートに順位をつけることが可能になる．最小値を使用した理由として，障害検出時刻から近い SSH 接続であるほど，障害検出時刻の直前に SSH 接続をしていたと判断できるからである．

提案では，upstream[分] 内に学生がサーバに SSH 接続した時間を取得した．upstream[分] の値として，50 分を設定した．しかし，50 という値には根拠が必要である．解決策として，直前接続度の算出に使用する SSH 接続時間を使用せず，障害が原因で SSH 接続を終了させられた学生の人数を使用する方法がある．これにより，作業が中断された学生が使用していた VM がある物理サーバを，高い順位にすることができる．

提案では，物理サーバ内の VM ごとに直前接続度を算出し，その平均を物理サーバの直前接続度とした．一方で，物理サーバごとに VM 数が異なるため，VM 数の多い物理サーバでは SSH 接続時間が少ない VM が多いほど，物理サーバの直前接続度が小さくなる．これは，物理サーバ内の全 VM の直前接続度を平均しているため，小さい直前接続度が平均を下げるためである．解決策として，物理サーバの直前接続度の算出方法を，物理サーバごとに VM それぞれの直前接続度の平均する方法ではなく，VM それぞれの直前接続度を加算する方法に変更する．加算する理由として，物理サーバの直前接続度が，直前接続度が小さい VM の数によって左右されないことがある．これは，VM の直前接続度が小さい場合でも，大きい直前接続度の値を増減させないためである．また，VM それぞれの直前接続度を加算することで，直前接続度の値が小さい VM でも，物理サーバの直前接続度を増加させることができる．そのため，より多くの VM が使われている物理サーバであると判断して，この物理サーバについてのアラートを高い順位にすることができる．

提案では，アラート通知順が，評価実験にて定めたアラートの通知順の指標と同じにはならなかった．解決策として，物理サーバの直前接続度に，その物理サーバの全 VM 数を乗算する方法がある．例えば，直前接続度が 0.12 である Lily と，直前接続度が 0.21 である Mint の 2 つの物理サーバがあるとする．Lily の VM 数は 44 台であるため，$0.12 \times 44 = 5.28$ である．一方で，Mint の VM 数は 19 台であるため，$0.21 \times 19 = 3.99$ である．この 2 つの値を比較すると，Lily の方が値が大きい．これにより，さらに物理サーバに重みを付けることができるため，提案ソフトウェアのアラートの通知順を，よりアラートの通知順の指標に近づけることができる．

7. おわりに

課題は，グループ化されたアラートの severity が全て同じ値で設定されているため，オンコール担当の学生がどのアラートを解決するべきなのか判断ができないことである．そのため，解決されるべきアラートのエスカレーションが後回しになり，MTTR が増加する．提案は，物理サーバごとに，障害が発生する前に学生がサーバに SSH 接続した時間の割合を表す直前接続度を設定し，直前接続度の値が大きいものから順番に，順位をつける方式を提案する．直前接続度は，サーバへの SSH 接続時間が，障害発生時刻から遡る時間内のどのくらいの割合を占めるのかという VM ごとの値であり，VM ごとの直前接続度を平均したものが物理サーバごとの直前接続度となる．また，順位の高いものから並べたアラートを通知する．評価実験では，各物理サーバにおいて VM を所有している CDSL の 4 年生の人数をもとにアラートの通知順の指標を作成した．そして作

成したアラートの通知順の指標に対して，提案ソフトウェアから通知されるアラートの通知順が一致するかどうかを比較した．その後，いくつのアラートが通知順の指標と一致しているかを指標との一致率として表し，指標との一致率を評価する．指標との一致率は，グループ内の全アラートの件数のうち，いくつのアラートが指標と同じ通知順であったのかを，百分率で表す．実験内容は，物理サーバとスイッチングハブを接続する LAN ケーブルを引き抜くことで，ICMP による通信エラーのアラートを通知することである．結果は，6 つのアラートのうち，2 つのアラートの通知順が指標の通知順と一致したため，指標との一致率が約 33%であった．

謝辞 各物理サーバから使用状況を取得するツールを提供してくださった東京工科大学コンピュータサイエンス学部の山野倖平さんに御礼申し上げます．

参考文献

[1] Zhang, K., Xu, J., Min, M. R., Jiang, G., Pelechrinis, K. and Zhang, H.: Automated IT system failure prediction: A deep learning approach, *2016 IEEE International Conference on Big Data (Big Data)*, pp. 1291–1300 (online), DOI: 10.1109/BigData.2016.7840733 (2016).

[2] Robinson, W.: Monitoring Web service requirements, *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, pp. 65–74 (online), DOI: 10.1109/ICRE.2003.1232738 (2003).

[3] Quentin, L.: Scalability Evaluation of Prometheus for HPC Monitoring (2024).

[4] Saikkonen, J.: Creating centralized logging and monitoring to Zero Day Delivery project template (2020).

[5] Panaite, D. C.: Evaluating the performance of eBPF-based security software in a virtualized 5G cluster, PhD Thesis, Politecnico di Torino (2024).

[6] Sukhija, N., Bautista, E., James, O., Gens, D., Deng, S., Lam, Y., Quan, T. and Lalli, B.: Event management and monitoring framework for HPC environments using ServiceNow and Prometheus, *Proceedings of the 12th international conference on management of digital ecosystems*, pp. 149–156 (2020).

[7] Yuan, C., Zhang, W., Ma, T., Yue, M. and Wang, P.-P.: Design and implementation of accelerator control monitoring system, *Nuclear Science and Techniques*, Vol. 34, No. 4, p. 56 (2023).

[8] Klymash, M., Zablotskyi, S. and Pohranychnyi, V.: Improving Alerting in the Monitoring System Using Machine Learning Algorithms, *2024 IEEE 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, IEEE, pp. 150–153 (2024).

[9] Boncea, R. and Bacivarov, I.: A system architecture for monitoring the reliability of iot, *Proceedings of the 15th International Conference on Quality and Dependability*, pp. 143–150 (2016).

[10] Huang, H., Jennings III, R. B., Ruan, Y., Sahoo, R. K., Sahu, S. and Shaikh, A.: PDA: A Tool for Automated Problem Determination., *LISA*, Vol. 7, pp. 1–14 (2007).

[11] Sirviö, J.: Monitoring of a Cloud-Based IT Infrastructure (2021).

[12] Nakamura, T. and Kijima, K.: Total system intervention for system failures and its application to information and communication technology systems, *Systems Research and Behavioral Science*, Vol. 28, No. 5, pp. 553–566 (2011).

[13] Jagannathan, A., Dye, C. M., Rajamani, K., Galtenberg, C., Luong, B. and Ford, E.: REFORM: Increase alerts value using data driven approach, *2023 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 184–192 (online), DOI: 10.1109/IC2E59103.2023.00028 (2023).

[14] Ďorďević, B., Timčenko, V., Nikolić, E. and Davidović, N.: Comparing performances of native host and virtualization on ESXi hypervisor, *2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH)*, IEEE, pp. 1–4 (2021).

[15] Manik, V. K. and Arora, D.: Performance comparison of commercial vmm: Esxi, xen, hyper-v & kvm, *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, pp. 1771–1775 (2016).

[16] Tuhkanen, A.: Server Consolidation with VMware ESXi 3.5 (2010).

[17] Bassey, C., Idowu, S. and Ojo, C.: Alert Prioritization Techniques in Security Monitoring: A Focus on Severity Averaging and Alert Entities, *Saudi J Eng Technol*, Vol. 9, No. 7, pp. 334–339 (2024).

[18] Alsubhi, K., Al-Shaer, E. and Boutaba, R.: Alert prioritization in intrusion detection systems, *NOMS 2008-2008 IEEE Network Operations and Management Symposium*, IEEE, pp. 33–40 (2008).

[19] Bierma, M., Doak, J. D. J. E. and Hudson, C.: Learning to rank for alert triage, *2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pp. 1–5 (online), DOI: 10.1109/THS.2016.7568907 (2016).

[20] Liu, Y., Shu, X., Sun, Y., Jang, J. and Mittal, P.: RAPID: real-time alert investigation with context-aware prioritization for efficient threat discovery, *Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 827–840 (2022).