showmountの出力で識別したNFSサーバの表示によるPromQLでのディスク容量が不足した原因特定の時間短縮

加藤 健吾[1]　平尾 真斗[2]　串田 高幸[1]

概要：Network File System(以下 NFS と定義する) は，ネットワーク上で複数のコンピュータがファイルシステムを共有できる仕組みである．NFS サーバにおいてディスク容量が不足する障害が発生した時，2 次対応の担当エンジニアは PromQL を用いてディスク使用率のグラフを見る．課題は， PromQL のクエリ作成時に，ディスク容量が不足する障害の根本原因となる NFS サーバがどの IP アドレス及びホスト名であるかをドキュメントで見る必要があることである．これにより，ディスク容量が不足する障害の原因の特定にかかる時間が長くなってしまう．本稿の提案では，showmount コマンドの出力の差分の有無で NFS のマウント関係の変化を検知し，変化があった場合にクエリ対象の選択肢すべてに対してリモートで showmount コマンドを使用することで NFS サーバの特定を行う．そして，NFS サーバにおけるディスク容量不足のアラートが発生した際に，特定した NFS サーバの IP アドレス及びホスト名を Slack のメッセージへ出力する．基礎実験では，CDSL の学生 4 人にディスク使用率のグラフを表示する PromQL のクエリを用いて，NFS サーバを特定したうえでディスク使用率のグラフを見て，原因の特定を行ってもらった．PromQL のクエリ対象を選択し始めてから，どの IP アドレス及びホスト名に対して対処が必要であるかを Slack メッセージにて送信するまでの時間を計測した．結果は，平均約 303 秒であった．このことから，NFS サーバを特定し，ディスク使用率が不足していることを確認するのには時間がかかると分かった．評価実験では，提案を適用した状態で CDSL の学生 5 人にディスク使用率のグラフを表示する PromQL のクエリを用いて，NFS サーバのディスク使用率のグラフを見て，原因の特定を行ってもらう．PromQL のクエリ対象を選択し始めてから，ディスク容量が不足している NFS サーバを特定するまでにかかった時間を評価する．

1. はじめに

背景

Web サービスは，インターネット上で情報を共有・配布する手段として始まり，現在では電子商取引のアプリケーションとしても使用されている [1]．Web サービスをユーザに提供し続けるうえで，システム管理者は Web サービスの動作を要件と一致させる必要がある [2]．しかし，Web サービスの動作は，ハードウェアリソースや，ネットワーク帯域幅，クライアント要求の数によって遅くなることや，停止することがある [2]．このような影響が，システム障害として要件の達成を妨げることがある．ここでいう障害とは，要件の達成を妨げる環境条件である [3]．現在の業務やサービスの多くは IT システムを利用している [4]．そのため，システム障害によってサービスが利用できなくなり，ユーザが不便を感じると，企業への信頼喪失にもつながる[*1]．そこで，システム障害の発生によって Web サービスの動作が要件を満たせなくなった場合に，障害を迅速に把握し対応を行うために，システム管理者は監視を行う [3]．

システム障害が発生した場合，1 次対応の担当エンジニアはアラート通知を受け取る．次にエンジニアは，事前に定められた手順書に沿って，アラート内容の調査及び対処を実施する [5]．1 次対応での調査及び対処において，システム障害が復旧された場合，エンジニアは手順書で定められた事後対応を実施する[*2]．一方で，システム障害が復旧されなかった場合，エンジニアは 2 次対応の担当エンジニアへとエスカレーションを行う [6]．エスカレーションされた 2 次対応の担当エンジニアは，手順書外のより専門的な調査及び対処を行う[*3]．2 次対応の調査及び対処では，障

[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

[*1] https://www.pagerduty.co.jp/blog/what-is-system-error
[*2] https://www.knowledgewing.com/kcc/share/method/single46.html
[*3] https://www.rworks.jp/monitoring/monitoring-column/monitoring-design/25595/

害原因の切り分けを行い，原因の特定をする[*4]．そして，特定された原因に応じた対処を実行する．

障害の原因特定を行うエンジニアは，ログ，時系列データ，トレースを用いて原因を切り分ける [7]．時系列データを得るために，エンジニアは Domain Specific Language クエリ (以下 DSL クエリと定義する) を作成し，監視ツールに対してテレメトリーデータの抽出を行う [7]．DSL クエリの例として，Prometheus が提供している PromQL がある [8]．PromQL は，メトリクスの抽出のほか，アラート生成の条件としても使用される．

監視ツールの 1 つに，Prometheus がある [9]．Prometheus は pull 型のアプローチ方法を採用しており，exporter を通して，メトリクスデータを収集する [10]．収集されるデータは exporter の種類によって異なる．例に Blackbox exporter と Node exporter をあげる．Blackbox exporter は主に監視クラスタ内に構築し，指定された対象ノードに対して外形監視を行う [11]．このようにして収集されたデータは，Prometheus 内に入ることで利用することができる [12]．収集されたデータのうち，特定のデータを利用したい場合，そのデータをクエリで指定して表示することができる [13]．

Network File System(以下 NFS と定義する) は，ネットワーク上のあるコンピュータのディスクを，別のコンピュータでも共有して使うことができる仕組みである [14]．この仕組みにより，ユーザは別のコンピュータのファイルシステムを，自身のコンピュータ上のファイルシステムであるかのように利用できる．

NFS では，NFS サーバと NFS クライアントが存在する．NFS サーバはリモートシステムとしてリモートファイルを提供し，NFS クライアントはローカルシステムとして，NFS サーバが提供するリモートファイルにアクセスする [15]．NFS クライアントがリモートファイルにアクセスする時，NFS RPC 要求が NFS サーバに送信され，NFS サーバはこれに応答を返す．また，NFS サーバは，複数の NFS クライアントに対してリモートファイルを提供することができる [16]．

NFS サーバ及び NFS クライアントにおけるディスク容量不足の障害が発生した際，2 次対応を行うエンジニアが原因特定のために Prometheus ダッシュボード上で PromQL のクエリを用いてディスク使用率を確認する．

課題

課題は，障害の原因特定にかかる時間が長くなってしまうことである．課題の原因は，PromQL のクエリ作成時に，障害の根本原因となる NFS サーバがどの IP アドレス及びホスト名であるかをドキュメントで見る時間が必要があることである．ディスク容量不足のアラートは，NFS サーバ 1 台，NFS クライアント 5 台の計 6 台で発生している．課題の状況を図 1 に示す．まず，NFS サーバ及び

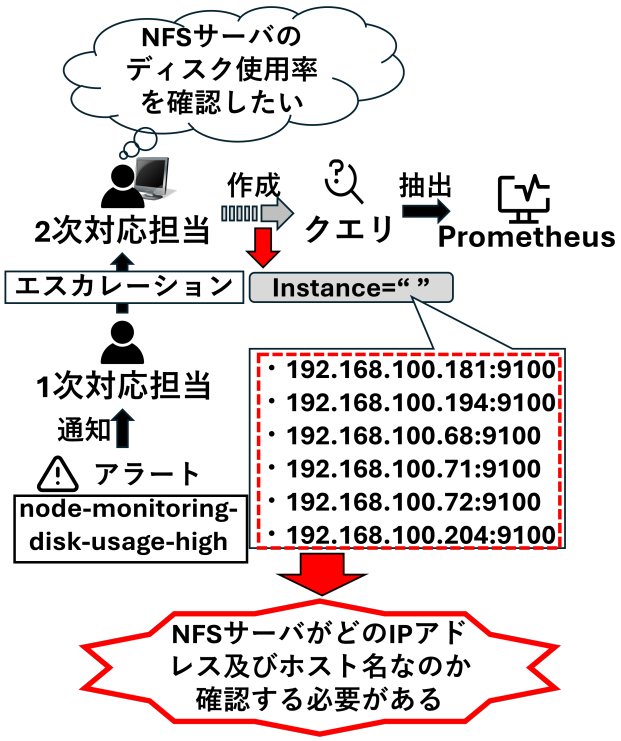

図 1: 課題の状況

NFS クライアントにおいてディスク容量が不足しているアラートが発生する．アラートの通知を受けた 1 次対応の担当エンジニアは，2 次対応の担当エンジニアへとエスカレーションを行う．2 次対応の担当エンジニアは，ディスク使用率の確認を行うため，Prometheus ダッシュボード上で PromQL のクエリを作成し，メトリクスデータの抽出を行う．しかし，PromQL のクエリの作成段階でクエリ対象である instance を設定する際に，どの IP アドレス及びホスト名が NFS サーバであるのかを確認する必要がある．

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，本稿で挙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

回帰分析を使用し，システムの負荷を測定する能力に基づいてメトリクスをランク付けすることで，システムの包括的なパフォーマンスモニタリングに必要なデータが何であるかを特定した研究がある [17]．この研究では，パ

[*4] https://www.geekly.co.jp/column/cat-technology/1903_035/

フォーマンステストベンチツールを使用し，サービス指向システムを実行して得られたデータの分析を行うことで，Zookeeper コンポーネントに関連しているものや，ネットワークトラフィック，メモリ，ディスク容量のメトリクスが重要であると示した．この研究では，システムで重要なメトリクスを示したが，どのノードにおける重要なメトリクスであるのかを示していない．

ログの解析と追跡やログのメトリクスへのマッピング，最も関連性の高いリソースメトリクスの特定により，異常検出のためのアサーション仕様を導出することで，システムのアプリケーション操作の信頼性を向上させるための監視及び異常検出技術を調査した研究がある [18]．この研究の提案アプローチは，様々な業界でのケーススタディで評価された結果，ログとメトリクスの分析を使用することで異常検出を効果的に行うことができると示した．しかしこの研究では，複数のノードで同時にアラートが発生した場合の異常検出においては，どのノードのメトリクスを確認するべきか分からない．

ネットワーク制御とエッジコンピューティングのアプローチを活用することで，工場全体の生産設備の状態監視を強化する，対話型ネットワーク状態監視戦略を提案した研究がある [19]．対話型ネットワーク状態監視戦略は，ドラッグアンドドロップによる監視アルゴリズム設計や，パラメータ及び結果を視覚的に表示する監視プロセスを提供した．アルミニウム製造工場での適用事例では，9 つの作業場の数百の生産設備において，手動では半日から数日かかるところを，数分から数十分に短縮した．この研究では，監視アルゴリズムの設計や監視プロセス，パラメータ及び結果を視覚的に表示できるようにしたが，どのノードの結果を表示すればいいか分からない状況が発生する．

DevOps コミュニティの最新世代のソフトウェアスタックに基づいて，ワークロード管理システムに統合された HPC に特化した分散型監視システムにより，エラーに対するユーザ問い合わせの応答時間を短縮した研究がある [20]．この監視システムでは，ダッシュボードにてメトリクスデータ上にシステムやノードレベルのイベントをオーバーレイ表示し，その情報をクエリに直接利用できるため，データ統計分析や集計が可能になっている．この研究では，使いやすい監視インターフェースを実現しているが，エンジニアがどのノードを確認すればいいかわからないという状況に対して，効果を発揮できない．

3. 提案

本稿では，2 次対応の担当エンジニアが，Prometheus ダッシュボード上で PromQL を用いてディスク使用率の確認を行う際に，どの IP アドレスおよびホスト名が NFS サーバを示すのか分かるようにすることで，原因特定までの時間を短縮することを目的とした．この目的を達成するため，showmount コマンドの出力の差分の有無で NFS のマウント関係の変化を検知し，変化があった場合にクエリ対象の選択肢すべてに対してリモートで showmount コマンドを使用することで NFS サーバの IP アドレス及びホスト名を特定する手法を提案する．提案の動作タイミングは，提案ソフトウェア導入時と，NFS のマウント関係が変化した時の 2 パターンある．図 2 は，NFS サーバ特定の手順を示したものである．提案ソフトウェアは PromQL の

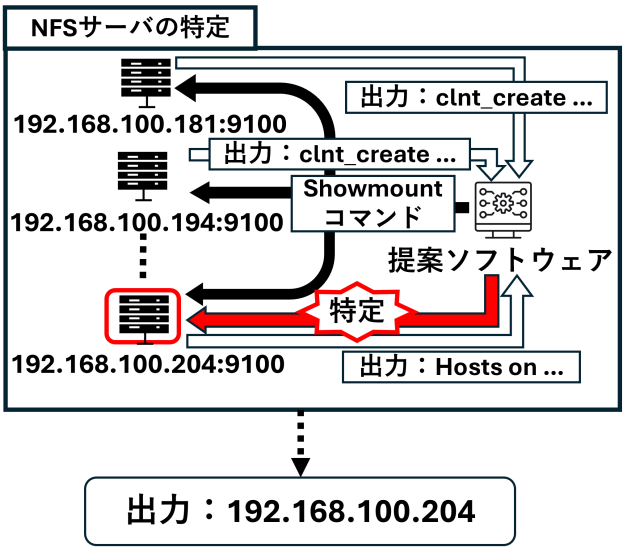

図 2: NFS サーバの特定手順

クエリ対象のすべての IP アドレス及びホスト名に対して showmount コマンドを実行し，「Hosts on」から始まる出力を出した IP アドレス及びホスト名を NFS サーバであると判断する．その後，NFS サーバの IP アドレス及びホスト名をリストとして出力する．図 3 は，提案ソフトウェアの 2 パターンの動作タイミングを示したものである．提案

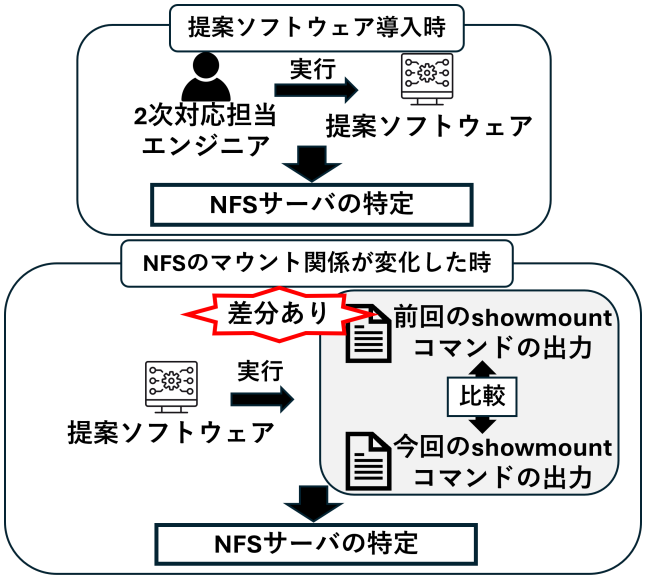

図 3: 提案ソフトウェアの動作タイミング

ソフトウェアは，導入時と NFS のマウント関係が変化した時に動作する．導入時というのは，2 次対応担当エンジニアが提案ソフトウェアを実行した時である．NFS のマウント関係が変化した時というのは，提案ソフトウェアが NFS サーバに対して showmount コマンドを実行した出力と，前回実行した showmount コマンドの出力を比較した際に，差分が存在した時である．これらの状況を満たすときに，NFS サーバの特定を開始する．

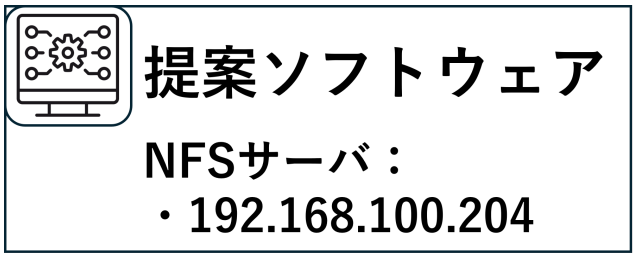

図 4: Slack メッセージを用いた出力の例

提案方式

提案方式のアルゴリズムは以下のようになっている．

3.1 NFS サーバの特定

選択肢に表示されている IP アドレス及びホスト名すべてに対して，ssh コマンドを用いて遠隔で showmount コマンドを実行する．実行するコマンドの例をコマンド 1 に示す．

コマンド 1: 実行するコマンド

```
1 $ ssh -i ~/.ssh/id_ed25519 cdsl@192
      .168.100.204 showmount
```

このコマンドでは，-i オプションで秘密鍵を指定し，cdsl@192.168.100.204 でユーザ名とコマンドを実行する対象を指定している．また，showmount は，対象に対して実行するコマンドの内容を示している．showmount コマンドを実行した対象が NFS サーバであれば，「Hosts on <コマンドを実行する対象の IP アドレス及びホスト名>:」から始まる出力結果が返される．NFS クライアントであれば，「clnt_create: RPC: Program not registered」という出力結果が返される．このため，提案ソフトウェアは showmount コマンドを実行した出力結果として，「Hosts on」から始まる出力をした IP アドレス及びホスト名を NFS サーバであると判断する．

3.2 出力

提案ソフトウェアは，特定した NFS サーバの IP アドレス及びホスト名を出力する．NFS サーバが複数ある場合があるため，リストの形式で出力される．出力のタイミングは，NFS サーバのディスク容量不足のアラートが発生した時とする．出力の方法は，Slack のメッセージを用いる．出力の例を図 4 に示す．提案ソフトウェアが「NFS サーバ：」以下の行に，NFS サーバの IP アドレス及びホスト名のリストをメッセージとして送信する．

3.3 マウント関係変化の検知

初回実行時以降に NFS のマウント関係の変化を検知した場合，再度提案ソフトウェアを動作して NFS サーバの特定を行う．検知は次のような方法で行う．提案ソフトウェアは，前回の提案ソフトウェア実行時に特定した NFS サーバに対して showmount コマンドを実行する．そして，得られた出力結果から NFS クライアントの IP アドレス及びホスト名の情報を取得する．取得した NFS クライアントの IP アドレス及びホスト名の情報を，前回取得したものと比較する．その結果，新しく IP アドレス及びホスト名が増えている，あるいは減っている場合，NFS のマウント関係が変化したと判断する．これは，NFS クライアントが新しくマウントもしくはアンマウントした場合，showmount コマンドの出力に表示される IP アドレス及びホスト名が増減するためである．showmount コマンドは 3 年間隔で実行する．これは，NFS サーバが配置されているハードウェアを 3 年から 5 年で新しくすることが望ましいからである [21]．

ユースケース・シナリオ

本稿では，ユースケースシナリオとして画像の保存や編集を行う画像プリントサイトにおいて，画像を保存している NFS でディスク容量が不足している障害が発生した状況を想定する．この状況における提案ソフトウェアが適用される箇所を図 5 に示す．まず，画像プリントサイトにおいてディスク容量不足の障害が発生する．この障害を Prometheus は検出し，アラートを作成する．また，Prometheus はアラートマネージャに対してアラートの通知を要求する．アラートマネージャはディスク容量不足のアラートを，1 次対応担当エンジニアのエンジニア 1 に対して通知する．エンジニア 1 が障害を解決できなかった場合，2 次対応担当エンジニアのエンジニア 2 に対してエスカレーションを行う．エンジニア 2 は，2 次対応として画像プリントサイトに対して，PromQL のクエリを用いて Prometheus からディスク使用率のメトリクスを抽出，取得する．提案ソフトウェアは，エンジニア 2 がディスク使用率の PromQL のクエリを作成する部分を対象として実行される．エンジニア 2 は，取得したメトリクス情報から原因となる IP アドレス及びホスト名，原因の種類を判断し，NFS サーバの容量を増加したり，データの削除を行うことで，障害対処を実行する．

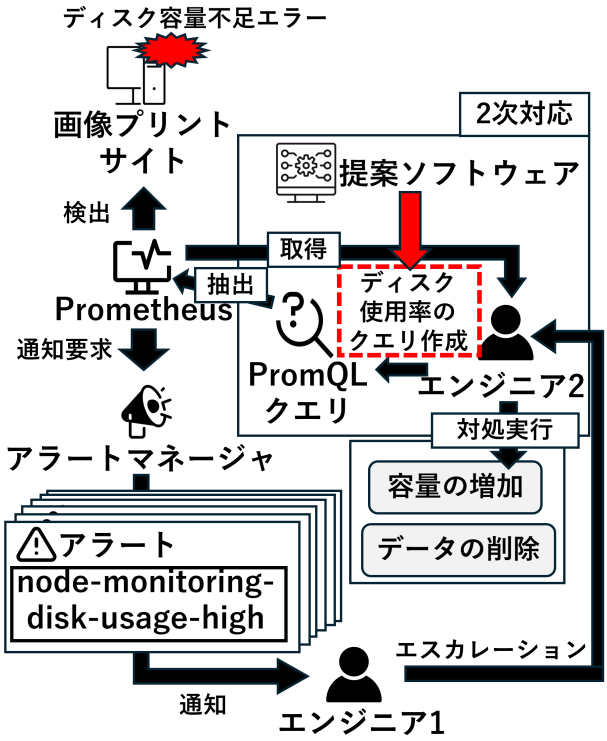

図 5: 提案ソフトウェアが適用される箇所

4. 実装

本提案の実装として，Truthight というソフトウェアを作成する．Truthight の NFS サーバ特定の処理の流れを図 6 に示す．Truthight の NFS サーバ特定の処理の流れは，

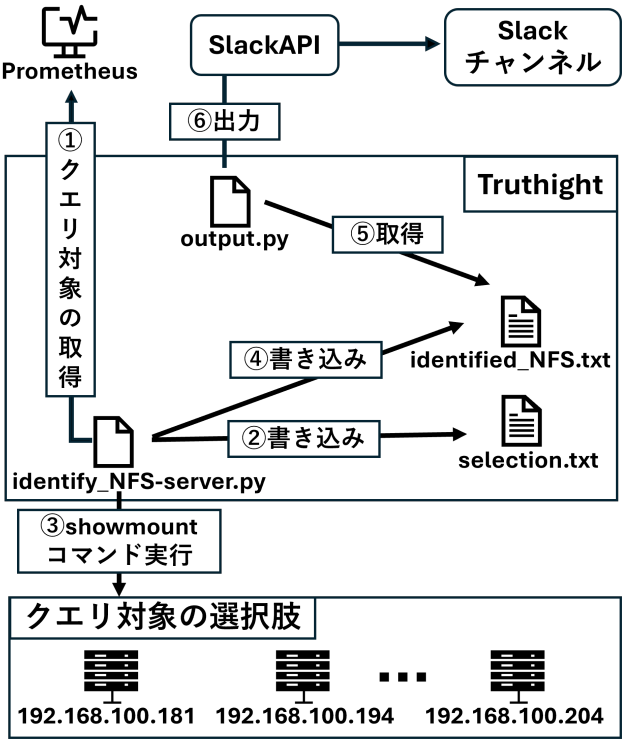

図 6: Truthight の NFS サーバ特定の処理

次のようになっている．

① identify_NFS-server.py が Prometheus から PromQL のクエリ対象の IP アドレス及びホスト名を取得する．

② identify_NFS-server.py が①で取得した内容を selection.txt に書き込む．

③ identify_NFS-server.py が PromQL のクエリ対象の選択肢のすべての IP アドレス及びホスト名に対して showmount コマンドを実行し，出力結果から NFS サーバの特定を行う．

④ identify_NFS-server.py が③で特定した NFS サーバの IP アドレス及びホスト名を identified_NFS.txt に書き込む．

⑤ ディスク容量不足のアラートが発生した場合，output.py が identified_NFS.txt から NFS サーバの IP アドレス及びホスト名を取得する．

⑥ output.py が SlackAPI を用いて Slack チャンネルへ NFS サーバの IP アドレス及びホスト名を出力する．

次に，Truthight の NFS マウント関係の変化を検知する処理の流れを図 7 に示す．Truthight の NFS マウント関係の

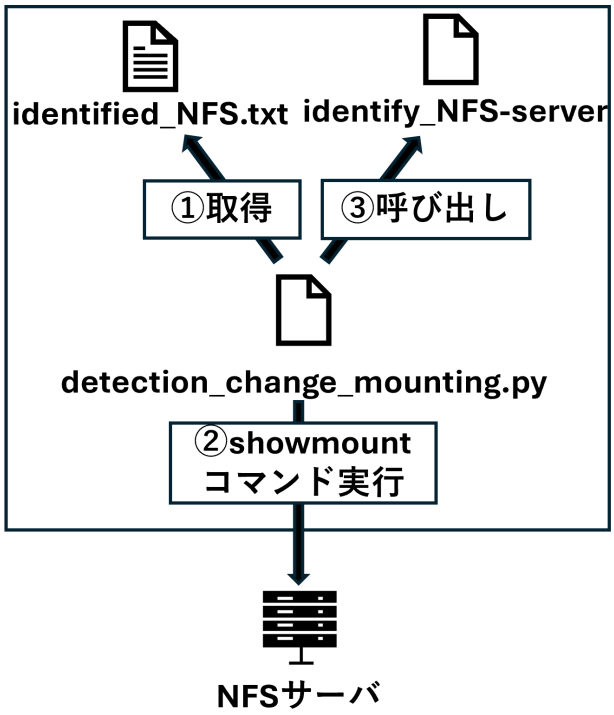

図 7: Truthight の NFS マウント関係の検知の処理

検知の処理の流れは次のようになっている．

① detection_change_mounting.py が identified_NFS.txt から，NFS サーバの IP アドレス及びホスト名を取得する．

② detection_change_mounting.py が①で取得した IP アドレス及びホスト名に対して showmount コマンドを実行する．

③ detection_change_mounting.py が前回の showmount コマンド実行結果と今回のものを比較し，差分があった場合，identify_NFS-server を呼び出す．

5. 評価実験

基礎実験

課題を示すために基礎実験を行う．CDSL の学生 4 人に，Prometheus のダッシュボードを使い，NFS サーバのディスク使用率を PromQL のクエリを用いて表示し，どの IP アドレス及びホスト名に対してディスク容量の増加やディスク内のデータの削除が必要であるかを判断してもらった．そして，NFS サーバの IP アドレス及びホスト名と，対処が必要であるということを Slack で送信してもらった．この一連の原因特定にかかった時間を計測した．ディスク使用率のクエリについては，対象ノードの設定の部分だけ行ってもらい，それ以外の部分については，事前に用意しておく．事前に用意した PromQL のクエリをコマンド 2 に示す．

コマンド 2: 用意した PromQL のクエリ

```
1 - (node_filesystem_avail_bytes{instance=""}
    / node_filesystem_size_bytes{instance=""
    })
```

この PromQL のクエリは，instance で選択した IP アドレス及びホスト名に対して，未使用のディスク領域を総ディスク領域で割ったものを，1 から減算することで，ディスク使用率を表している．また，図 8 に PromQL のクエリ対象の選択肢を示す．被験者はこの各 VM の IP アドレス及び

図 8: PromQL のクエリ対象の選択肢

ホスト名から，NFS サーバを表すものを特定し，instance の値に設定することで，NFS サーバに対してのディスク使用率のグラフを表示するクエリを作成する．PromQL のクエリ対象の選択肢から NFS サーバを選択する際，NFS サーバかどうかの判断は次の手順で行った．

① ssh cdsl@<IP アドレス及びホスト名>で対象 Virtual Machine(以下 VM と定義する) にアクセスする．

② showmount コマンドを実行する．

③ 実行結果が「Hosts on <コマンドを実行した IP アドレス及びホスト名>:」から始まるものであれば NFS サーバであると判断する．

ディスク容量の増加やディスク内のデータの削除が必要であるかどうかの判断は，NFS サーバに対してのディスク使用率を確認する PromQL のクエリによって表示されたグラフを見て，80%という閾値を超えていることが確認できた場合に必要であるとしてもらった．また，ディスク容量の増加やディスク内のデータの削除が必要な NFS サーバの IP アドレス及びホスト名と，ディスク容量の増加やディスク内のデータの削除が必要であるという内容を Slack にて送信してもらった．被験者に見てもらった Prometheus ダッシュボード上のグラフを図 9 に示す．緑の線が NFS

図 9: Prometheus ダッシュボード上のグラフ表示

サーバのディスク使用率を表している．表示されている部分の縦軸の値は，すべて 0.85 となっており，ディスク使用率が 80%を超えていることが確認できる．

時間の計測については，被験者が Prometheus ダッシュボードを開いたうえで，コマンド 2 をクエリ検索欄に記述されている状態から計測を開始し，被験者から Slack でメッセージが送信されたタイミングで計測を終了した．

NFS サーバ及び NFS クライアントのディスク使用率が不足するというアラートが発生する障害シナリオとして，Wordpress で構築された画像のプリントサイトを取り挙げたのサイトに対して画像データを送信し続け，NFS サーバの容量不足を引き起こした．

画像データ送信のシナリオには，カメラのキタムラの画像データの事例を参考にした．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 万件の画像データの処理依頼を請ける．これは，1 分間あたりに約 13 件のアクセスの画像データを保存するということである．この実験では，1 件あたりに送信される画像データの枚数を，画像編集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．iPhone の場合，画像データ 1 枚あたりの画像データの容量が平均 2.5MB であるため，1 件につき 50MB の画像データを送信する．この実験では，NFS サーバ内の VM ディスク容量が 50GB であり，NFS サーバ内の VM

のディスク使用率が 80%を超えた際にアラートを出すように設定している．50GB の 80%は 40GB であり，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，ディスク使用率が 80%を超えるためには 33.8GB 分の画像データを送る必要がある．1 分間に 13 人のユーザが画像を 20 枚ずつ送ると 650MB となるため，約 52 分間画像データを送り続けることで，ディスク使用率が 80%を超える．

実験結果を表 1 に示す．NFS サーバの IP アドレス及びホスト名を特定し，PromQL のクエリを用いてディスク使用率をダッシュボード上に表示させることで原因を特定するまでにかかった時間は，学生 1 が約 295 秒，学生 2 が約 204 秒，学生 3 が約 432 秒，学生 4 が約 281 秒であった．平均は約 303 秒であった．

表 1: 基礎実験結果

被験者	学生 1	学生 2	学生 3	学生 4
原因箇所の特定時間 (秒)	295	204	432	281

実験環境

実験に用いる VM の構成は次のようになっている．

- OS：Ubuntu22.04
- vCPU：4[core]
- RAM：4[GB]
- SSD：25[GB]

図 10 に実験環境の概要を示す．画像データを送信するユーザを想定し，Locust を用いて Wordpress に対して画像データを送信する．Wordpress は MySQL をデータベース管理システムとして用いており，MySQL は NFS クライアントの/mnt/wordpress ディレクトリにデータを保存する．また，NFS クライアントは/mnt/wordpress ディレクトリを，NFS サーバの/wordpress/mysql ディレクトリにマウントしている．監視クラスタには，Prometheus と AlertManger を配置する．Prometheus は，Wordpress のクラスタと NFS サーバの VM 内に exporter を配置することでメトリクスの収集を行う．

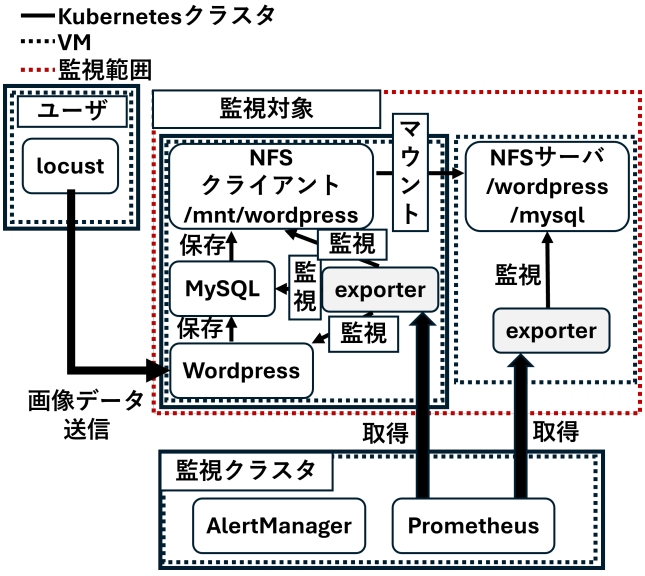

図 10: 実験環境の概要

以下に各 Kubernetes クラスタの構成を示す．

- ユーザのアクセスを想定したクラスタ
  - マスタ：1 台
  - ワーカ：2 台
- 監視クラスタ
  - マスタ：1 台
  - ワーカ：1 台
- Wordpress のクラスタ
  - マスタ：1 台
  - ワーカ：5 台

また，NFS は次のような構成になっており，合計 6 台である．

- NFS サーバ
  - outside-nfs4(IP アドレス：192.168.100.204)
- NFS クライアント
  - out-prometheus-w2(IP アドレス：192.168.100.181)
  - out-prometheus-w3(IP アドレス：192.168.100.194)
  - outside-prometheus-m(IP アドレス：192.168.100.68)
  - outside-prometheus-w4(IP アドレス：192.168.100.71)
  - outside-prometheus-w5(IP アドレス：192.168.100.72)

実験計画

評価実験では CDSL の学生 5 人に，基礎実験と同様に Prometheus のダッシュボードを使い，NFS サーバのディスク使用率のグラフを PromQL のクエリを用いて表示し，どの IP アドレス及びホスト名に対してディスク容量の増加やディスク内のデータの削除が必要であるかをクエリ結果のグラフが 0.8 を超えていることを確認することで判断してもらう．そして，NFS サーバの IP アドレス及びホスト名と，対処が必要であるということを Slack で送信してもらう．この一連の原因特定にかかった時間を計測する．ディスク使用率のクエリについては，対象ノードの設定の部分だけ行ってもらい，それ以外の部分については，事前に用意しておく．これを，提案ソフトウェア適用前と適用後で行い，かかった時間を比較し，どれだけ時間を短縮することができたのかを評価する．

6. 議論

本稿では，NFS のマウント関係の変化を検知するアルゴリズムにおいて，showmount コマンドを実行する間隔を 3 年とした．しかし 3 年ごとにする場合，最長で約 3 年間

NFS マウント関係の変化が検知できない．解決方法として，各 VM に cloud-init を用いて mount コマンドまたは umount コマンドを検知するプログラムを配置しておくことで，検知した場合に提案ソフトウェアに対して変化したことを伝達するというものがある．

冗長なアラートが発生しやすい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 で増加していた．WordPress はコンテンツの読み書きを MySQL に対して行う．つまり MySQL がダウンしてしまうと，WordPress で読み書きできなくなる．そのため WordPress よりも MySQL の Pod を PromQL のクエリの対象として優先する．WordPress と MySQL の関係を取得する方法として，サービスメッシュである Istio を用いて，WordPress と MySQL の通信の経路を取得する．MySQL はデータベースのため通信が集中する．アクセスユーザから MySQL までのアクセス経路のエッジの数を重みとして MySQL に設けることで，PromQL のクエリ対象として MySQL を優先させることができる．

7. おわりに

課題は， PromQL のクエリ作成時に，根本原因となる NFS サーバがどの IP アドレス及びホスト名であるかをドキュメントで見る必要があることである．提案は，showmount コマンドの出力の差分の有無で NFS のマウント関係の変化を検知し，変化があった場合にクエリ対象の選択肢すべてに対してリモートで showmount コマンドを使用することで NFS サーバの特定を行う．そして，NFS サーバにおけるディスク容量不足のアラートが発生した際に，特定した NFS サーバの IP アドレス及びホスト名を Slack のメッセージとして出力する．基礎実験では，CDSL の学生 4 人にディスク使用率のグラフを表示する PromQL のクエリを用いて，NFS サーバを特定したうえでディスク使用率のグラフを見て，原因の特定を行ってもらった．そして PromQL のクエリ対象を選択し始めてから，どの IP アドレス及びホスト名に対して対処が必要であるかを Slack メッセージにて送信するまでの時間を計測した．結果は，平均 303 秒であった．このことから，NFS サーバを特定し，ディスク使用率が不足していることを確認するのには時間がかかると分かった．評価実験では，提案を適用した状態で CDSL の学生 5 人にディスク使用率のグラフを表示する PromQL のクエリを用いて，NFS サーバのディスク使用率のグラフを見て，原因の特定を行ってもらう．PromQL のクエリ対象を選択し始めてから，ディスク容量が不足している NFS サーバを特定するまでにかかった時間を評価する．

参考文献

[1] Curbera, F., Nagy, W. and Weerawarana, S.: Web services: Why and how, *Workshop on Object-Oriented Web Services-OOPSLA*, Vol. 2001, Citeseer (2001).

[2] Wang, Q., Shao, J., Deng, F., Liu, Y., Li, M., Han, J. and Mei, H.: An Online Monitoring Approach for Web Service Requirements, *IEEE Transactions on Services Computing*, Vol. 2, No. 4, pp. 338–351 (online), DOI: 10.1109/TSC.2009.22 (2009).

[3] 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).

[4] 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).

[5] Gamini, H. K., Paidi, N., Seelamanthula, S. K., Batchu, P. and Burgal, F. A.: GC-155-Runbook Automation (2022).

[6] 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).

[7] Jiang, Y., Zhang, C., He, S., Yang, Z., Ma, M., Qin, S., Kang, Y., Dang, Y., Rajmohan, S., Lin, Q. et al.: Xpert: Empowering incident management with query recommendations via large language models, *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13 (2024).

[8] 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).

[9] Chen, L., Xian, M. and Liu, J.: Monitoring system of openstack cloud platform based on prometheus, *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, IEEE, pp. 206–209 (2020).

[10] Hellstern, T.: CERNBox Monitoring with Pushgateways, Technical report (2002).

[11] Angenent, H., Müller, D. and Vogl, R.: Bringing HPC clusters into Science Mesh, *Proc. Eur. Univ. 95*, pp. 249–256 (2023).

[12] Giannopoulos, D., Papaioannou, P., Ntzogani, L., Tranoris, C. and Denazis, S.: A holistic approach for 5g network slice monitoring, *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, IEEE, pp. 240–245 (2021).

[13] Li, J. et al.: Orchestration Mechanism Impact on Virtual Network Function Throughput, Master's thesis (2019).

[14] Pawlowski, B.: NFS, Its Applications and Future., *LISA* (2004).

[15] Sahrawat, A., Trivedi, V., Jeon, N., Kwon, J. and Hahm, C.-h.: On the improvement of file browsing time in single client NFS system, *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, IEEE, pp. 371–375 (2013).

[16] Banks, G.: Making The Linux NFS Server Suck Faster, *linux. conf. au* (2007).

[17] Lomio, F., Jurvansuu, S. and Taibi, D.: Metrics selection for load monitoring of service-oriented system, *Proceedings of the 5th International Workshop on Machine Learning Techniques for Software Quality Evolution*,

pp. 31–36 (2021).
[18] Farshchi, M.: Anomaly detection using logs and metrics analysis for system application operations (2018).
[19] Xiao, H., Zhou, H., Hu, W. and Liu, G.-P.: Design and implementation of an interactive networked condition monitoring strategy for plant-wide production equipment toward Industry 4.0, *Expert Systems with Applications*, p. 124376 (2024).
[20] Birngruber, E., Forai, P. and Zauner, A.: Total recall: holistic metrics for broad systems performance and user experience visibility in a data-intensive computing environment, *Proceedings of the Second International Workshop on HPC User Support Tools*, pp. 1–12 (2015).
[21] Matsuzawa, K., Hayasaka, M. and Shinagawa, T.: The quick migration of file servers, *Proceedings of the 11th ACM International Systems and Storage Conference*, pp. 65–75 (2018).

