テキストに含まれる定義した文字列の検出による ファイルのログタイプの特定時間の短縮

岡田 京太郎[1]　三上 翔太[1]　串田 高幸[1]

概要：ログはシステムの障害対応に使用される．ログの管理や検索には Elasticsearch を使用する．ログファイルを収集するために Filebeat を使用し，収集したログは Logstash で変換され，Kibana で可視化する．Filebeat の設定では収集対象のログのファイルパスとタイプの指定を行う必要がある．課題は，ログタイプの種類を把握していない場合に管理者はログタイプの種類を調べる必要があり，特定までに時間を要することである．提案手法は，管理者が入力したファイル内テキストに定義した文字列をもとに，ログタイプを特定する．すべての行に stdout または stderr が検出された場合，ログタイプを Container と特定し，すべての行にホスト名，プロセス名とプロセス ID が連続しているプロセス識別子が検出された場合，ログタイプを Syslog と特定する．ログタイプが Container と Syslog ではない場合ログタイプを Filestream と特定する．3 人の学生を評価対象とし，提案ソフトウェア使用前と使用後のログタイプの特定開始から提示されたコマンドに特定したログタイプを記入するまでの時間を評価した．提案ソフトウェアを使用せずに 18 個のファイルのログタイプを特定してもらい，ログタイプの特定開始から提示されたコマンドに特定したログタイプを入力するまでの時間を計測した．18 個のファイルのログタイプの特定に要した 3 人の学生の平均時間は約 319 秒であり，特定したログタイプは実際のログタイプとすべて一致していた．3 人の学生の提案ソフトウェア使用前と使用後の 18 個のファイルのログタイプの特定に要した平均時間は約 319 秒から約 46 秒で約 273 秒短縮し，約 85.6%削減した．提案ソフトウェアを使用した場合もログタイプは実際のログタイプとすべて一致していた．

1. はじめに

背景

ログはシステムに関する活動を時系列で記録している [1–3]．例えば，ログファイル内のログはタイムスタンプ，イベントの説明，システム情報，エラー情報で構成されている [4,5]．ログファイルは，オペレーティングシステム，データベース，ソフトウェアアプリケーションによって生成される [6]．ログ分析は，システム監視，トラブルシューティング，セキュリティ管理でもちいられる [7,8]．また，ログファイルにはイベントやシステム情報が記録されており，エンジニアはログを使ってシステムを把握し，システム障害を識別できる [9,10]．

ログの収集，保存，検索，分析の一連のログ管理プロセスを実行するための手法が必要とされている．この一連のプロセスは ELK Stack をもちいて行われる．ELK Stack は Elasticsearch，Logstash，Kibana，Filebeat で構成されており，仮想ハードウェア環境で実行できる [11]．ELK Stack はログ分析に使用され，ログ分析に役立つ機能が備わっている [12]．ログの分析や構造化には Logstash，ログの保存や検索には Elasticsearch，ログの可視化には Kibana，ログの収集には Filebeat を使用する [13]．Elasticsearch とは膨大なデータをリアルタイムで処理できる検索エンジンである [14, 15]．Logstash とはログを取り込み，変換し，他の場所に送信するシステムである [16, 17]．Kibana とはデータの分析，表示，検索に使用される分析ツールである [18]．Filebeat とは Logstash にデータを転送するためのログシッパーである [19, 20]．Filebeat をインストールする場合，Kubernetes 環境において Filebeat を helm を使用してデプロイし，Elasticsearch へのログデータを送信する標準コマンドがある．コマンドにログファイルの収集先とログファイルの種類の指定を行う必要がある．

ログタイプは 31 種類ある．ログを収集する場合に収集するログのファイルパスの指定が必要となるログタイプは，Container，Filestream，Google Cloud Storage，Syslog，Unix の 5 種類である．その中で仮想マシン (以後 VM と呼ぶ) のオペレーティングシステム (以後 OS と呼ぶ) 内のログファイルを収集する場合にファイルパスの指定が必要と

[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

なるログタイプは，Container，Filestream，Syslog の 3 種類である．

Container のログは，タイムスタンプ，標準出力または標準エラー出力，ログフラグ，ログレベル，ログメッセージから構成されている．標準出力である stdout と標準エラー出力である stderr はコンテナ環境で使用され，Docker や Kubernetes のコンテナランタイムのログ出力を表す．

Syslog のログは，タイムスタンプ，ホスト名，プロセス名とプロセス ID が連続しているプロセス識別子，ログメッセージから構成されている．ホスト名は，どのデバイスやマシンのログであるかを表す．プロセス名とプロセス ID が連続しているプロセス識別子は，システムの性能やプロセスの状態を表す．

Filestream は，Ubuntu 内の Container と Syslog 以外のすべてのログファイルである[*1]．

課題

課題は，管理者がログタイプを把握していない場合，ログタイプを調べるため，作業の工数が増えてログタイプの特定に時間を要することである．管理者がログタイプを把握している場合，作業はコマンドの入力のみである．しかし，管理者がログタイプを把握していない場合，Web サイトにアクセスし，閲覧する．Web サイトでログタイプを調べて，コマンドを入力する．このように，作業の工数が増えるため，時間を要する．

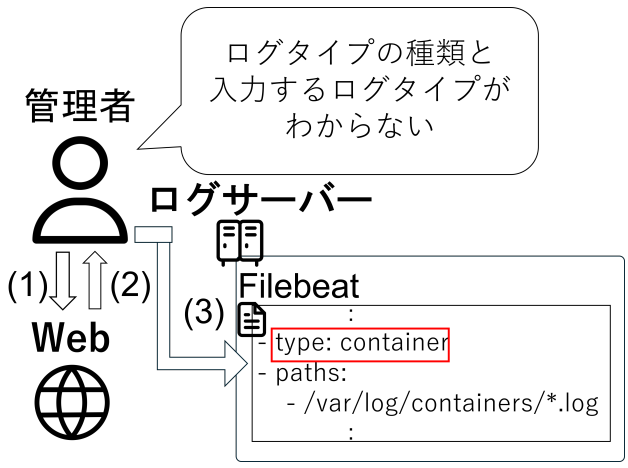

図 1　ログタイプを特定するまでのプロセス

図 1は，管理者がログタイプを把握していない場合，ログタイプの特定に時間を要することを表している．管理者はログサーバーを管理しており，Filebeat でログの収集を行っている．Web ブラウザでは，必要な情報を調べることができる．Filebeat はログサーバーに送るツールである．まず (1) で，管理者がログタイプを調べる．調べる方法の例として，生成 AI の使用や Web ブラウザで検索する方法があげられる．図 1では例として Web ブラウザを使用した場合を表している．(2) で Web ブラウザで検索をかけ，検索結果からログタイプを特定する．(3) で特定したログタイプを赤枠で囲まれている部分に記入する．管理者はログタイプの種類を把握していない場合，ログタイプの種類やファイル内のテキストを確認する必要があり時間を要する．

基礎実験

基礎実験は，特定するログタイプが Container の場合で行い，ログタイプを把握していない場合に，ログタイプの特定にどのくらい時間を要するか計測した．ログタイプが書かれていない helm で Filebeat をインストールする場合のコマンドを提示し，提示したコマンドにログタイプを入力するまでの時間を計測した．実験環境として，以下の VM を 1 台用意した．

- CPU：4Core
- メモリ:16GB
- ストレージ：25GB
- OS：Ubuntu24.04

また，以下に VM にインストールしたツールとデプロイした Pod のバージョンを記載する．

- k3s：1.30.6
- Helm:v3.16.3
- Elasticsearch：8.5.1
- Logstash：8.5.1
- Kibana：8.5.1
- Filebeat：8.5.1

基礎実験では，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) の井出佑さん，筒井優貴さん，山本真也さんに協力してもらった．ファイルパスは書かれているが，ログタイプが書かれていないコマンドを提示し，ログタイプの特定に要する時間を計測した．ログタイプを特定してもらったファイルの個数は 18 個である．コマンド 1はログタイプが書かれていないコマンドである．

基礎実験を行うにあたり，以下の説明を行いログタイプの特定を行ってもらった．

- コマンド 1は helm を使用して，Filebeat をデプロイする場合に実行するコマンドであること．
- Filebeat をインストールする場合，ファイルパスの指

[*1] https:https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-filestream.html

コマンド 1 ログタイプが書かれていないコマンド

```
- type: [1]
- paths:
    - /var/log/containers/*.log
```

定とログタイプの指定をする必要があること．

- コマンド 1はファイルパスは指定されているが，ログタイプの指定がされていないこと．
- 他者に聞く以外は何を使用してもよいこと．
- コマンド 1の type:[1] に特定したログタイプを記入するまでの時間を計測すること．

説明する際に，コマンド 1は helm を使用して，Filebeat をデプロイする場合のコマンドであることを説明した．ログタイプの特定は，CDSL の学生にコマンド 1を表示し，説明した内容を理解したことを確認後，対象の学生がコマンド 1に入力できる状態になってから行った．計測を開始する/var/log/containers/*.log をもとにログタイプを特定し，コマンド 1の type:[1] に特定したログタイプを記入するように説明した．

表 1 基礎実験の計測結果

名前	特定時間
井出佑さん	60 秒
筒井優貴さん	310 秒
山本真也さん	489 秒

表 1は基礎実験の計測結果である．18 個のファイルのログタイプを特定してもらい，ログタイプの特定にかかった時間は井出佑さんが約 60 秒，筒井優貴さんが約 310 秒，山本真也さんが約 489 秒であり，3 人のログタイプの特定までの平均時間は約 319 秒であった．井出佑さんは生成 AI を使用した際に，コマンド 1をコピーし，生成 AI に貼り付け，ログタイプを特定するように指示を出した．生成 AI からの返答が特定するログタイプと一致しているのか確認せず，生成 AI からの返答をそのまま入力した．筒井優貴さんと山本真也さんは生成 AI にログタイプを特定するように指示を出し，返答を得た際，生成 AI からの返答がログタイプの種類とその説明であったため，ログタイプを特定するまで繰り返し質問をし，返答で特定されたログタイプが全て特定するログタイプと一致しているのか確かめていたため，ログタイプの特定に時間を要した．また，3 人の学生が特定した 18 個すべてのログタイプは実際のログタイプとすべて一致していた．

各章の概要

第 2 章では，本稿の課題や提案との関連研究について述べる．第 3 章では，課題に対しどのように解決するかの提案手法とユースケースとシナリオについて述べる．第 4 章では，提案手法をもとに作成したソフトウェア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．第 6 章では，提案手法についての議論を述べる．第 7 章では，本稿のまとめを述べる．

2. 関連研究

ログは構造化されていないことがあり，実務者によるログ分析が困難な場合がある．非構造化であるログを LogAssist をもちいて構造化する研究を行っている論文がある [21]. LogAssist は構造化されていないログを分析する．次に，グループ化 ID によりワークフローを形成することでイベントシーケンスの混在に関する課題を対処する．最後に，n-gram モデルを使用して，イベントシーケンスの識別を行い，識別されたシーケンスを使用し，ログを圧縮する．LogAssist でログを圧縮することにより，必要な場合のみログにアクセスできるようになる．これにより，ログ分析における調べる作業に要する労力を削減させた．本稿ではファイルパスをもとにファイル内のテキストを読み込み，テキストに定義した文字列をもとにログタイプを特定する．この論文では，非構造化なログを構造化させログ分析における実務者の労力削減に焦点を当てているため，ログタイプの特定は行っていない．

ログ分析においてログメッセージは，固定部分と可変部分から構成される．また，システムによってログメッセージ形式にバリエーションがあり，完全な形式が不明である．テンプレート識別問題を解決するために，MoLFI をもちいてテンプレートを作成し，ログメッセージを自動的に識別する論文がある [22]. MoLFI はトレードオフ分析にもとづいて検索ベースのアプローチを行う．システムにおいてバリエーションがあるログメッセージ形式を定式化し，ログエントリに一致するかつログイベントに固有のログメッセージテンプレートを生成することで，ログメッセージ形式を識別する．これにより，ログメッセージで使用されるテンプレート識別問題を解決する．この論文では，テンプレートを生成する点と可変データへの対応の観点で関連しているが，本稿が目的とするログタイプの特定は行っていない．

ログファイル内のログを分析するために，クラスタリング技術をもちいてイベントタイプの抽出を行った論文がある [23]. ログファイルには，一定のメッセージタイプと可変のパラメータータイプが混在しており，固有の変動性があるため，イベントタイプに抽象化を行う必要がある．クラスタリング技術を使用し，ログを分析することで異常検出や予測を行い，ログファイルの抽象化をする．この論文

では，ログファイルの抽象化のために，クラスタリング技術を使用し，ログを分析することで異常検出や予測を行っているが，ログタイプの特定は行っていない．

3. 提案

本稿では, テキストの特定の文字列をもとにログタイプを特定することを目的とする．提案手法では，すべての行に stdout または stderr が検出された場合，ログタイプを Container と特定し，すべての行にホスト名，プロセス名とプロセス ID が連続しているプロセス識別子が検出された場合，ログタイプを Syslog と特定する．ログタイプが Container と Syslog ではない場合，ログタイプを Filestream とする．これらの工程で管理者が入力したファイルのテキストに定義した文字列をもとにログタイプを特定することでログタイプの特定に要する時間の短縮をする．

提案方式

提案方式は，提案ソフトウェアに管理者がファイルパスを入力した場合，提案ソフトウェアがファイルパスを取得する．次に，ファイルパスで指定されたファイル内のテキストを読み込む．そして，Container に定義した文字列が検出された場合はログタイプが Container とする．Syslog に定義した文字列が検出された場合はログタイプが Syslog とする．ログタイプが Container と Syslog ではない場合，Filestream とする．

Container の特定方法

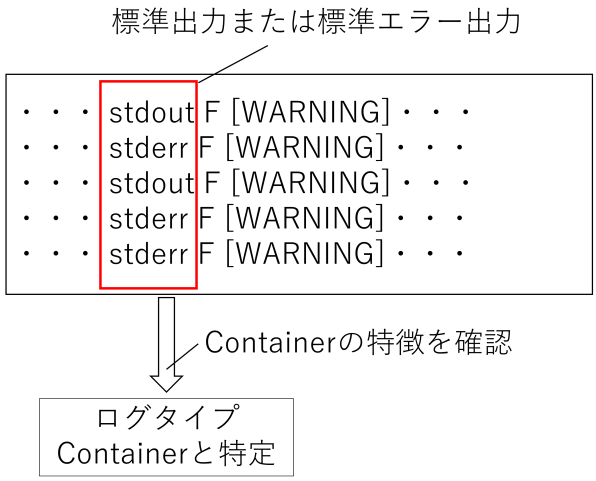

図 2 ログタイプが Container の場合の特定方法

図 2は，ログタイプが Container の場合の特定方法を表している．提案ソフトウェアで，入力されたファイルパスのファイル内のテキストを読み込み，ファイル内のテキストを 1 行ずつ確認する．ファイル内のすべての行に図 2の赤枠で表すように stdout または stderr が検出された場合，ログタイプが Container と特定し，提案ソフトウェアでログタイプを Container と出力する．ログタイプが Container のログは構成要素の 1 つに標準出力または標準エラー出力を表示する要素がある．例として，コンテナ環境のログでは標準出力は stdout と記載され，標準エラー出力は stderr と記載される．

Syslog の特定方法

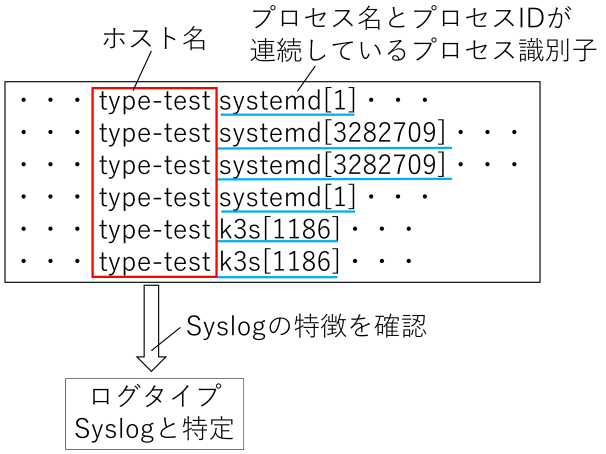

図 3 ログタイプが Syslog の場合の特定方法

図3は，ログタイプが Syslog の場合の特定方法を表している．ファイル内のテキストを 1 行ずつ確認する．ファイル内のすべての行に図3の赤枠内のホスト名と青の下線が引かれているプロセス名とプロセス ID が連続しているプロセス識別子が検出された場合，ログタイプが Syslog と特定し，提案ソフトウェア上でログタイプを Syslog と出力する．ホスト名を特定した後に，ホスト名をもとにプロセス名とプロセス ID が連続しているプロセス識別子を特定する．Syslog はログがタイムスタンプ，ホスト名，プロセス名とプロセス ID が連続しているプロセス識別子，メッセージから構成される．ホスト，プロセス名とプロセス ID が連続しているプロセス識別子は自由形式なため，定義を決めることで判定した．

ホスト名の判定方法は，ファイル内の 1 行目のテキストを変数として格納し，テキストをスペース区切りで単語に分ける．次の行のテキストと一語一句共通している単語を検出する．検出した単語をホスト名の候補とする．すべての行で検出したホスト名の候補が 1 つだった場合，その単語をホスト名とする．

プロセス名とプロセス ID が連続しているプロセス識別子の判定方法は，ファイル内のテキストをスペース区切りで単語に分ける．各単語に対して，プロセス名とプロセス ID が連続しているプロセス識別子の正規表現である (r'(\S+)\[(\d+)\]') と一致しているか確かめる．プロセス名は (\S+) ですべての Unicode の文字，数字，記号を表し，プロセス ID は \[(\d+)\] で [] で囲まれた 1 文字以上の数字を表す．一致した単語がすべての行にあった場合，その

単語をプロセス名とプロセス ID が連続しているプロセス識別子とする．

Filestream は，Container，Syslog と特定されなかったファイルとする．

ユースケースとシナリオ

本稿では，Ubuntu 内のログファイルを収集する場合を想定する．

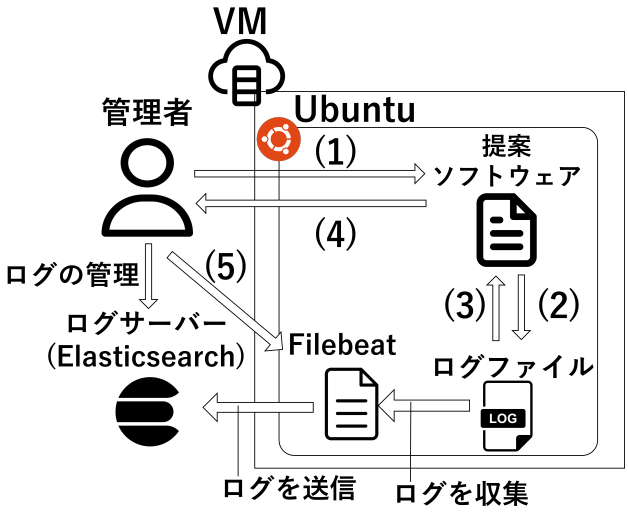

図 4　ログタイプの特定と特定したログタイプをコマンドライン上に表示するまでのプロセス

図 4は提案ソフトウェアを使用した場合の特定したログタイプをコマンドライン上に表示するまでのプロセスを表している．提案ソフトウェアを使用した場合の特定したログタイプをコマンドライン上に表示するまでのプロセスを以下に記述する．

(1)　管理者が提案ソフトウェアに収集したいログのファイルパスを入力し，コマンドを実行する．

(2)　提案ソフトウェアが入力されたファイルパスのファイル内のテキストを確認する．

(3)　提案ソフトウェアが入力されたファイルパスのファイル内のテキストをもとにログタイプを特定する．

(4)　特定されたログタイプを提案ソフトウェアで出力する．

(5)　出力されたログタイプを管理者がコマンドとして入力する．

ログファイルを収集する場合，管理者が収集するログファイルを探す．次に，収集するログのファイルパスとログタイプの指定を管理者が行う．ログタイプを把握していない場合は，管理者がログタイプを調べる必要があるが，提案ソフトウェアを使用することで，管理者が入力したファイルパスのファイル内のテキストをもとにログタイプを特定し，ログタイプの特定時間を短縮できる．

4.　実装

ソフトウェアの作成には，開発言語である Python3.12.3 を使用した．開発したソフトウェアの機能と処理のプロセスについて説明する．

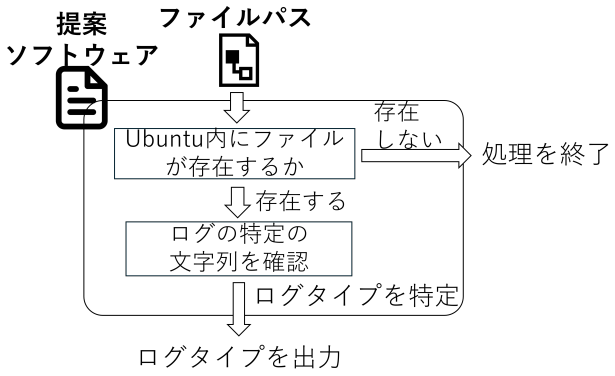

図 5　提案ソフトウェアの概要

図5は，提案ソフトウェアの概要を表している．管理者が入力したファイルパスのファイルが Ubuntu 内に存在するか確認する．提案ソフトウェアが入力されたファイルのログに定義した文字列を確認し，ログタイプを出力する．

Ubuntu 内にファイルが存在するか確認

Ubuntu 内に管理者が入力したファイルパスのファイルが存在するか確認する．管理者が入力したファイルパスを取得し，ファイルパスをもとに Ubuntu 内に入力したファイルパスのファイルが存在するか確認する．ファイルの存在が確認できた場合はログに定義した文字列を確認する処理に進み，ファイルの存在が確認できなかった場合は処理を終了する．提案ソフトウェアでは，入力されたファイルパスを変数として格納し，Ubuntu 内に入力されたファイルが存在するかを，関数を使用して確認する．入力されたファイルパスが Ubuntu 内に存在していればログに定義した文字列を確認する処理に進む．入力されたファイルパスが Ubuntu 内に存在しない場合は，「Ubuntu 内に存在するファイルパスを入力してください」とメッセージを出力し，処理を終了する．

ログに定義した文字列をもとにログタイプを判断する

Ubuntu 内に入力したファイルパスが存在した場合，ログに定義した文字列が存在するか確認する．ファイル内のログを 1 行ずつ読み込むことでログの特徴をもとにログタイプを判断する．ファイル内のログを 1 行ずつ読み込む．読み込んだ行をスペースで区切り，単語ごとに分ける．次に特定の単語があるか判別を行う．if 文を使用し，定義した文字列をもとにログタイプが Container，Syslog，Filestream かを判断する．ファイル内のすべての行に stdout または

stderr の文字列が検出された場合，ログタイプを Container とする．

ホスト名，プロセス名とプロセス ID が連続しているプロセス識別子が検出された場合，Syslog とする．すべての行に同じ単語があった場合，その単語をホスト名とする．すべての行に文字列と角括弧内に数字を含む単語があった場合，括弧内の数字をプロセス ID とし，単語の部分をプロセス名とする．すべての行にホスト名，プロセス名とプロセス ID が連続しているプロセス識別子が検出された場合，Syslog とする．Container と Syslog に定義した文字列が確認できない場合，Filestream とする．

ログタイプを出力する

ログタイプを出力する．特定したログタイプをコマンドライン上に出力する．

5. 評価実験

評価実験では，基礎実験と同じ学生 3 人を対象に行った．基礎実験で計測した提案ソフトウェア使用前のログタイプの特定時間と提案ソフトウェア使用後のログタイプの特定時間を比較し，どれだけ時間を短縮できたかで評価した．

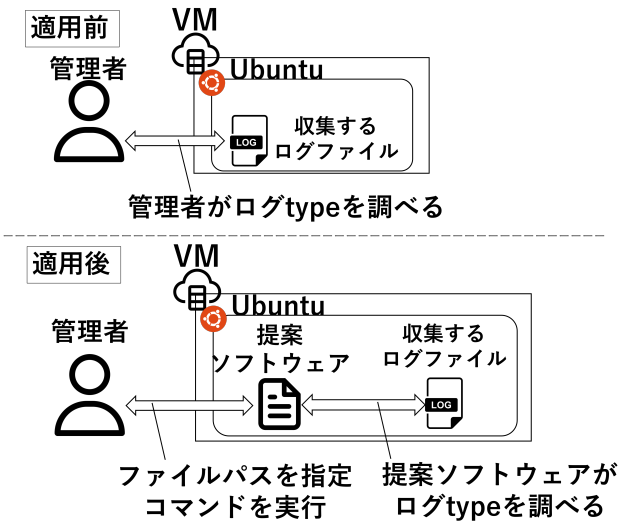

図 6 提案ソフトウェア使用前と後のログタイプ特定のプロセス

図6は提案ソフトウェア使用前と提案ソフトウェア使用後のログタイプ特定までのプロセスを表している．使用前は管理者がファイルを収集するためにコマンドを実行し，収集するファイルのログタイプを調べ，特定し Filebeat の設定ファイルに入力をする．使用後は，管理者がファイルを収集するためにコマンドを実行する．次に，収集するファイルのパスを提案ソフトウェアに入力する．提案ソフトウェアが，入力されたファイルパスにもとづいてファイルのログタイプを特定し，ログタイプを出力する．そして，管理者が出力されたログタイプを入力する．そのため，管理者がログタイプを特定する時間を短縮できる．

実験環境

評価実験は，基礎実験で述べた実験環境と同一の環境で実施した．ログタイプは 31 種類あるが，ログを収集する場合にファイルパスの指定が必要となるのは，Containers，Filestream，Google Cloud Storage，Syslog，Unix の 5 種類である．Google Cloud Storage と Unix は本稿のユースケースである Ubuntu 内のログ収集の対象外であるためログタイプの特定は，Container，Syslog，Filestream の 3 種類を対象とする．

実験結果と分析

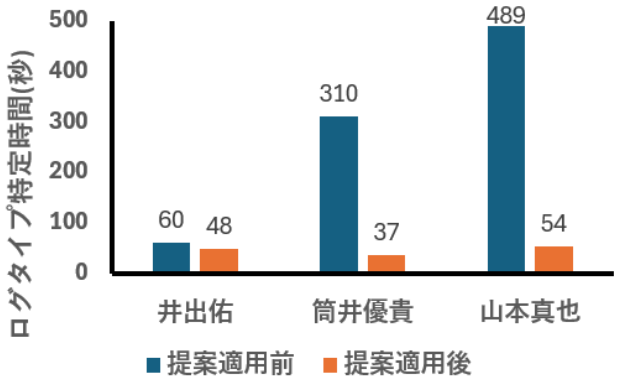

図 7 提案ソフトウェア使用前と後の計測結果

図7では学生 3 人の提案ソフトウェア使用前と使用後の 18 個のファイルのログタイプの特定時間を比較している．提案ソフトウェア使用前のログタイプの特定に要した時間は，井出佑さんが約 60 秒，筒井優貴さんが約 310 秒，山本真也さんが約 489 秒の時間を要した．提案ソフトウェア使用後のログタイプの特定に要した時間は，井出佑さんが約 48 秒，筒井優貴さんが約 37 秒，山本真也さんが約 54 秒の時間を要した．ログタイプの特定に要した時間は，井出佑さんは約 60 秒から約 48 秒で約 12 秒短縮し，約 20.0%削減した．筒井優貴さんは約 310 秒から約 37 秒で約 273 秒短縮し，約 90.0%削減した．山本真也さんは約 489 秒から約 54 秒で約 435 秒短縮し，約 90.7%削減した．提案ソフトウェア使用前と使用後でログタイプの特定時間を平均約 319 秒から平均約 46 秒に短縮し，約 85.6%削減した．提案ソフトウェアにより，ログタイプを特定したため時間を短縮することができた．提案ソフトウェア使用前と使用後のログタイプは実際のログタイプとすべて一致していた．

6. 議論

本稿の提案では，評価実験の対象者が提案ソフトウェアにファイルパスを入力し，ログタイプを Container，Syslog，Filestream の 3 種類から特定できる．しかし，入力された

ファイルパスがログのファイルパスでない場合，Filestreamと出力される．これは，情報エントロピーをもちいたログの特定で入力されたファイルパスがログファイルでない場合にFilestreamと出力されないようにすることができる．情報エントロピーをもちいることでログファイルを特定する研究がある[24]．この研究ではログファイル特定率が100%だった．本稿の提案ソフトウェアを使用する前に，この研究の提案ソフトウェアを使用することで，入力されたファイルパスがログファイルでない場合にFilestreamと出力されないようにすることができる．

7. おわりに

課題は，ログタイプの種類を把握していない場合に管理者はログタイプの種類を調べる必要があり，特定までに時間を要することである．提案手法は，管理者が入力したファイル内テキストに定義した文字列をもとに，ログタイプを特定した．すべての行にstdoutまたはstderrが検出された場合，ログタイプをContainerと特定し，すべての行にホスト名，プロセス名とプロセスIDが連続しているプロセス識別子が検出された場合，ログタイプをSyslogと特定した．ログタイプがContainerとSyslogではない場合ログタイプをFilestreamと特定した．3人の学生を評価対象とし，提案ソフトウェア使用前と使用後のログタイプの特定開始から提示されたコマンドに特定したログタイプを記入するまでの時間を評価した．提案ソフトウェアを使用せずに18個のファイルのログタイプを特定してもらい，ログタイプの特定開始から提示されたコマンドに特定したログタイプを入力するまでの時間を計測した．18個のファイルのログタイプの特定に要した3人の学生の平均時間は約319秒であり，特定したログタイプは実際のログタイプとすべて一致していた．3人の学生の提案ソフトウェア使用前と使用後の18個のファイルのログタイプの特定に要した平均時間は約319秒から約46秒で約273秒短縮し，約85.6%削減した．提案ソフトウェアを使用した場合もログタイプは実際のログタイプとすべて一致していた．

謝辞

本稿の執筆にあたり，ご助言を賜りました，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川端もものさん，山野倖平さん，宮本港斗さんに御礼申し上げます．また，基礎実験と評価実験にご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，筒井優貴さん，山本真也さんに御礼申し上げます．

参考文献

[1] Yi, S., Hu, X. and Wu, H.: An automatic reassembly model and algorithm of log file fragments based on graph theory, *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 686–689 (online), DOI: 10.1109/ICSESS.2015.7339150 (2015).

[2] Liu, Z., Zhang, X., Li, G., Cui, H., Wang, J. and Xiao, B.: A Secure and Reliable Blockchain-based Audit Log System, *ICC 2024 - IEEE International Conference on Communications*, pp. 2010–2015 (online), DOI: 10.1109/ICC51166.2024.10623012 (2024).

[3] Cao, Q., Qiao, Y. and Lyu, Z.: Machine learning to detect anomalies in web log analysis, *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 519–523 (online), DOI: 10.1109/CompComm.2017.8322600 (2017).

[4] Etoh, F., Takahashi, K., Hori, Y. and Sakurai, K.: Study of Log File Dispersion Management Method, *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, pp. 371–374 (online), DOI: 10.1109/SAINT.2010.104 (2010).

[5] Nagappan, M. and Vouk, M. A.: Abstracting log lines to log event types for mining software system logs, *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pp. 114–117 (online), DOI: 10.1109/MSR.2010.5463281 (2010).

[6] Teixeira, C., de Vasconcelos, J. B. and Pestana, G.: A knowledge management system for analysis of organisational log files, *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–4 (online), DOI: 10.23919/CISTI.2018.8399229 (2018).

[7] Nagappan, M.: Analysis of execution log files, *2010 ACM/IEEE 32nd International Conference on Software Engineering*, Vol. 2, pp. 409–412 (online), DOI: 10.1145/1810295.1810405 (2010).

[8] Chen, P., Chao, G., Yang, L., He, H., Hong, L., Li, M., Gao, D. and Guo, S.: A Robust Log Parsing Algorithm—Practice of Logslaw in Heterogeneous Logs of Pacific Credit Card Center of Bank of Communications(PCCC), *2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA)*, pp. 126–133（オンライン），DOI: 10.1109/ICIPCA59209.2023.10257676 (2023).

[9] Nguyen, H.-T., Nguyen, L.-V., Le, V.-H., Zhang, H. and Le, M.-T.: Efficient Log-based Anomaly Detection with Knowledge Distillation, *2024 IEEE International Conference on Web Services (ICWS)*, pp. 578–589 (online), DOI: 10.1109/ICWS62655.2024.00078 (2024).

[10] Fu, Q., Lou, J.-G., Wang, Y. and Li, J.: Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis, *2009 Ninth IEEE International Conference on Data Mining*, pp. 149–158 (online), DOI: 10.1109/ICDM.2009.60 (2009).

[11] Son, S. J. and Kwon, Y.: Performance of ELK stack and commercial system in security log analysis, *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, pp. 187–190 (online), DOI: 10.1109/MICC.2017.8311756 (2017).

[12] Prakash, T., Kakkar, M. and Patel, K.: Geo-identification of web users through logs using ELK stack, *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pp. 606–610 (online), DOI: 10.1109/CONFLUENCE.2016.7508191 (2016).

[13] Rochim, A. F., Aziz, M. A. and Fauzi, A.: Design Log Management System of Computer Network Devices Infrastructures Based on ELK Stack, *2019 International Conference on Electrical Engineering and Computer Sci-*

ence (ICECOS), pp. 338–342 (online), DOI: 10.1109/ICECOS47637.2019.8984494 (2019).

[14] Langi, P. P. I., Widyawan, Najib, W. and Aji, T. B.: An evaluation of Twitter river and Logstash performances as elasticsearch inputs for social media analysis of Twitter, *2015 International Conference on Information & Communication Technology and Systems (ICTS)*, pp. 181–186 (online), DOI: 10.1109/ICTS.2015.7379895 (2015).

[15] Wei, B., Dai, J., Deng, L. and Huang, H.: An Optimization Method for Elasticsearch Index Shard Number, *2020 16th International Conference on Computational Intelligence and Security (CIS)*, pp. 191–195 (online), DOI: 10.1109/CIS52066.2020.00048 (2020).

[16] Bajer, M.: Building an IoT Data Hub with Elasticsearch, Logstash and Kibana, *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pp. 63–68 (online), DOI: 10.1109/FiCloudW.2017.101 (2017).

[17] Ahmed, F., Jahangir, U., Rahim, H., Ali, K. and Agha, D.-e.-S.: Centralized Log Management Using Elasticsearch, Logstash and Kibana, *2020 International Conference on Information Science and Communication Technology (ICISCT)*, pp. 1–7 (online), DOI: 10.1109/ICISCT49550.2020.9080053 (2020).

[18] Bhatnagar, D., SubaLakshmi, R. J. and Vanmathi, C.: Twitter Sentiment Analysis Using Elasticsearch, LOGSTASH And KIBANA, *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pp. 1–5 (online), DOI: 10.1109/ic-ETITE47903.2020.351 (2020).

[19] Divya, P. J., George, R. S., Madhusudhan, G. and Padmasree, S.: Organization-wide IOC Monitoring and Security Compliance in Endpoints using Open Source Tools, *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, pp. 1–6 (online), DOI: 10.1109/GCAT55367.2022.9971999 (2022).

[20] Lertwuthikarn, T., Barroso, V. C. and Akkarajitsakul, K.: Resource Optimization for Log Shipper and Preprocessing Pipeline in a Large-Scale Logging System, *2022 IEEE 5th International Conference on Knowledge Innovation and Invention (ICKII)*, pp. 196–200 (online), DOI: 10.1109/ICKII55100.2022.9983590 (2022).

[21] Locke, S., Li, H., Chen, T.-H. P., Shang, W. and Liu, W.: LogAssist: Assisting Log Analysis Through Log Summarization, *IEEE Transactions on Software Engineering*, Vol. 48, No. 9, pp. 3227–3241 (online), DOI: 10.1109/TSE.2021.3083715 (2022).

[22] Messaoudi, S., Panichella, A., Bianculli, D., Briand, L. and Sasnauskas, R.: A Search-Based Approach for Accurate Identification of Log Message Formats, *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pp. 167–16710 (2018).

[23] Nagappan, M. and Vouk, M. A.: Abstracting log lines to log event types for mining software system logs, *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pp. 114–117 (online), DOI: 10.1109/MSR.2010.5463281 (2010).

[24] Mikami, S.: .