メトリクス名と入力した文字列が一致した候補の表示による Prometheus のアラートの設定に要する時間の削減

有田 海斗[1]　三上 翔太[1]　串田 高幸[1]

概要：物理マシンでは，データストア容量が不足し，新たに仮想マシンを作成できないことがある．管理者はアラートにより，アラートがないときと比べて，早期にデータストア容量の不足を知ることができる．そのため，仮想マシンを作成できなくなる前に未然に防ぐことができる．アラートを生成するソフトウェアとして Prometheus がある．Prometheus では，管理者が YAML 形式の設定ファイルにアラートの設定を行うことで，通知が行われる．課題は，アラートの式設定を行う際，複数のメトリクス名の変数を組み合わせることがあるため，設定ファイルに入力する文字数が多くなり，設定に時間を要することである．基礎実験では，4 人の学生を対象に，VMware 社の Elastic Sky X Integrated のデータストア使用率が 80.0%を超えたときにアラートを出す式設定を行った．データストア使用率のアラートの式設定は，vmware_datastore_capacity_size と_datastore_freespace_size の 2 種類のメトリクス名が組み合わされたものである．基礎実験でアラートの式設定に要する 4 人の平均時間は，約 245 秒であった．提案では，アラートの式設定を行う際に，メトリクス名の候補を表示し，その候補の中から選択したメトリクス名を出力することで，入力する文字数を削減する手法を提案する．監視サーバのメトリクス名を全て取得し，Prometheus のメトリクス名と管理者が入力している文字列が前方一致していたら候補を表示する．評価実験では，4 人の学生を対象に，データストア使用率が 80.0%を超えたときにアラートを出す式設定に要する時間を測定した．評価実験では，提案適用前の 4 人の平均時間約 245 秒に対し，適用後は約 137 秒で，約 44.1%の時間を削減することができた．

1. はじめに

背景

東京工科大学コンピュータサイエンス学部に属する研究室の Cloud and Distributed Systems Laboratory(以後 CDSL とする) では，7 台の物理マシン (以後 PM とする) がある．PM とは，ハードウェアであり，ソフトウェアや仮想化技術が動作する基盤となる物理的なコンピュータである．7 台の PM のうち，6 台は学生が自由に Virtual Machine(以後 VM とする) の作成をすることができる．

VM とは，仮想化技術により作成された仮想ハードウェア環境であり，コンピュータアーキテクチャに新しいレイヤーを導入するものである [1]．VM は，PM 内のリソースを分割することで，それぞれ独立したマシンとして動作する．リソースの例に，CPU，ストレージ，メモリがあげられる．VM は 1 つの PM で複数作成することができる [2–4]．学生は，VMware 社のハイパーバイザーである Elastic Sky X Integrated(以後 ESXi とする) 上に VM を作成し，それぞれ所有している Personal Computer から VM の OS に SSH で接続し，実験を行っている．

PM では，データストア容量が不足し，新たに VM を作成できないことがある．管理者は，データストアに対してアラートの設定を行うことで，アラートがないときと比べて，早期にデータストア容量の不足を知ることができる．そのため，VM を作成できなくなる前に未然に防ぐことができる．例として，2024 年 12 月 12 日に，CDSL で使用している PM のひとつである Jasmine のデータストア使用率が，全体容量の 2.93TB のうち 2.8TB が割り当てられており，約 96.0%であった．VM は PM のデータストアを使って作成するため，データストア使用率が 100.0%になった際，データストア容量を新たに割り当てられないため新しい VM を作成できない．そこで，管理者はデータストアに対してアラートの設定を行うことで，アラートがないときと比べて，早期にデータストア容量の不足を知ることができる．データストア容量が不足しているため，データストア容量を追加することで，VM を作成できなくなる前に未然に防ぐことができる．

アラートを生成するソフトウェアとして Prometheus が

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1

ある．Prometheus は，監視およびアラート機能を使用できるようにするオープンソースの監視ソフトウェアである [5–7].

CDSL では，Jasmine を監視している Prometheus の監視サーバがある．Prometheus でデータストアに対してアラートの設定を行うことで，Slack に通知を送ることができる．CDSL で設定したアラートの式設定を設定ファイル 1 に示す.

設定ファイル 1: CDSL で設定したアラートの式設定

```
expr: (vmware_datastore_capacity_size{ds_name
    !="StoreNAS-Public",instance!="arita-
    master:32704"} -
    vmware_datastore_freespace_size{ds_name!="
    StoreNAS-Public",instance!="arita-master
    :32704"}) / vmware_datastore_capacity_size
    {ds_name!="StoreNAS-Public",instance!="
    arita-master:32704"} *100 > 80
```

設定ファイル 1 は，ESXi の StoreNAS-Public のデータストアと，arita-master:32704 のインスタンスを除いたデータストアの最大容量を 100.0%とした際の，データストア使用率が 80.0%を超えたときにアラートを出す式設定である．このアラートの閾値である 80.0%は，管理者の経験によって設定された．このアラートの式設定で使用されているメトリクス名は 2 種類である．1 種類目の vmware_datastore_capacity_size は，PM のデータストアの最大容量のメトリクス名である．2 種類目の vmware_datastore_freespace_size は，PM のデータストアの空き容量のメトリクス名である．また全てのメトリクス名に，StoreNAS-Public のデータストアを除外すること，arita-master:32704 を除外することを設定ファイル 1 の「{}」内にラベルとして記入している．StoreNAS-Public は，OS の image や，VM の設定ファイルである.xml，BIOS 設定ファイルである.nvram を例とした，アーカイブ化した VM のデータが保存されている．よって StoreNAS-Public は，管理者が関与していない間にデータストア容量が使用されることはないため，アラートを必要としておらず除外している．データストア使用率は，データストアの最大容量から，空き容量を引いたものを，最大容量で割り，100 倍することによって表す．そのデータストア使用率に，「> 80」の不等号の条件を与えることにより，データストア使用率が 80.0%を超えたときにアラートを出す式設定ができる.

課題

課題は，Prometheus においてアラートの式設定に時間を要することである．Slack にアラートを送信するためには，YAML 形式の設定ファイルにアラートの式設定をする必要がある．しかし複数のメトリクス名を組み合わせる際，設定ファイルに入力する文字数が多くなり，設定に時間を要する．CDSL でもちいる設定ファイル 1 を設定する手順を図 1 に示す.

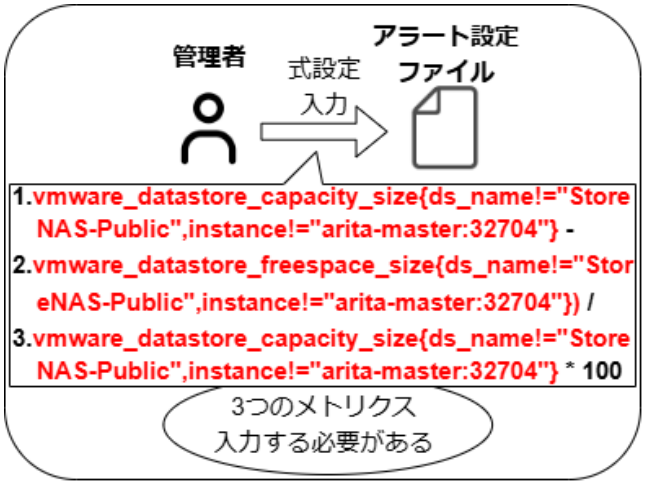

図 1: CDSL でもちいる設定ファイル 1 を設定する手順

この設定ファイル 1 の式設定は，管理者は 3 つのメトリクス名を入力する必要があるため，時間を要する.

基礎実験

アラートの式設定に要する時間を求めるために，基礎実験を行う．基礎実験の実験環境を図 2 に示す.

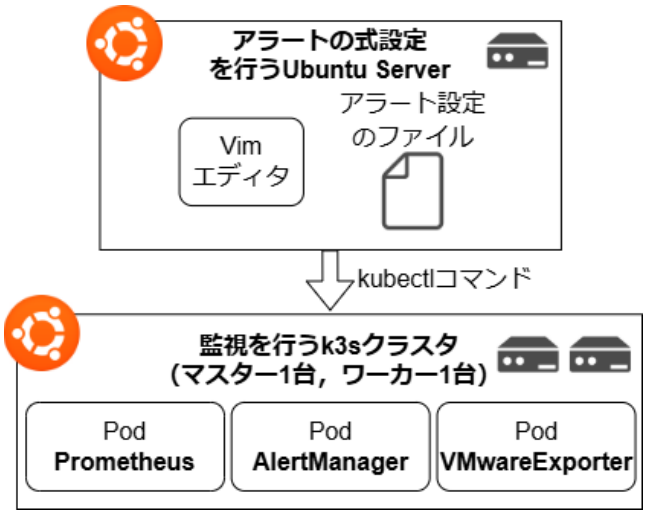

図 2: 実験環境

本稿の実験のために，Ubuntu Server として動作する VM3 台を用意した．それらの VM のうち 1 台で，アラートの式設定に要する時間を測定する．この Ubuntu Server は，監視サーバの k3s クラスタに kubectl コマンドを打つことができる．アラートの式設定を行うエディタとして，vim エディタを使っている．また監視サーバは，マスターに 1

台，ワーカーに 1 台の VM を使い k3s クラスタを構成しており，Prometheus と AlertManager，VMwareExporter の Pod がある．すべての VM は，CPU が 2Core，メモリが 16GB，ストレージが 25GB の構成である．すべての VM に Ubuntu 24.04.1 をインストールし，アラートの式設定を行う vim エディタのバージョンは 9.1 である．k3s のバージョンは v1.30.6+k3s1 となり，k3s で作成された Prometheus のバージョンは 2.53.1 である．

設定ファイル 1 の Jasmine のデータストア使用率が 80.0%を超えたときにアラートを出す式設定を，CDSL に所属している井出佑，近藤悠斗，西村克己，筒井優貴の 4 人の学生に入力してもらい，その時間を測定した．実験方法を図 3 に示す．

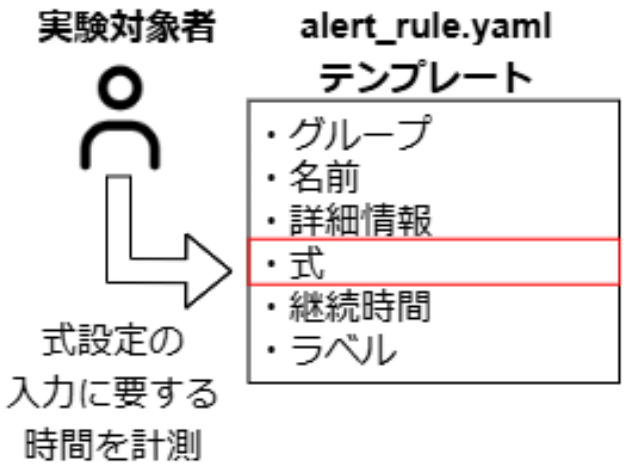

図 3: 実験方法

実験方法は，図 3 のように実験対象者に alert_rule.yaml テンプレートを提示し，アラートの式設定を行った．alert_rule.yaml テンプレートを設定ファイル 2 に示す．

alert_rule.yaml テンプレートだけでは，アラートの式設定が未入力のため，実験の前に設定方法について説明を行った．また，実験前と実験中に設定に関する質問を受けた．実験のために説明した内容は以下の通りである．

- 設定ファイルを開くまでのコマンド入力手順の説明
  - 実験を行う Ubuntu Server に SSH でログインする
    - ssh cdsl@arita-master3
    - cdsl@arita-master3's password: パスワードを入力
  - test_new ディレクトリに移動
    - cd test_new
  - vim エディタで YAML 形式のファイルを作成
    - vim 名前-学籍番号.yaml
- アラートの設定方法の説明
  - 手順

設定ファイル 2: alert_rule.yaml テンプレート

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: esxi-rule
  namespace: monitoring
data:
  esxi-monitoring-rules.yml: |-
    groups:
    - name: esxi-vm-rule
      rules:
      - alert: disk-usage-high-80
        annotations:
          description: "HOST：‘{{$labels.
              host_name}}‘␣disk␣usage␣high"
          runbook_url: https://{{$labels.
              host_name}}/ui/#/login
          summary: "HOST：‘‘{{$labels.host_name
              }}‘␣disk␣usage␣high"
        expr:
        for: 3m
        labels:
          severity: warning
```

    - 設定ファイル 2 の alert_rule.yaml テンプレートを名前-学籍番号.yaml ファイルに貼り付け
      - alert_rule.yaml テンプレートをコピー
      - vim エディタ内で右クリック
    - vim エディタをインサートモードにする
      - 「i」を入力
    - アラートの式設定を行うために，expr: の行に設定を入力
      - アラート内容は，ESXi の StoreNAS-Public のデータストアと，arita-master:32704 のインスタンスを除いたデータストアの最大容量を 100.0%とした際の，データストア使用率が 80.0%を超えたとき
    - vim エディタをコマンドモードにする
      - Esc キーを押す
    - 設定に使用したファイルを保存
      - 「:wq」を入力
  - 使用するメトリクス
    - データストアの最大容量
      - vmware_datastore_capacity_size
    - データストアの空き容量
      - vmware_datastore_freespace_size
  - アラートの入力方法
    - expr: アラート内容
  - アラートの式
    - (最大容量 - 空き容量) / 最大容量 * 100 > 80
- ラベルの設定方法の説明
  - 使用ラベル

    - StoreNAS-Public のデータストアを除外する
        - ds_name!=”StoreNAS-Public”
    - arita-master:32704 のインスタンスを除外する
        - instance!=”arita-master:32704”
- ラベルの付け方
    - メトリクス名 { ラベル 1, ラベル 2}

実験では．実験対象者が式の入力を開始したときから，入力が終了し式が正しく設定できるまでの時間を測定した．また，実験中の質問とその回答をする時間も測定時間に含めた．式が正しく入力できたかは，実験中に目視で確認している．

提案適用前のデータストア使用率が 80.0%を超えたときにアラートを出す式設定に要する時間を測定した実験の結果を，表 1 に示す．

表 1: 提案適用前のアラートの式設定に要する時間

名前	要する時間
井出 佑	約 305 秒
近藤 悠斗	約 195 秒
西村 克己	約 238 秒
筒井 優貴	約 241 秒

4 人の学生が，データストア使用率が 80.0%を超えたときにアラートを出す式設定をするのに要する時間は，井出佑が約 305 秒，近藤悠斗が約 195 秒，西村克己が約 238 秒，筒井優貴が約 241 秒となり，平均時間は約 245 秒になった．

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方式について述べる．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，評価実験として実験内容と実験結果と分析について述べる．第 6 章では，提案手法についての議論を述べる．最後に，第 7 章にて結論を述べる．

2. 関連研究

マークアップ言語である Ansible YAML で，大規模言語モデルを使い YAML 形式のファイルを生成する研究がある [8]．研究では，GitHub，Google BigQuery，GitLab，Ansible Galaxy からデータセットを作成し，事前トレーニングと微調整を加えた後，評価を行った．自動的に設定ファイルを生成しているため，設定を書くのに要する時間はなくなるが，Prometheus には適用されていない．

Visual Studio Code において，対話的定理証明を行うための言語である，Mizar 言語の予測変換を作成した研究がある [9]．研究では，Mizar 言語の中身を分割する前処理フェーズ，トレーニング用のデータを用意して木構造を作り次の入力を予測するためのトレーニングフェーズ，実際の予測変換を行う予測変換フェーズに分かれている．Mizar 言語の予測変換を作成しているが，Prometheus には適用されていない．

AWS 上にクラウド環境を自動デプロイする研究がある [10]．研究では，クラウド環境を導入する一連のプロセスの中に Prometheus のインストールを組み込んでおり，Github Actions を使って，Prometheus を自動でデプロイしている．Prometheus 自体は自動デプロイすることができるが，Prometheus のアラートの設定までは自動で設定されない．

3. 提案

提案方式

本稿では，Prometheus におけるアラートの式設定の際に，メトリクス名の候補を表示し，その候補の中から選択したメトリクス名を出力することで，入力する文字数を削減する手法を提案する．提案方式では，管理者がアラートの式設定のために，YAML 形式の設定ファイルに文字を入力した際，メトリクス名の候補の表示を行う．その後，表示した候補を選択すると，選択したものが出力される．

文字が入力されると，式設定を行っている文字列の取得が行われる．例として，「expr: vmware_d」と入力されたら，「vmware_d」の文字列を取得する．メトリクス名の候補に関係しない文字を認識しないために，文字列を定義する．定義した文字列では，A から Z，a から z の英字と，1 から 9 の数字，アンダーバーのみが連続している文字の列を指す．また，定義された文字列同士の間に，定義に当てはまらない文字がある際，2 つの文字列はそれぞれ別の文字列とする．本稿に出てくる文字列は，この定義を満たしたものとする．

文字列が取得されたら，メトリクス名と文字列が前方一致しているか確認する．図 4 は，メトリクス名と文字列

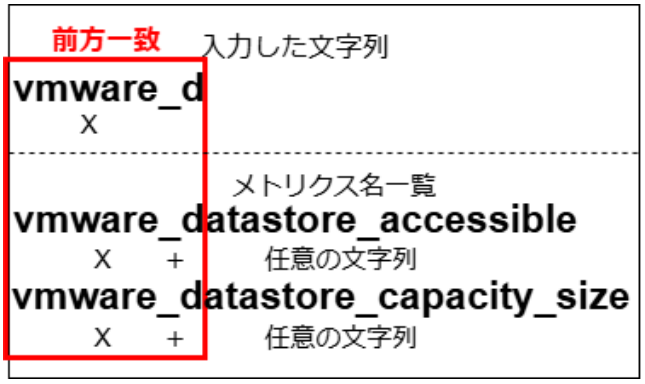

図 4: メトリクス名と文字列が前方一致

の前方一致を示す．入力された文字列を X とする．そし

て，文字列 X と前方一致しているメトリクス名を探す．対象となるメトリクス名の文字が X+任意の文字列となる際，X が対象となるメトリクス名の先頭部分と一致しているため，メトリクス名と文字列 X は前方一致していると定義する．例として図 4 では，「vmware_d」を X として赤枠で囲っている．「vmware_datastore_accessible」や「vmware_datastore_capasity_size」は，文字列 X である「vmware_d」に，「atastore_accessible」や「atastore_capasity_size」が加わったメトリクス名であるため，メトリクス名と文字列 X は前方一致している．

その後, 前方一致したメトリクス名を候補として入力した文字列の下に表示し, 表示した候補を選択すると, 選択したものが出力される．例として「vmware_datastore_accessible」を選択することで，そのメトリクス名がファイル内に出力される．

アラートの式設定を行っている文字列の取得方法

メトリクス名には，テキストファイルでも使用される文字列が含まれている．そのため，メトリクス名と文字列が前方一致すると，Prometheus の設定ファイルではないファイルにもメトリクス名の候補を表示する．よって，アラートの式設定を行っているときだけ候補を表示する．

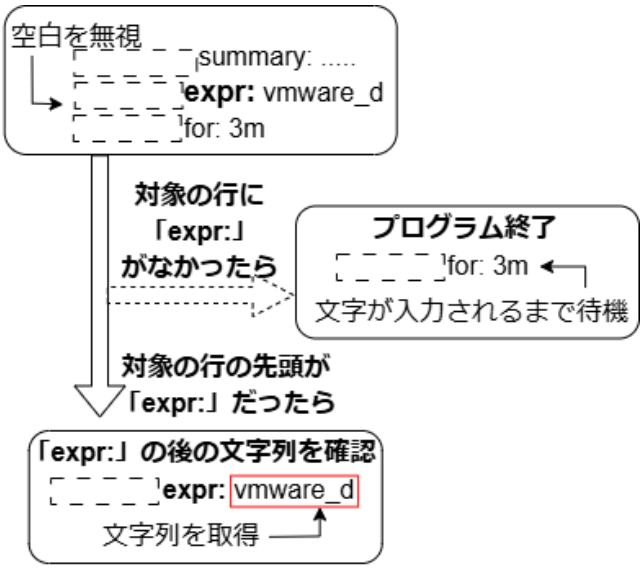

図 5: アラートの式設定の際だけメトリクス名の候補を表示し，候補を表示する対象の文字列の取得方法

図 5 は，アラートの式設定の際だけメトリクス名の候補を表示し，候補を表示する対象の文字列の取得方法である．管理者が YAML 形式の設定ファイルに入力した文字から，アラートの式設定を行っているか判断する．アラートの式設定を行っているかどうかの判別方法は，空白がある際には空白を無視し，カーソルがある位置の行の一番最初の文字が「expr:」であるかで判別している．一番最初の文字が「expr:」だった際，「expr:」の後ろから改行が行われるまでの文字列を確認する．対象となる文字列が入力されていたら，メトリクス名と入力した文字列が前方一致しているか確認し，メトリクス名の候補を表示する．その行に「expr:」が存在しなかった際，プログラムを終了させ，新しい文字が入力されるまでプログラムは待機する．

メトリクス名と入力した文字列が前方一致しているか確認する方法

アラートの式設定を行っている文字列の取得ができたら，その文字列を対象にメトリクス名の候補を表示する．

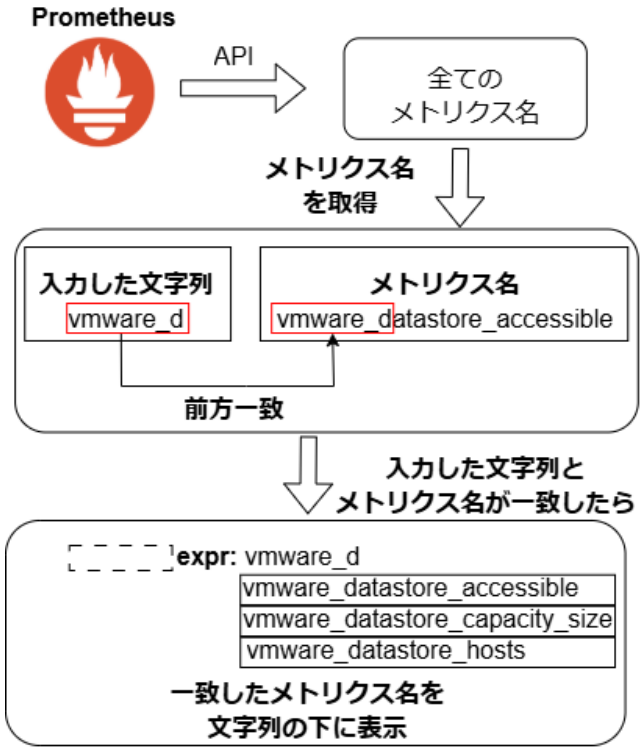

図 6: アラートの式設定を行っているメトリクス名が，入力された文字列と前方一致しているかを確認する方法

図 6 は，アラートの式設定を行っているメトリクス名が，入力された文字列と前方一致しているかを確認する方法を示している．Prometheus の API を使って，監視サーバが監視している全てのメトリクス名を取得する．そして，入力した文字列と取得したメトリクス名が前方一致していないか確認する．入力した文字列「vmware_d」とメトリクス名の「vmware_datastore_accessible」は，赤枠で囲まれた部分が「vmware_d」で前方一致している．最後に，「vmware_datastore_accessible」をアルファベットの昇順に文字列の下に表示している．

ユースケース・シナリオ

本稿では，PM の管理者がアラートがないときと比べて

早期の問題発見をするため，PM のデータストア使用率のアラートを設定を行う場面をユースケースにしている．VM は PM のデータストアを使って作成するため，データストア容量が不足すると，新たな VM を作成できなくなる．よって，管理者はデータストアの不足を出来るだけ早く問題を発見し，不足していたらデータストア容量を拡張するため，データストア使用率のアラートの設定を行う．アラートを設定しデータストアを拡張するまでの手順を図 7 に示す．

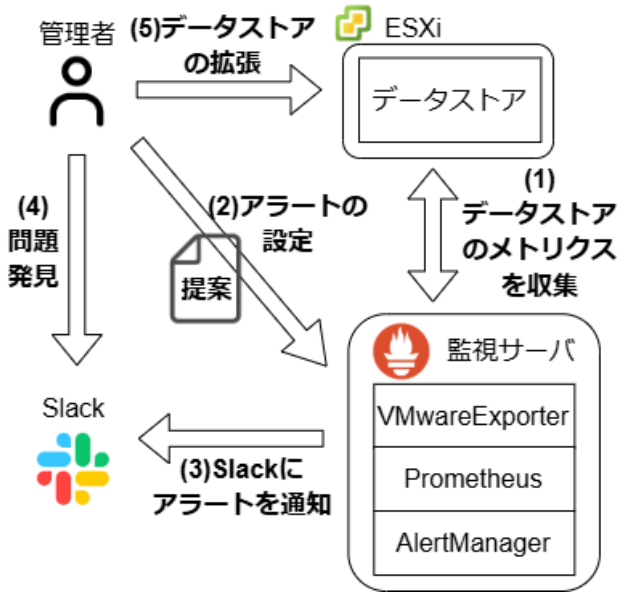

図 7: アラートを設定しデータストアを拡張するまでの手順

図 7 では，(1) で監視サーバがデータストアのメトリクスを収集する．監視サーバの VMwareExporter が ESXi からデーアストアのメトリクスを取得し，Prometheus に送っている．(2) は管理者が監視サーバに対してアラートの設定を行っている．提案手法であるメトリクス名の候補の表示を行い，データストア使用率が 80.0%を超えたのときにアラートを出す式設定を行う．提案適用前では全ての文字を入力していたが，提案適用後ではメトリクス名が候補として表示されるため，文字を入力する手間が減り，時間を削減することができる．(3) は，Slack にアラートの通知を行っている．(2) で設定したアラートの設定をもとに，データストア使用率が 80.0%を上回っていたら監視サーバの AlertManager が Slack のアラートの．(4) は，管理者が問題発見をする．Slack に通知されたアラートを管理者が確認することによって，データストア容量が 80.0%を上回っていることを知ることができる．(5) は，データストアの拡張を行う．PM のデータストア容量が 80.0%を上回っていることを知った管理者は，データストアの拡張を行うことによって，データストアの不足が起きる前に問題を解決することができる．

4. 実装

提案手法である，Prometheus のメトリクス名の候補を表示するために，vim エディタの設定を行った．vim エディタは，~/.vimrc のファイルに設定を記入することで，vim エディタが起動されるたびに設定が実行され，vim エディタをそのユーザ毎に設定した環境で使用することが可能になる．

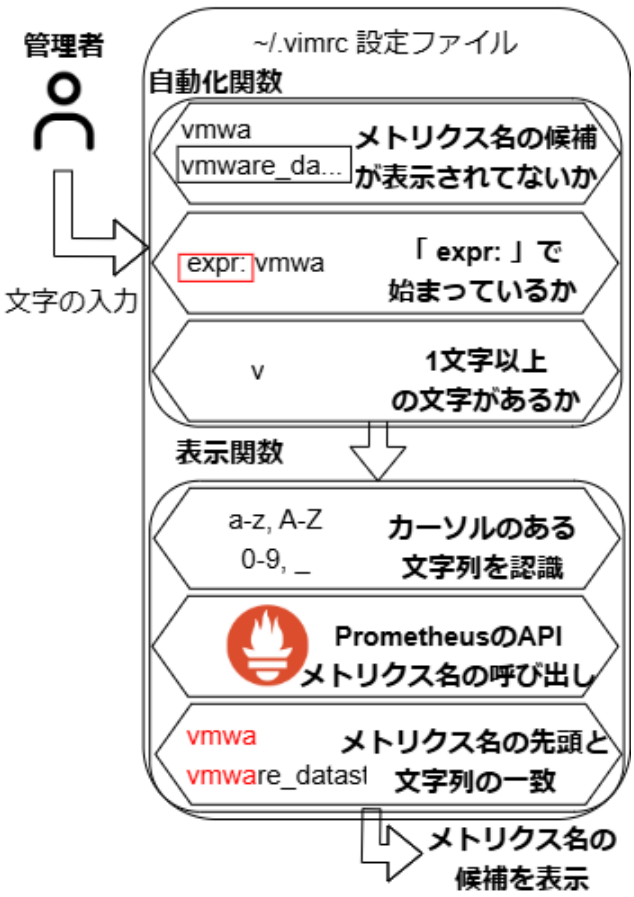

図 8: ~/.vimrc に実装した，Prometheus のメトリクス名の候補を表示するアルゴリズム

図 8 は，~/.vimrc に実装した，Prometheus のメトリクス名の候補を表示するアルゴリズムを示している．メトリクス名の候補を表示するために，2 つの関数が使用されている．候補の表示を行うのか判断する自動化関数と，実際に表示を行う表示関数である．

自動化関数

ユーザが vim エディタに文字を入力しようとしたとき，自動化関数が実行される．自動化関数では，3 つの条件をもとにメトリクス名の候補を表示するのか判断をする．

1 つ目は，メトリクス名の候補が既に表示されてないかを判別する．もし候補が既に表示されていたら，自動化関数が終了する．2 つ目は，文字が入力された行が「expr:」

で始まっているのかどうか判別する．アラートの式設定は「expr:」で始まるため，その行のみに提案手法を適用することによって，Prometheus の式設定を書いているのかどうか判別できる．3 つ目は，「expr:」の後に選択されたカーソルがある位置の前に，1 文字以上文字が入力されているのかどうかで判別する．この条件により，1 文字も書かれていない際も候補が表示されることを防ぐことができる．

以上 3 つの条件が満たされているとき，Ctrl+X と Ctrl+U の入力がプログラム内で実行され，表示関数を呼び出す．

表示関数

表示関数は，Ctrl+X と Ctrl+U の入力で呼び出すことができる変数 completefunc に定義した関数であり，本稿では，自動化関数から呼び出される．呼び出された表示関数は，関数が 2 回呼び出される．1 回目ではメトリクス名の候補を表示する対象の文字列を認識する．認識する文字列を提案手法で定義された文字列の定義により，A から Z，a から z の英字と，1 から 9 の数字，アンダーバーのみに限定する．vimrc には文字クラスが設定されており，単語構成文字列を指定する文字クラスである \w を使用することで，文字列を指定できる．カーソルがある文字列の最初の文字の位置を特定するため，行の最初からカーソルがある位置まで while 文で文字列を繰り返し取得する．最後に，取得した文字列の最初の文字の位置を返す．

2 回目では，Prometheus の API からメトリクス名を取得する．その後，メトリクス名と 1 回目で取得した文字列が前方一致していたら，そのメトリクス名を取得した文字列の下に候補として表示し，選択できるようにする．

5. 評価実験

評価実験では，設定ファイル 1 の ESXi の StoreNAS-Public のデータストアと，arita-master:32704 のインスタンスを除いたデータストアの最大容量を 100.0%とした際の，データストア使用率が 80.0%を超えたときにアラートを出す式設定に要する時間を，提案適用前と適用後で評価を行う．基礎実験で，提案適用前のアラートの式設定に要する時間を求めた．そのため，提案適用後のアラートの式設定に要する時間を測定し，提案適用前から設定に要する時間がどれくらい変化したのかを求めた．

提案手法を適用後，基礎実験に参加した CDSL に所属している 4 人の学生を対象に，再びアラートの式設定に要する時間を測定した．実験では，基礎実験の実験方法である図 3 と同じような実験方法で実験した．実験対象者に，設定ファイル 2 が書かれた alert_rule.yaml テンプレートを学生に提示した．設定ファイル 2 は，アラートの式設定を入力する expr: の行だけは設定されていない，YAML 形式のファイルである．提示した設定ファイルのテンプレートを使用し，vim エディタでアラートの設定を行った．実験では，基礎実験と同じように実験の前に設定方法について説明した．また，実験前と実験中に質問を受けた．

実験では．実験対象者が式の入力を開始したときから，入力が終了し式が正しく設定できるまでの時間を測定した．また，実験中の質問とその回答をする時間も測定時間に含めた．式が正しく入力できたかは，実験中に目視で確認している．

実験環境

評価実験は，基礎実験の環境である図 2 と同一の環境で実施した．本稿の実験のために，Ubuntu Server として動作する VM3 台を用意した．それらの VM のうち 1 台で，アラートの式設定に要する時間を測定する．この Ubuntu Server は，監視サーバの k3s クラスタに kubectl コマンドを打つことができる．アラートの式設定を行うエディタとして，vim エディタを使っている．また監視サーバは，マスターに 1 台，ワーカーに 1 台の VM を使い k3s クラスタを構成しており，Prometheus と AlertManager，VMwareExporter の Pod がある．すべての VM は，CPU が 2Core，メモリが 16GB，ストレージが 25GB の構成である．すべての VM に Ubuntu 24.04.1 をインストールし，アラートの式設定を行う vim エディタのバージョンは 9.1 である．k3s のバージョンは v1.30.6+k3s1 となり，k3s で作成された Prometheus のバージョンは 2.53.1 である．

実験結果と分析

データストア使用率が 80.0%を超えたときにアラートを出す式設定に要する時間を，図 9 に示す．

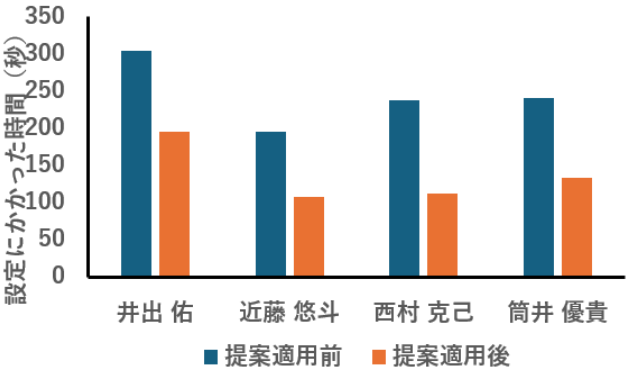

図 9: データストア使用率が 80.0%を超えたときにアラートを出す式設定に要する時間

データストア使用率が 80.0%を超えたときにアラートを出す式設定に要する時間は，井出佑は提案適用前は約 305 秒だったのが提案適用後は約 196 秒になり，約 35.7%削減した．近藤悠斗は提案適用前は約 195 秒だったのが提案適用後は約 107 秒になり，約 45.1%削減した．西村克己は，提案適用前は約 238 秒だったのが提案適用後は約 112 秒

になり，約 52.9%削減した．筒井優貴は，提案適用前は約 241 秒だったのが提案適用後は約 133 秒になり，約 44.82%削減した．4 人の平均時間は，提案適用前は約 245 秒だったのが提案適用後は約 137 秒となり，44.1%の時間を削減できた．

基礎実験でアラートの式設定に要する時間が長い人ほど時間削減率が高いため，文字を打つのに時間を要する人ほど時間削減率が高い．

同じ実験対象者が基礎実験と評価実験で合計 2 回，同じアラートの式設定をした．そのため，2 回目の評価実験では 1 回目の基礎実験のときに設定した記憶が残っており，学習効果により作業時間が削減された．

提案適用前は，入力したメトリクス名に誤字がないかの確認や，間違っていた際の書き換えの時間がかかっていた．提案適用後では，メトリクス名の候補の表示によってメトリクス名が自動で入力されたため，入力時のミスを気にする時間がなくなり，文字を入力している時間以上の削減効果があった．

6. 議論

本稿の提案手法では, メトリクス名の候補の表示を「expr:」の後ろから改行が行われるまでの間の文字列に限定することによって，Prometheus の設定ファイルではないファイルにも候補を表示することを防ぐことができる．しかし，候補を表示する対象であるメトリクス名ではなく，ラベルを入力する際にもメトリクス名の候補が表示されている．解決方法として，カーソルの位置が「{」の後ろの文字列かつ，その後ろに「}」がない際，候補の表示を無効にする方法がある．ラベルは「{」と「}」の間に入力される．そのため，カーソルの位置が「{」の後ろの文字列かつ，その後ろに「}」がない際はラベルであるとして，候補の表示を無効にする．

本稿の実験では，同じ実験対象者が基礎実験と評価実験で合計 2 回，同じアラートの式設定をした．そのため，2 回目の評価実験では 1 回目の基礎実験のときに設定した記憶が残っており，学習効果により作業時間が削減された．解決方法として，実験対象者の入力した文字数で評価する方法がある．実験では, アラートの式設定にかかった時間を提案適用前と適用後で比較して評価を行っていたが，入力した文字数を比較して評価を行う．入力する文字数自体は人間の記憶と関係性がないため，学習効果により作業時間が削減されても，公平な評価をすることができる．

本稿の提案手法では，メトリクス名の候補を表示することができる．候補の表示はアルファベットの昇順に表示されるが，アルファベットの後半のメトリクス名を選択するのに時間を要する．解決方法として，Github からメトリクス名の組み合わせを集め，既に入力したメトリクス名と演算子の次に入力されるメトリクス名の候補を，Github で集めたデータの中から多かった順に表示する方法がある．アラートの式設定の際，メトリクス名の後ろに加算や除算を行う演算子をつけることがある．例として，設定ファイル 1 では，減算，乗算，除算が使用されている．演算子を使って，複数のメトリクス名を組み合わせてアラートの式設定を行う．そして，Prometheus のアラートを設定した YAML 形式の設定ファイルを Github で集め，それぞれのアラートの式設定が，どのようにメトリクス名と演算子を組み合わせるのかを調査する．そして候補を表示するとき，Github で集めたメトリクス名の組み合わせの情報を使い，既に入力したメトリクス名と演算子の次に入力されることが多い順に並び替える．

7. おわりに

PM では，データストア容量が不足し，新たに VM を作成できないことがある．管理者はアラートにより，アラートがないときと比べて，早期にデータストア容量の不足を知ることができる．そのため，VM を作成できなくなる前に未然に防ぐことができる．アラートを生成するソフトウェアとして Prometheus がある．Prometheus では，管理者が YAML 形式の設定ファイルにアラートの設定を行うことで，通知が行われる．課題は，アラートの式設定を行う際，複数のメトリクス名の変数を組み合わせることがあるため，設定ファイルに入力する文字数が多くなり，設定に時間を要することである．基礎実験では，4 人の学生を対象に，Jasmine のデータストア使用率が 80.0%を超えたときにアラートを出す式設定を行った．データストア使用率のアラートの式設定は，vmware_datastore_capacity_size と_datastore_freespace_size の 2 種類のメトリクス名が組み合わされたものである．結果は，アラートの式設定に要する平均時間は，約 245 秒であった．提案では，アラートの式設定を行う際に，メトリクス名の候補を表示し，その候補の中から選択したメトリクス名を出力することで，入力する文字数を削減する手法を提案する．監視サーバのメトリクス名を全て取得し，Prometheus のメトリクス名と管理者が入力している文字列が前方一致していたら候補を表示する．評価実験では，4 人の学生を対象に，データストア使用率が 80.0%を超えたときにアラートを出す式設定に要する時間を測定した．結果は，提案適用前の平均時間約 245 秒に対し，適用後は約 137 秒で，約 44.1%の時間を削減することができた．

謝辞

本稿の執筆にあたり，Prometheus について指導していただいた，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の平尾真斗さん，東京工科大学コンピュータサイエンス学部の加藤健吾さんに御礼申し上げます．また，基礎実験と評価実験に協力していただ

いた，東京工科大学コンピュータサイエンス学部の井出佑さん，近藤悠斗さん，西村克己さん，筒井優貴さんに御礼申し上げます．

参考文献

[1] Sala, G., Sgandurra, D. and Baiardi, F.: Security and Integrity of a Distributed File Storage in a Virtual Environment, *Fourth International IEEE Security in Storage Workshop*, pp. 58–69 (online), DOI: 10.1109/SISW.2007.10 (2007).

[2] Mandal, R., Mondal, M. K., Banerjee, S., Chatterjee, P., Mansoor, W. and Biswas, U.: Design and implementation of an SLA and energy-aware VM placement policy in green cloud computing, *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 777–782 (online), DOI: 10.1109/GCWkshps56602.2022.10008675 (2022).

[3] Ashu, Kaur, A., Singh, M. and Singh, P.: A taxonomy, survey on placement of virtual machines in cloud, *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 2054–2058 (online), DOI: 10.1109/ICECDS.2017.8389810 (2017).

[4] Mann, Z. c.: Multicore-Aware Virtual Machine Placement in Cloud Data Centers, *IEEE Transactions on Computers*, Vol. 65, No. 11, pp. 3357–3369 (online), DOI: 10.1109/TC.2016.2529629 (2016).

[5] Mart, O., Negru, C., Pop, F. and Castiglione, A.: Observability in Kubernetes Cluster: Automatic Anomalies Detection using Prometheus, *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 565–570 (online), DOI: 10.1109/HPCC-SmartCity-DSS50907.2020.00071 (2020).

[6] Sharma, V.: Managing Multi-Cloud Deployments on Kubernetes with Istio, Prometheus and Grafana, *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Vol. 1, pp. 525–529 (online), DOI: 10.1109/ICACCS54159.2022.9785124 (2022).

[7] Abirami, T., Mapari, S., Jayadharshini, P., Krishnasamy, L. and Vigneshwaran, R. R.: Streamlined Deployment and Monitoring of Cloud-Native Applications on AWS with Kubernetes Prometheus Grafana, *2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, pp. 1149–1155 (online), DOI: 10.1109/ICAICCIT60255.2023.10465818 (2023).

[8] Pujar, S., Buratti, L., Guo, X., Dupuis, N., Lewis, B., Suneja, S., Sood, A., Nalawade, G., Jones, M., Morari, A. and Puri, R.: Invited: Automated Code generation for Information Technology Tasks in YAML through Large Language Models, *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–4 (online), DOI: 10.1109/DAC56929.2023.10247987 (2023).

[9] Taniguchi, H. and Nakasho, K.: Visual Studio Code Extension and Auto-completion for Mizar Language, *2021 Ninth International Symposium on Computing and Networking (CANDAR)*, pp. 182–188 (online), DOI: 10.1109/CANDAR53791.2021.00033 (2021).

[10] Abirami, T., Mapari, S., Jayadharshini, P., Krishnasamy, L. and Vigneshwaran, R. R.: Streamlined Deployment and Monitoring of Cloud-Native Applications on AWS with Kubernetes Prometheus Grafana, *2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, pp. 1149–1155 (online), DOI: 10.1109/ICAICCIT60255.2023.10465818 (2023).