# ファイルにおける行ごとの最頻文字数の一致率を用いたログの特定

三上 翔太[1]　高橋 風太[2]　串田 高幸[1]

**概要**：システムに障害が発生した際，管理者はログを検索し，原因の特定を行う．ログを検索する際，Elasticsearch を使用する場合があり，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 Kibana で可視化する．その際，Filebeat では収集対象のログファイルのディレクトリ指定を行う必要があり，それらは管理者が行う．その場合，VM 内の多数あるファイルの中からすべてのログファイルの収集に時間がかかる．そこで本稿ではログファイルが VM 内のどこにあるのかを自動で識別する手法を提案する．本稿の提案手法では，ファイルを文字単位で分析し，最も頻出する文字をカウントして 1 行あたりの最頻文字の出現数の一致率が仮に 10 %以上ならログファイルと識別する．基礎実験では，Ubuntu 24.04，K3s vl.29.5+k3s1，Helm v3.3.15.2，WordPress 6.5.4，Python 3.12 の仮想環境がインストール VM 内の全ファイルの読み込みに要する時間を測定した．実験の結果，226858 件のファイルの読み込みに約 1141.67 秒の時間を要した．評価実験では，VM 内のログファイルを探すために要した時間を評価する．

## 1. はじめに

### 背景

東京工科大学の Cloud and Distributed Systems Laboratory(CDSL) には所属する学生が使用できる Kubernetes 共有クラスター (unikube) がある．Kubernetes とはデプロイやスケーリングを自動化したり，コンテナ化されたアプリケーションを管理するためのオーケストレーションツールである [1–3]．Kubernetes で作成可能なリソースの 1 つに Pod がある．Pod とは Kubernetes における最小のデプロイ可能な計算単位であり，1 つ以上の関連したコンテナのグループを表す [4]．unikube の管理者は学生であり，その学生が管理を行っている．管理の一環として Pod の状態の確認を行っている．その際，STATUS が Running ではない Pod と READY が 1/1，2/2，3/3 ではない Pod がなぜそのような状態になっているのかをその Pod のログを検索，閲覧して調査している．

ログは，システム操作の重要な動作を記録するもので，システムの健全性状態を検出するための優れた情報源である [5,6]．ログには，システムの動作の記録のほかにタイムスタンプが記録されている [7]．ログ分析を通じて，システムの異常検出と障害特定を行い，障害が見つかった後に障害の原因を見つけることができる [8–10]．ログを検索する際，Elasticsearch を使用する場合がある．Elasticsearch とは，ビックデータ検索ツールでログデータの検索も行うことができる分散型の全文検索エンジンである [11–13]．Elasticsearch でログを扱う際，Filebeat でログファイルの収集を行う [14]．Filebeat とはサーバ上のログファイルやその他のファイルを監視し，それらのデータを収集して Elasticsearch や Logstash に転送するログシッパーである [15]．Filebeat は Kubernetes 環境で使用できるので，Kubernetes がインストールされていないシステムからログを収集する際，そのシステムのディレクトリやファイルを別のシステムに同期する必要がある．

Filebeat では収集対象のログファイルのディレクトリ指定は管理者が手動で行う必要がある．その際，システムすべてのログを収集できていない場合がある．

Filebeat で収集したログは，Logstash へ送信される．Logstash とは，複数のソースから同時にデータを取り込み，変換，出力するツールである [16]．Logstash で変換されたログを Elasticsearch で検索している．また，収集したログを可視化する際，Kibana を用いる．Kibana とは，データを可視化，分析するツールである．Kibana では時系列でログを扱うことができるため，タイムスタンプを有するログを扱うことが多い [17]．Kibana を用いることで，

[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
〒 192–0982 東京都八王子市片倉町 1404-1

文字列であるログをグラフ化することができる．

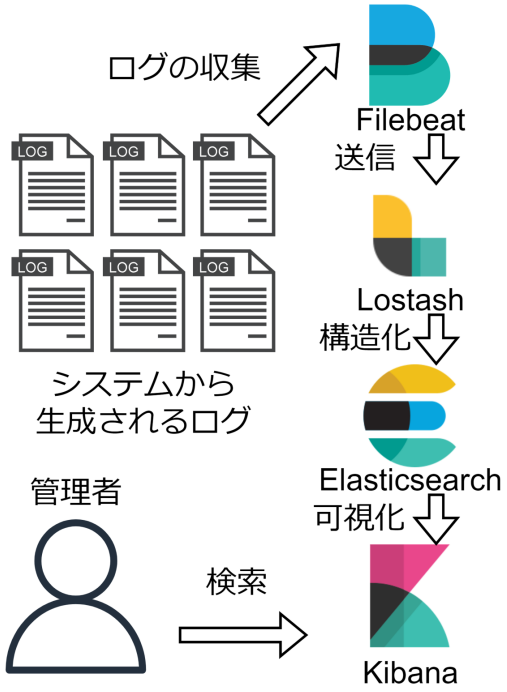


**図 1**　ログを検索するまでの流れ

図 1 は，ログを収集してから検索するまでの示したものである．Filebeat で収集したログを Logstash で扱いやすい形に変換し，Elasticsearch で検索，Kibana で可視化する流れで管理者がログを扱う．

**課題**

課題はログファイルかどうか判断して収集する必要があるが，どれがログファイルか判断する時間がかかることである．

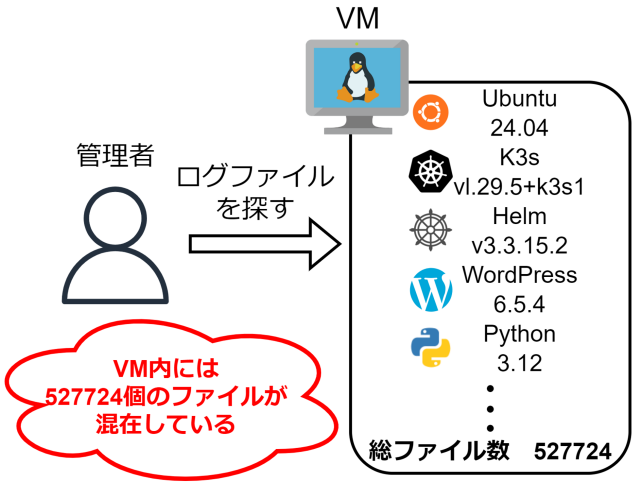


**図 2**　VM 内からログファイルを探す際の流れ

図 2 は，Ubuntu24.04，K3s vl.29.5+k3s1，Helm v3.15.2，WordPress6.5.3，Python3.12 の仮想環境をインストールした仮想マシン (VM) からログファイルを探す流れを表している．図 2 の VM は，ファイル数が 527724，ディレクトリ数が 63801，データ量が 12GB であり，その中からログファイルを探すために時間を要する．例えば，VM 内の/var/log ディレクトリ内の 79 個のファイルから 61 個のログファイルを探すために 1 時間 9 分 10 秒の時間を要した．

**各章の概要**

第 2 章では，本稿の関連研究について述べる．第 3 章では，本稿の課題について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

ログパーサーの性能を評価し，それらがシステムの異常検出に及ぼす影響について分析している論文がある [7]．5 つの大規模ログデータセットを用いて精度，効率性，異常検知タスクへの影響を包括的に評価した．評価結果から現状のログパーサーの長所や短所，ログ前処理の重要性，スラスタリングベースの手法のスケーラビリティ課題の知見を得た．この論文はデータ駆動型のログパーサーを用いたログ解析についての研究なのでファイルの中身がログファイルかどうかの特定は行っていない．

セキュリティログを分析し，異常を迅速に特定するための知的ログ分析システム MMSLAS(Massive and Multi-Source Security Log Analysis System) を提案している論文がある [18]．MMSLAS は分散アーキテクチャを採用し，大量のログデータを処理している．システムの中核となるログ分析モジュールでは，ビジネスツール分析，行動分析，機械学習ベースの分析という 3 つのアプローチを結合している．これにより，複数のログデータからの情報を相関分析し，セキュリティ脅威を検出することが可能となった．この論文ではビジネスツール分析，行動分析，および機械学習にもとづく分析を組み合わせることでログ解析を実現している．本稿では，ファイルの中身の分析を行うがこの論文ではファイルの中身の分析を行っていない．

Linux OS イベントログの自動解析にもとづいてセキュリティイベントを検出する方法を提案している論文がある [19]．提案手法は，イベントログを単一のデータ配列に読み込み，クラスタリングアルゴリズムで構造が類似したログをグループ化し，正規表現を用いてクラスタ化させたイベントを分類してセキュリティイベントを特定するという流れになっている．この論文ではイベントログのみとなっており，すべてのログの検出を行っていない．

正規表現にもとづくログ解析システムの構築に関する

研究を行っている論文がある [20]．この論文では，ログフォーマットの記述とログ内容の解析を分離するアプローチを採用している．XML を使用してログフォーマットの記述とログデータ項目の属性記述のためのスクリプトドキュメントを作成し，これをログ解析の設定ファイルとして使用している．スクリプトはログタイプに対応し，特定のパーサーから分離されている．Apache ログを例にプロトタイプシステムを開発した．この論文は正規表現を用いたログ解析システムの構築に焦点を当てているが，ログファイルであることが前提となっているのでログファイルであるかどうかについての解析は行っていない．

## 3. 提案方式

本稿では，ログファイルに見られる規則性を利用してログファイルを識別する手法を提案する．ここでの規則性とはファイル内の一行での最頻文字数のファイル全体の一致率である．本稿の提案方式は，主にデータ前処理・変換フェーズと頻度分析フェーズから構成される．

### データ前処理・変換フェーズ

このフェーズではファイルの中身の規則性を見つけるフェーズのために，ファイルの中身を扱いやすい形に変換する．最初にファイル内で 1 番多い文字の種類の検出を行う．ファイルの中身を読み取って文字の種類ごとに個数をカウントして数字と文字列以外で一番多かったものを最頻文字とする．

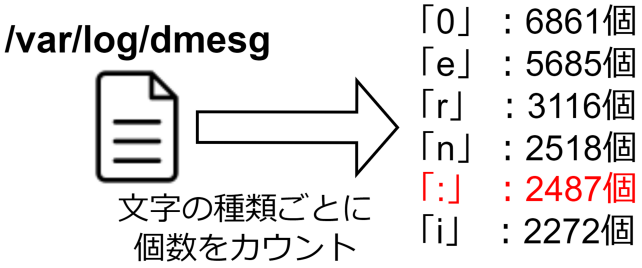


**図 3** 最頻文字の検出

図 3 は，最頻文字を検出する流れを示したものである．例として/var/log/dmesg を挙げる．ファイルの中身を 1 文字ずつ読み込み，文字ごとにカウントする．同じ文字があればカウントを増やす．例として図 3 では，数字と文字以外で”:”が一番多かったため，”:”が最頻文字となる．また，ログファイル内は英語で表記されており単語ごとに空白があるため，空白も最頻文字の対象から除外した．

最頻文字を検出できたら，次に最頻文字のカウントを行う．

図 4 は，最頻文字カウントの流れを示したものである．先ほど検出した最頻文字を用いる．最頻文字を識別したらカウントする．例として図 4 では，ファイルの中身の 3 行

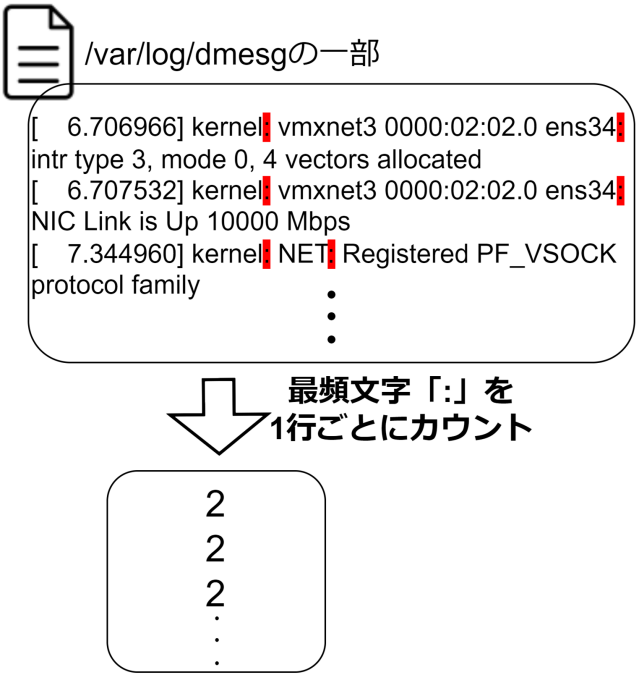


**図 4** 最頻文字のカウント

における内容と文字数は異なるが，最頻文字のカウント数は一致している．

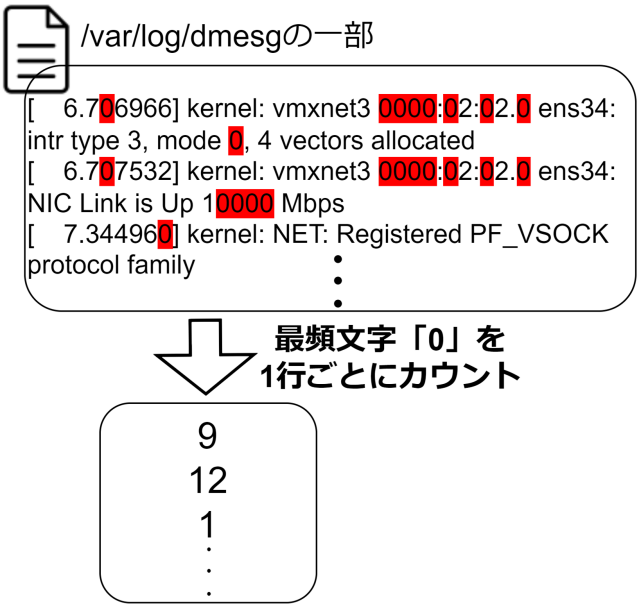


**図 5** 数字を最頻文字とした場合

図 5 は数値を最頻文字の対象とした場合である．ログの内容は図 4 と同じである．内容が同じであるにも関わらず，各行の最頻文字のカウントを行った際，異なる最頻文字数となったので数値は最頻文字の対象から外した．

図 6 は文字を最頻文字の対象とした場合である．ログの内容は図 4 と同じである．内容が同じにも関わらず，各行の最頻文字のカウントを行った際，異なる最頻文字数となったので数値は最頻文字の対象から外した．

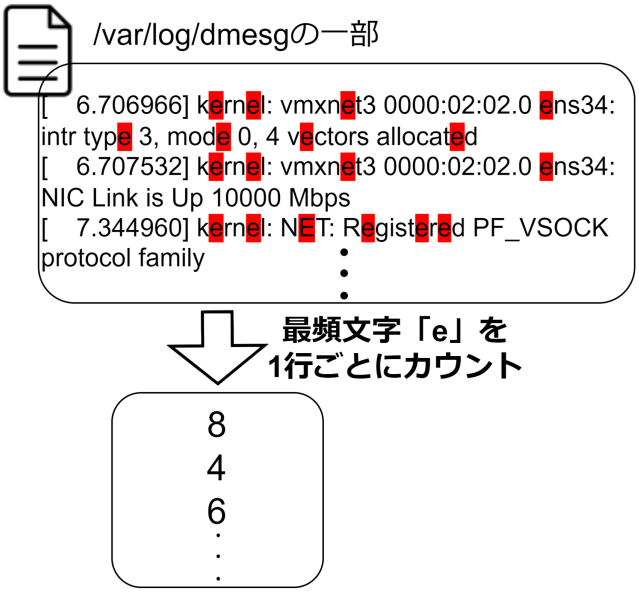


**図 6** 文字を最頻文字とした場合

### 頻度分析フェーズ

このフェーズではデータ前処理・変換フェーズで変換させたものを用いて規則性を見つける．規則性を見つける際は，ファイルの中身の最頻文字の各行の頻出数が仮にファイル全体の 10 %以上同じならばログファイルとする．

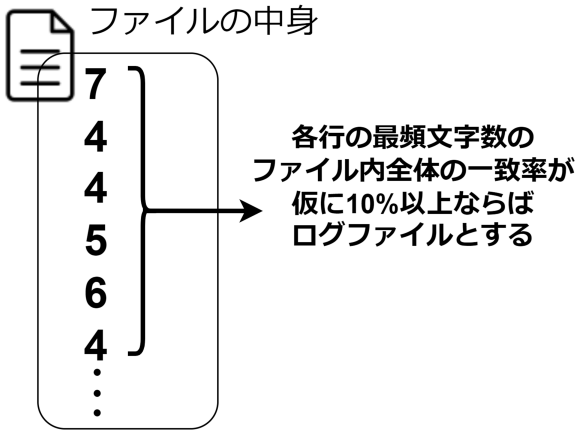


**図 7** 複数行での比較

図 7 は, 前行との比較で規則性を見つける際の流れを示したものである. データ前処理・変換フェーズで変換された中身を分析し, 各行の最頻文字数のファイル内全体一致率が 10 %以上ならば仮にログファイルとする.

図 8 は，Ubuntu24.04 がインストールされている VM の/var/log ディレクトリにある 108 個のファイルの各行の最頻文字数のファイル内全体一致率とログファイルである確率を表している．各行の最頻文字数一致率が 0〜10 %のログファイル率が 30 %と各行の最頻文字数一致率 20〜100 %時のログファイル率より低いので各行の最頻文字数のファイル内全体一致率が 10 %以上という値を仮に定めた.

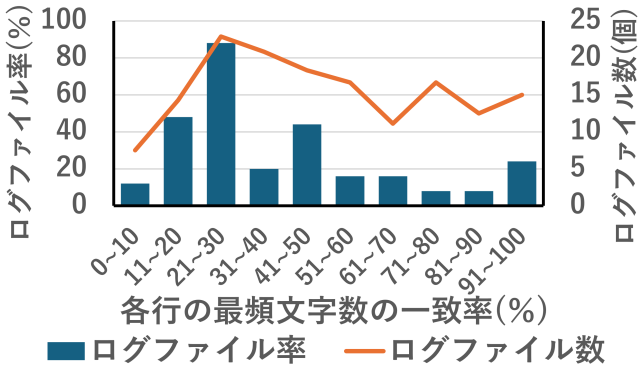


**図 8** 最頻文字数の一致率ごとのログファイル数とログファイル率

### ユースケース・シナリオ

本稿では，unikube のログファイルを収集するユースケースとする．ログの管理は Elasticsearch で行っている．その際，VM からのログの収集では Filebeat を使用するため，Filebeat でのディレクトリの指定が必要になる．その際，ログファイルのディレクトリは管理者が特定する．そこで提案ソフトウェアを用いてログファイルのディレクトリを特定する．特定したディレクトリを管理者が Filebeat で指定し，収集する．収集したログは Elasticsearch に集約され，Kibana を通じて可視化，分析される．

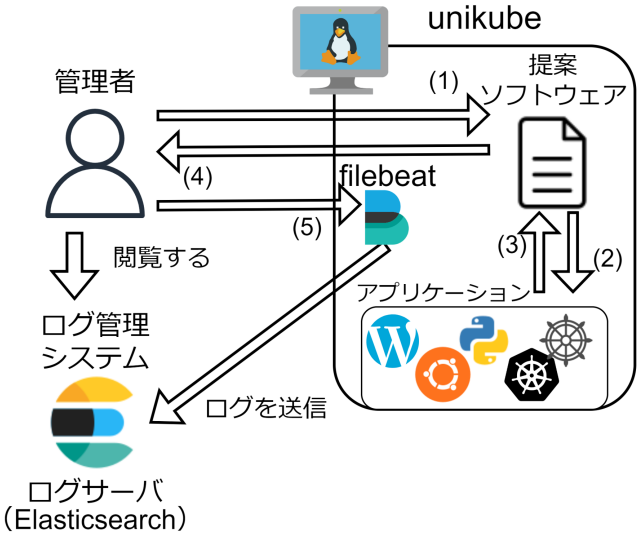


**図 9** ログを収集する際の流れ

図 9 は unikube のログを Elasticsearch に送信するまでの流れを示したものである．まず，(1) で管理者がログファイル収集先で提案ソフトウェアを実行する．(2) で提案ソフトウェアがログファイルの特定を行っている．(3) で提案ソフトウェアの結果が出力され，(4) で管理者に結果が表示される．(5) で提案ソフトウェアの結果をもとに管理者が Filebeat にて管理者がログファイルのディレクトリの指定を行う.

## 4. 実装

実装では提案手法をもとに, ソフトウェアを作成した. ソフトウェアの作成には，開発言語である Python3.9 を用いた．開発したソフトウェア (L-Scan) 内のファイルがどのような機能を持っているのか説明する.

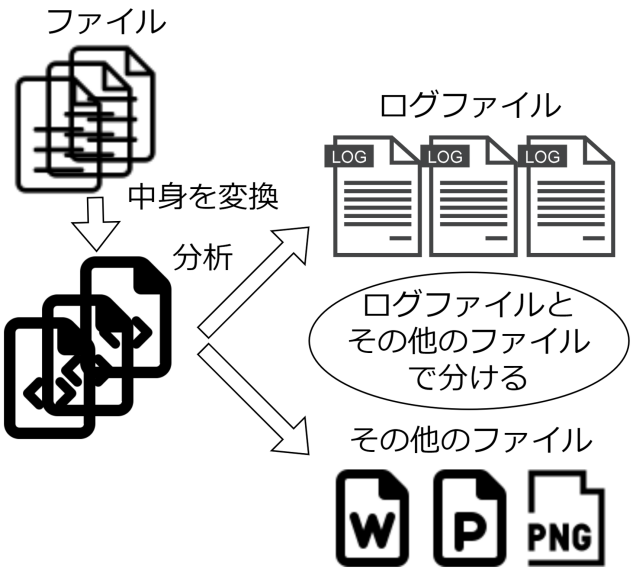


**図 10** L-Scan の概要

図 10 は，提案ソフトウェアの概要を示したものである．ソフトウェアは 3 つの機能から成り立っている．ファイルの読み込み，ファイル内の変換，ファイル内の分析の順で実行される.

### ファイルの読み込み

指定されたルートディレクトリから開始し，すべてのサブディレクトリとファイルを読み込む．プログラムが root 権限で実行されていることを確認する．そうすることで権限により読み込みができなくなることを防いでいる．次に，特定のディレクトリ（例：/proc，/dev，/sys，/run）を読み込み対象から除外する．カスタム例外クラス（TimeoutException）により，タイムアウト発生時に処理を行う．読み込みの際には各ファイルの 1 文字 1 文字を読み込んで最頻文字の特定を行う．数字と文字は最頻文字の対象外となっている.

### ファイル内の変換

次に，ファイル内のデータを分析しやすい形に変換する．この変換では，先ほど特定した最頻文字を用いる．各行ごとに最頻文字の出現回数をカウントし，そのカウントされた数値を用いて，元の文字列を置き換える．各行の文字列がその行の最頻文字の出現回数にもとづいて数値に変換されることで，各行の最頻文字の数がわかる．最終的にファイル内すべてが数値のみとなる.

### ファイル内の分析

最後に，変換されたファイルの内容を分析する．この分析は，ファイル内の各行における最頻文字の出現回数を集計し，その集計結果をもとに行う．まず，各行ごとに最頻文字の出現回数を記録し，それをもとにファイル内各行の最頻文字の出現回数を比較する．この比較により，各行の最頻文字の出現回数が明確になる．次に，その集計結果を解析し，1 番多かった頻出数がファイル全体の行数の 10 % 以上の場合，仮にそのファイルをログファイルとして識別する.

## 5. 評価実験

### 基礎実験

基礎実験では VM 内の全ファイルの読み込みに要する時間を計測した.

**表 1** 除外したディレクトリ情報一覧

| ディレクトリ | ファイル数 | データ量 |
|---|---|---|
| proc | 104,025 | 1.14 GB |
| dev | 209 | 0.00 B |
| sys | 69,310 | 399.28 MB |
| run | 33,139 | 1.34 GB |

表 1 は，基礎実験の際に除外したディレクトリをまとめたものである．proc，dev，sys，run ディレクトリには特殊なファイルが含まれているため排除した．proc，dev は仮想ファイルシステムであり，実際のファイルではない．sys はカーネルとデバイスの仮想ファイルシステムで，システム設定や詳細を提供する．run は実行時の一時データを保存し，システムの稼働中にのみ存在する揮発性のデータを含む．これらのディレクトリは，実験の一貫性を確保するために計測対象から除外した.

### 実験環境

Ubuntu24.04 をインストールした VM を用意した．下記に VM のスペックを記載する.

- VM のスペック
  - CPU：3core
  - メモリ：8GB
  - ストレージ：25GB

VM 内には，Ubuntu 24.04，K3s vl.29.5+k3s1，Helm v3.3.15.2，WordPress 6.5.4，Python 3.12 の仮想環境がインストールされている.

### 基礎実験の結果

下記に実験結果を記載する．

- 実験結果
  - 読み込んだディレクトリ数：36494
  - 読み込んだファイル数：226858
  - 要した時間：約 1141.67 秒 (約 19 分 1.67 秒)

結果として，VM 内の全ファイルの読み込みに約 1141.67 秒の時間を要した．

### 評価実験計画

評価実験について記述する．評価実験では，unikube にて行う．内容としては unikube 内で学生にアプリケーションをインストールしてもらい，そのアプリケーションに関するログファイルを探す．その際，ログファイルを探すために要した時間を評価する．

## 6. 議論

本稿では基礎実験で VM 内の全ファイルの読み込みを行った．その際，約 19 分 1.67 秒の時間を要した．ログファイルを探す度に約 19 分 1.67 秒の時間がかかってしまう．また，環境次第ではそれ以上の時間を要することになる．これは，Python の threading モジュールを使用した並列処理により解決できる．threading モジュールを使用することで使用可能な CPUcore 数に応じて処理速度の向上を可能とする．

本稿では図 8 のデータにおいて数値が 1 番低いという理由で最頻文字数の一致率が 10 %以上ならばログファイルと定めた．しかし，データを 10 %ごとに分けた根拠がない．より細かくデータを分けることで提案手法の精度が上がるが，分割数は小数点以下もできるため，どこまで細かくするべきかの基準がない．したがって，実験可能な限りデータの分割を行う．

本稿では/var/log ディレクトリの 108 個のファイルのみを対象として実験を行った．しかし，他のディレクトリでもログファイルを特定できるとは限らない．これは，他のディレクトリのログに共通する特徴を探すことでログファイルを特定できる．いくつかのログに共通している特徴の例としてタイムスタンプがある．正規表現を用いたパターンマッチングによりあらかじめタイムスタンプの定義を作成し，ファイル内の文字列と比較することでタイムスタンプを特定できる．

本稿の図 8 のデータは Ubuntu24.04 をインストールした直後の VM から抽出した．しかし，インストール直後と一定期間使用した VM ではログの割合が異なる．これは，作成してから一定期間使用している VM と作成直後の VM で実験を行うことでログの割合が異なる環境でのデータ比較をすることができる．実験として，3 カ月使用した VM との比較を予定している．

## 7. おわりに

システムに障害が発生した際，管理者はログを検索し，原因の特定を行う．ログを検索する際，Elasticsearch を使用する場合があり，ログファイルを収集するために Filebeat を使用する．その際，Filebeat では収集対象のログファイルのディレクトリ指定を管理者が行う．その場合，システム内のすべてのログファイルの収集に時間がかかる場合がある．そこで本稿ではログファイルがシステム内のどこにあるのかを自動で識別する提案をする．本稿の提案手法では，ファイルを文字単位で分析し，最も頻出する文字をカウントして 1 行あたりの最頻文字の出現数の一致率が 10 % 以上なら仮にログファイルと識別する．課題に対する基礎実験として VM 内の全ファイルの読み込みに要する時間を測定した．結果は，226858 件のファイルの読み込みに約 1141.67 秒の時間を要した．評価実験では，VM 内のログファイルを探すために要した時間を評価する．

**謝辞** 本稿の執筆に当たりご助言を賜りました, 株式会社ライフコーポレーションの大野 有樹さん，東京工科大学院バイオ・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 真斗さんに御礼申し上げます．

### 参考文献

[1] Malviya, A. and Dwivedi, R. K.: A Comparative Analysis of Container Orchestration Tools in Cloud Computing, *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 698–703 (online), DOI: 10.23919/INDIACom54597.2022.9763171 (2022).

[2] Park, J., Choi, U., Kum, S., Moon, J. and Lee, K.: Accelerator-Aware Kubernetes Scheduler for DNN Tasks on Edge Computing Environment, *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 438–440 (online), DOI: 10.1145/3453142.3491411 (2021).

[3] Islam Shamim, M. S., Ahamed Bhuiyan, F. and Rahman, A.: XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices, *2020 IEEE Secure Development (SecDev)*, pp. 58–64 (online), DOI: 10.1109/SecDev45635.2020.00025 (2020).

[4] Zhu, M., Kang, R., He, F. and Oki, E.: Implementation of Backup Resource Management Controller for Reliable Function Allocation in Kubernetes, *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pp. 360–362 (online), DOI: 10.1109/NetSoft51509.2021.9492724 (2021).

[5] Kamal, A., Hajamydeen, A. I. and Amril Jaharadak, A.: Log Necropsy: Web-Based Log Analysis Tool, *2022 IEEE 10th Conference on Systems, Process Control (ICSPC)*, pp. 176–179 (online), DOI: 10.1109/ICSPC55597.2022.10001797 (2022).

[6] Wang, Y.: Design of Visual Log Analysis System, *2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE)*, pp. 1649–1652 (online), DOI: 10.1109/IC-

 

SECE58870.2023.10263397 (2023).

[7] He, P., Zhu, J., He, S., Li, J. and Lyu, M. R.: An Evaluation Study on Log Parsing and Its Use in Log Mining, *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 654–661 (online), DOI: 10.1109/DSN.2016.66 (2016).

[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 (online), DOI: 10.1109/ICIPCA59209.2023.10257676 (2023).

[9] 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).

[10] Rastogi, R., Akash, S., Shobha, G., Poonam, G., Pratiba, D. and Singh, A.: Design and development of generic web based framework for log analysis, *2016 IEEE Region 10 Conference (TENCON)*, pp. 232–236 (online), DOI: 10.1109/TENCON.2016.7847996 (2016).

[11] Thacker, U., Pandey, M. and Rautaray, S. S.: Performance of elasticsearch in cloud environment with nGram and non-nGram indexing, *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 3624–3628 (online), DOI: 10.1109/ICEEOT.2016.7755381 (2016).

[12] 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).

[13] Bendechache, M., Svorobej, S., Endo, P. T., Mario, M. N., Ares, M. E., Byrne, J. and Lynn, T.: Modelling and Simulation of ElasticSearch using CloudSim, *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pp. 1–8 (online), DOI: 10.1109/DS-RT47707.2019.8958653 (2019).

[14] 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).

[15] 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).

[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] 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).

[18] Sun, Y., Guo, S. and Chen, Z.: Intelligent Log Analysis System for Massive and Multi-Source Security Logs: MMSLAS Design and Implementation Plan, *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pp. 416–421 (online), DOI: 10.1109/MSN48538.2019.00085 (2019).

[19] Stepanenko, D. V., Stoychin, K. L. and Shevchenko, D. V.: Analysis of Operating System Event Logs when Investigating Information Security Incidents, *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pp. 313–315 (online), DOI: 10.1109/USBEREIT58508.2023.10158875 (2023).

[20] Zhang, J. and Wang, F.: Research on the construction of log parsing system based on regular expression, *2022 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, pp. 627–630 (online), DOI: 10.1109/ICITBS55627.2022.00137 (2022).

 