タイムスタンプとコンテナ名を用いたログ集計による障害原因の調査時間の短縮

岡田 京太郎[1]　小山 智之[2]　串田 高幸[1]

概要：ログはシステムの実行記録であり，開発や保守に使用される．ログファイルは Filebeat で収集され，Logstash によって構造化される．構造化されたログは Elasticsearch に保存され，Kibana で可視化される．ログファイルの収集に Filebeat を使用する．収集したログは Logstash で構造化される．構造化されたログは Elasticsearch で保存され，ログは Kibana で可視化する．システムで障害が起きた際，管理者は Kibana でログを検索し，原因調査を行う．2025 年 3 月 20 日から 2025 年 4 月 20 日の期間に CDSL のログサーバでログが約 140 万件出ていた．ログの検索をする際，約 140 万件の中から障害の原因に関係するログを見つけるため，クエリ検索や時間の絞り込みを行う必要がある．課題は，障害が発生してログを調査する際にログの検索クエリの作成にスキルやログのメッセージを理解するための専門知識が必要である．そのため，スキルや経験が不足していると調査による原因の特定や解決に時間がかかることである．基礎実験では STNS の障害の原因調査でログの調査にかかる時間を計測した．障害の原因調査にかかった時間の合計は約 238 分であり，ログの調査に約 41 分であった．これは，全体の約 17.2%であり，ログの調査にかかった時間は，全体の時間のうち 2 番目に時間がかかっていた．提案手法は，Elasticsearch のログのタイムスタンプをもとに過去 3 時間分のログを 15 分ごとに件数で集計する．15 分ごとのログ件数から中央値を求め，最新の 15 分のログ件数と比較を行う．中央値より最新の時間帯のログ件数が多い場合，異常有りと判断することで障害に関連するコンテナ名を特定する．評価実験では，STNS の障害と Rook-Ceph で起きた障害の 2 種類のシナリオを対象に，管理者が特定した原因と提案ソフトウェアで特定した原因を比較し，各シナリオにおいて障害に関連するコンテナ名を検出できたかの検出率で評価する．

1. はじめに

背景

ログとはシステムにおける実行の記録であり，システムの開発や保守で使用される [1]．例えば，障害が起きた際に管理者は障害発生の時間帯に取得したログを確認して何が原因で障害が起きたかを調査する．ログは ELK Stack と Filebeat を使用して収集を行う．ELK Stack とは，Elasticsearch，Logstash，Kibana の 3 つのオープンソースプロジェクトを組み合わせたものである [2]．ログの構造化や解析を Logstash，ログの検索や保存を Elasticsearch，ログの可視化を Kibana，ログの収集を Filebeat で行う．Logstash とは，ログの収集や解析を行い，Elasticsearch に保存するためのオープンソフトウェアである [3]．Elasticsearch とは，ソーシャルメディア分析を行うことができるツールで，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 [4]．Kibana とは，データの分析，表示，検索に使用できるオープンソースの分析ツールである [5]．Filebeat とは，サーバ側の前処理パイプラインである Logstash にデータを転送するためのログシッパーである [6]．

図 1 は，踏み台サーバにログインするまでの過程を表す．図 1 は，クライアント，GitHub，core server，Pod，git-sync，builder-assets-volume，builder，config-volume，STNS，踏み台サーバ，ログファイル，CDSL の学生で構成される．CDSL とは，東京工科大学のクラウド・分散システム研究室のことであり，以後 CDSL とする．クライアントは，GitHub に公開鍵を登録する際に使用するコンピュータを表す．GitHub は，公開鍵を登録するためのウェブサービスを表す．core server は，Pod やログファイルが動作するサーバを表す．Pod は，コンテナやストレージが動作するための実行環境を表す．git-sync は，core server の Pod 内にあるコンテナを表す．builder-assets-volume は，core server の Pod 内にあるストレージを表す．builder は，core server の Pod 内にあるコンテナを表す．config-volume は，

[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
[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

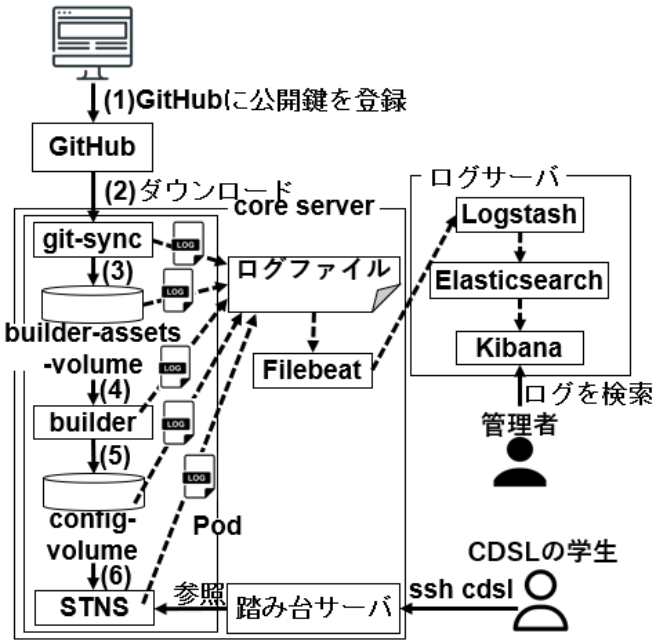

図 1　踏み台サーバにログインするまでの過程

core server の Pod 内にあるストレージを表す．STNS は，core server の Pod 内にあるコンテナを表す．踏み台サーバは，STNS サーバにログインするための中継サーバを表す．ログファイルは，core server の Pod 内にあるコンテナやストレージのログをファイル形式で記録していることを表す．CDSL の学生は，踏み台サーバにログインする人を表す．(1) でクライアントから公開鍵を GitHub に登録する．(2) で GitHub に登録された公開鍵が core server にある git-sync にダウンロードする．(3) で git-sync にダウンロードされた公開鍵は builder-assets-volume にダウンロードする．(4) で builder-assets-volume にダウンロードされた公開鍵が builder にダウンロードする．(5) で builder にダウンロードされた公開鍵が config-volume にダウンロードする．(6) で config-volume にダウンロードされた公開鍵が STNS にダウンロードされ，管理する．(7) で Pod 内のコンテナとストレージのログがログファイルに記録する．(8) で CDSL の学生が踏み台サーバにログインをする．

課題

課題は，管理者が障害の原因調査でログを確認する場合，ログの検索クエリの作成にスキルやログのメッセージを理解するための専門知識が必要であり，スキルや経験が不足していると調査による原因の特定や解決に時間がかかることである．例えば，2025 年 3 月 20 日から 2025 年 4 月 20 日の期間に CDSL のログサーバでログが約 140 万件出ていた．ログの検索をする際，約 140 万件の中から障害の原因に関係するログを見つけるため，クエリ検索や時間の絞り込みを行う必要がある．管理者が，ログの検索クエリの作成ができ，スキルやログのメッセージを理解できた場合，瞬時に原因調査に取り組むことができる．しかし，検査クエリが何かわからない場合，クエリを検索する時間が生じる．また，ログの内容を理解できなかった場合，生成 AI や Web サイトを使用してログの内容を調べる必要がある．

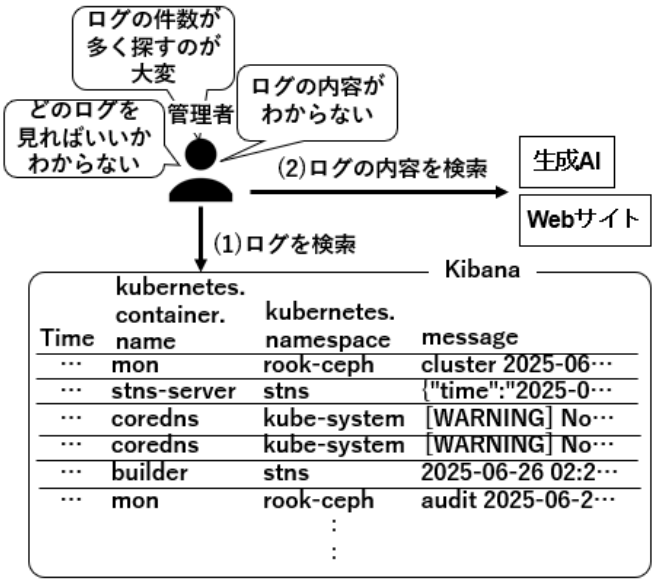

図 2　ログで原因調査を行う際の過程

図 2 は，ログで原因調査を行う際の過程を表す．図 2 は，管理者，Kibana，生成 AI と Web サイトで構成される．管理者はシステムの運用を行い，管理しているシステムで障害が発生した際に，原因調査や原因解決を行う人を表す．Kibana は原因調査や原因解決を行う際にログを検索するのに使用するオープンソースのプラットフォームを表す．生成 AI と Web サイトは，管理者が Kibana でログを検索した際に，ログの内容がわからない場合に使用されることを表す．(1) で，管理者がログを Kibana で検索する．障害に関連するログを Kibana で検索し，原因の調査を行う．(2) でログの内容を検索する．Kibana のログに書かれている内容が理解できなかった際，管理者はログの内容を検索する必要がある．そのため，ログの内容を理解できない場合，調べる作業が必要があり，障害解決までの時間を要する．

基礎実験

基礎実験は，2025 年 6 月 25 日，2025 年 6 月 26 日，2025 年 7 月 3 日に CDSL で学生であり STNS と踏み台サーバの管理者である佐藤健斗さん (以後佐藤さんとする) が STNS の障害の原因調査を行い，ログでの調査にどのくらい時間がかかったかを計測した．

図 3 は，2025 年 6 月 25 日の佐藤さんが調査にかかった時間の分類を表す．Y 軸は，原因調査にかかった時間を表し単位は秒をとる．凡例は，調査で行った内容を表す．凡

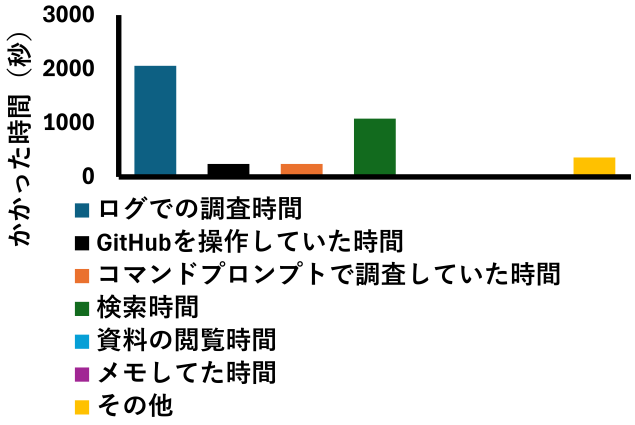

図 3　2025 年 6 月 25 日の佐藤さんが調査にかかった時間の分類

例は，ログでの調査時間，GitHub を操作していた時間，コマンドプロンプトで調査していた時間，検索時間，資料の閲覧時間，メモしてた時間，その他がある．2025 年 6 月 25 日の調査でかかった時間は，約 66 分である．ログでの調査に約 34 分，GitHub を操作していた時間が約 4 分，コマンドプロンプトで調査していた時間が約 4 分，検索時間が約 18 分，その他の時間が約 6 分である．

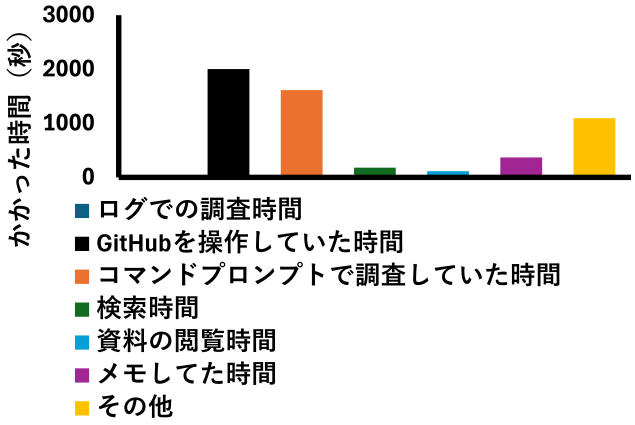

図 4　2025 年 6 月 26 日の佐藤さんが調査にかかった時間の分類

図 4 は，2025 年 6 月 26 日の佐藤さんが調査にかかった時間の分類を表している．Y 軸は，原因調査にかかった時間を表し単位は秒をとる．凡例は，調査で行った内容を表す．2025 年 6 月 26 日の調査でかかった時間は，約 89 分である．GitHub を操作していた時間が約 33 分，コマンドプロンプトで調査していた時間が約 26 分，検索時間が約 2 分，資料の閲覧時間が約 1 分，メモをしていた時間が約 6 分，その他の時間が約 18 分である．

図 5 は，2025 年 7 月 3 日の佐藤さんが調査にかかった時間の分類を表している．Y 軸は，原因調査にかかった時間を表し単位は秒をとる．凡例は，調査で行った内容を表す．2025 年 7 月 3 日の調査でかかった時間は，約 82 分である．ログでの調査に約 7 分，GitHub を操作していた時間が約 2 分，コマンドプロンプトで調査していた時間が約

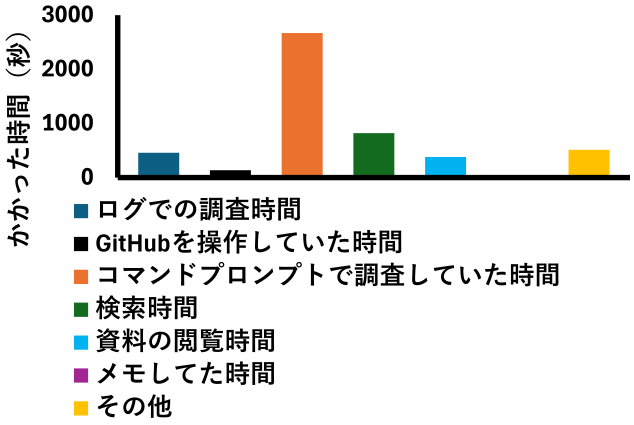

図 5　2025 年 7 月 3 日の佐藤さんが調査にかかった時間の分類

44 分，検索時間が約 13 分，資料の閲覧時間が約 6 分，その他の時間が約 8 分である．

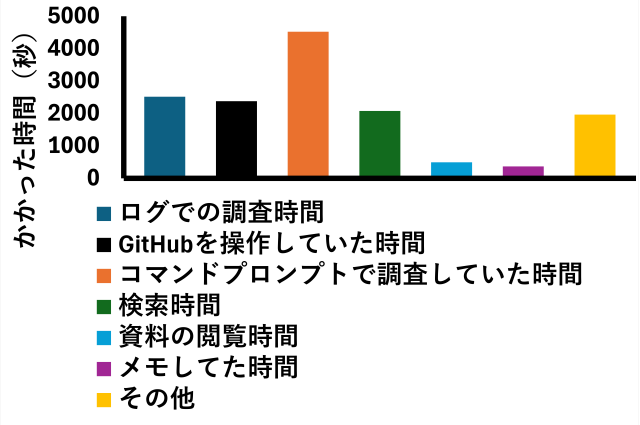

図 6　佐藤さんの調査にかかった全体の時間の分類

図 6 は，佐藤さんが調査にかかった全体の時間の分類を表しており，図 3，4，5 の結果を合計したものである．Y 軸は，原因調査にかかった時間を表し単位は秒をとる．凡例は，調査で行った内容を表す．全体の調査にかかった時間は，約 238 分である．ログでの調査に約 41 分，GitHub を操作していた時間が約 39 分，コマンドプロンプトで調査していた時間が約 75 分，検索時間が約 34 分，資料の閲覧時間が約 8 分，メモをしていた時間が約 6 分，その他の時間が約 32 分である．原因調査から原因解決で 1 番時間がかかっていたのはコマンドプロンプトで調査していた時間で約 75 分であり，STNS の調査全体の時間のうち約 31.6%である．主に原因解決の際にコマンドプロンプトを操作していた．原因解決のために，システムの pod や node の状態を繰り返し確認していたため時間がかかっていた．2 番目に時間がかかっていたのはログでの調査で約 41 分であり，STNS の調査の全体の時間のうち，約 17.2%である．ログでの調査は主に原因調査の際に行っていた．絞り込みをするためのクエリがわからなかったり，障害に関連するログがどこにあるかわからなかったりして，繰り返しログ

を探していたため時間がかかっていた．

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，課題に対して解決するための提案手法とユースケースや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の環境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．第 7 章では，本稿のまとめを述べる．

2. 関連研究

ログデータの複雑さと規模の大きさから，人間のオペレーターや管理者が問題診断と根本原因の分析を行うことは困難な場合がある．人間のオペレーターがログデータの注意を障害の根本原因に集中させ，問題診断のプロセスの複雑さを軽減するために，サービス指向システムにおける根本原因の分析とシステム診断の目的でストリーミングログデータの分析を支援し，選択されたデータサブセットを解釈するフレームワークを提案する論文がある [7]．フレームワークは SAT ソルバーアルゴリズムと連携し，システムアナリストが定義した目標モデルを用いてログから必要なイベントのみを抽出させる．目標モデルには，前提条件，発生，効果の述語が含まれる．SQL クエリや潜在的意味索引を使用して障害に関連するログエントリを特定する．これにより根本原因の分析の効率が向上し，人間のオペレーターの分析負担を軽減できる．この論文では，根本原因の分析を行う際の人間のオペレーターの負担軽減に焦点を当てているため，原因調査の時間短縮は行っていない．

情報技術産業の発展により，サーバー障害は重大な経済的な損失に繋がるため，根本原因の分析を自動で実施することが求められる．オペレーティングシステムログを活用し，サーバー障害の正確かつ効率的な根本原因の分析を行う自動化ソリューション ServerRCA を提案する論文がある [8]．ServerRCA では，構造化されていないオペレーティングシステムログをログテンプレートに変換する．次に，障害ログの階層構造を活用する階層マッチングアプローチにより障害イベントを正確に特定する．また，人間によるフィードバックメカニズムを組み込むことにより精度向上を図り，既存の障害イベントから障害伝播チェーンを構築する．この論文では，サーバー障害により利用できない際の根本原因の分析を特定することを目的としているが，本稿が目的とする根本原因の分析の時間短縮は行っていない．

クラスターシステムの障害診断においてシステムログは障害の原因を特定する際の重要な情報源である．しかし，ログの量が膨大で複雑にログイベントが絡み合っているため，障害の原因特定に役立つログイベントは一部に過ぎない．また，手動で特定の障害の原因を特定するのは困難な場合がある．この論文では，syslog から障害の根本原因となるイベントの順序を再構築し，イベント間の相関関係を導き出す手法 FDiag を提案する論文がある [9]．FDiag は，ログエントリを構造化されたメッセージテンプレートに変換し，統計的な相関分析を用いて障害との因果関係を特定する．この論文では，ログを構造化メッセージテンプレートに変換し，統計的な相関分析を用いて障害との因果関係を特定するが，本稿ではコンテナログを時間帯とコンテナ名別に集計し，基準のログ件数の中央値と最新の時間帯のログ件数を比較することで障害の原因となるコンテナ名を特定する．

3. 提案

提案手法では，現在時刻の 3 時間前から現在時刻までの Elasticsearch に保存されているログからタイムスタンプとコンテナ名を取得する．タイムスタンプをもとにコンテナ名別に 15 分単位でログ件数を集計する．その後，15 分ごとのログ件数から中央値 (以後，基準のログ件数とする) を求める．基準のログ件数と最新の時間帯のログ件数を比較することによる障害が起きたコンテナ名の特定を目的とする．ログの時間帯による件数の傾向を反映させるため，ログの集計期間を 3 時間とした．集計期間が短い場合には，一時的にログ件数が増えた場合に影響を受けやすい．集計期間が長い場合には，ログの集計に時間がかかる．GitHub の可用性レポートの集計結果[*1]によると約 70%の障害が 200 分以内に解決している．そのため，3 時間 (180 分) は障害が仮に起きてログ件数に影響を受けにくい期間である．基準のログ件数の集計から中央値を求め，最新の時間帯のログと比較する．基準のログ件数が最新の時間帯のログ件数より大きければ異常無しとする．最新の時間帯のログ件数が基準のログ件数より大きければ異常有りとする．中央値を求めて最新のログ件数と比較することで，障害がどのコンテナで発生したか特定し，原因調査の時間を短縮する．

提案方式

提案方式は，提案ソフトウェアを管理者が実行した場合，ライブラリと設定の読み込みとログを検索する期間を決定する．次に，Elasticsearch からログを取得する．取得したログからタイムスタンプとコンテナ名を取り出す．その後，タイムスタンプとコンテナ名ごとに基準のログ件数を集計する．集計した基準のログ件数を最新のログを比較して障害に関連するログを出力する．比較をして基準のログ件数より最新のログの値が大きければ異常有りと判断し，関連するコンテナ名を出力する．

[*1] https://github.com/tomoyk/github-availability-report-analysis

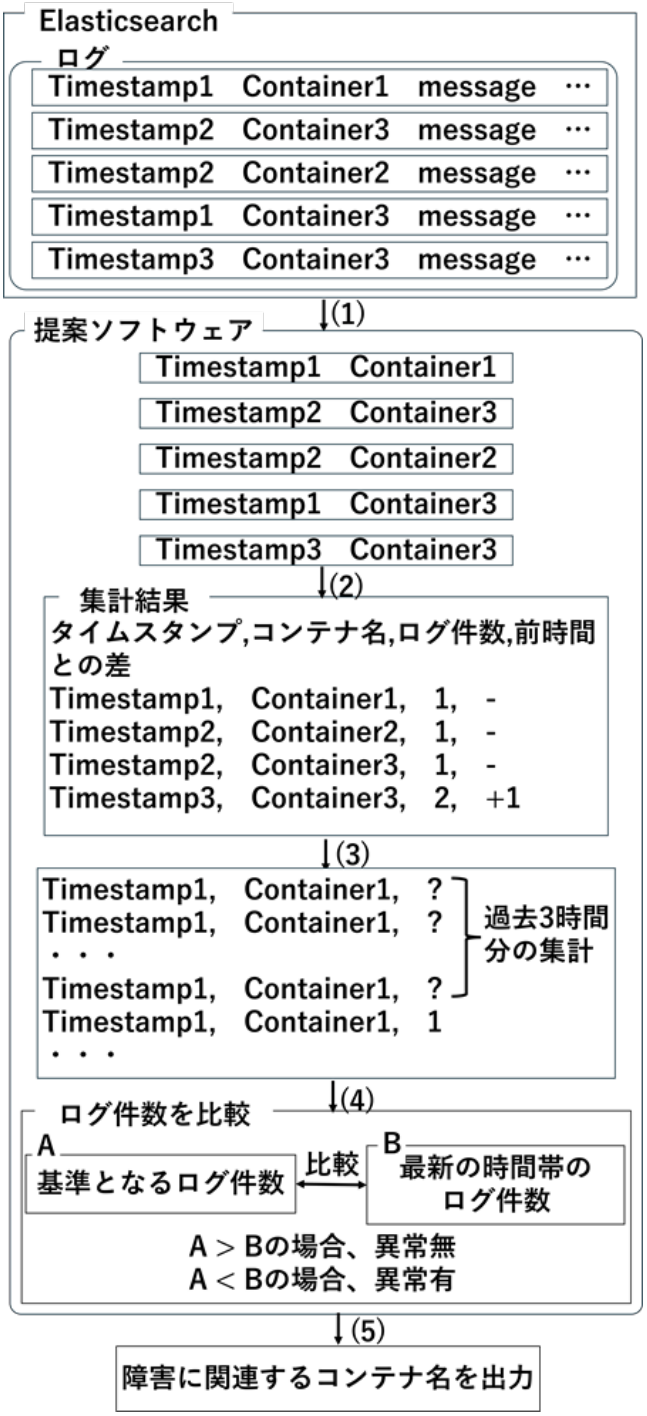

図 7 時間帯ごとのコンテナ名のログ件数を出力するまでの過程

図 7 は，時間帯ごとのコンテナ名のログ件数を出力するまでの過程を表している．図 7 は，Elasticsearch，提案ソフトウェアで構成されている．Elasticsearch はログサーバでログが保存されている．各ログにはタイムスタンプやコンテナ名，メッセージが含まれている．図 7 の 1 件目のログのタイムスタンプは Timestamp1 であり，コンテナ名は Container1 であり，メッセージは message である．

ログ 1 は，Elasticsearch に保存されているログの例である．ログ 1 の「Jun 17, 2025 @ 17:25:07.475」はタイムスタンプであり，図 7 の Elasticsearch のログにある Timestamp1 の列に該当する．ログ 1 の「builder」はコンテナ名であり，図 7 の Elasticsearch のログにある Containe1 の列に該当する．ログ 1 の「error: could not lock config file /root/.gitconfig: File exists」はメッセージであり，図 7 の Elasticsearch のログにある message に該当する．

ログ 1 Elasticsearch に保存されているログの例

```
1 Jun 17, 2025 @ 17:25:07.475
  builder     error: could not lock config file
  /root/.gitconfig: File exists
```

提案ソフトウェアは Elasticsearch のログからタイムスタンプごとのコンテナ名を取得し，基準のログ件数と最新のログを比較して障害に関連するログを出力する．(1)Elasticsearch から取得するログを提案ソフトウェアに入力する．(2) 提案ソフトウェアに入力されたログからタイムスタンプとコンテナ名を取得する．(3) 時間帯別にコンテナ名の件数を集計し，集計の結果からコンテナ名ごとに基準のログ件数を求める．(4) 基準のログ件数 (A) と最新の時間帯のログの件数 (B) を比較する．基準のログ件数 (A) が最新の時間帯のログの件数 (B) より大きい場合には，異常無しと判断する．基準のログ件数 (A) が最新の時間帯のログの件数 (B) より小さい場合には，異常有りと判断する．(5) ログ件数を比較の結果から，障害に関連するコンテナ名を決定し，コンテナ名を出力する．

ユースケース・シナリオ

本稿では，Pod のコンテナで障害が発生した場合を想定する．

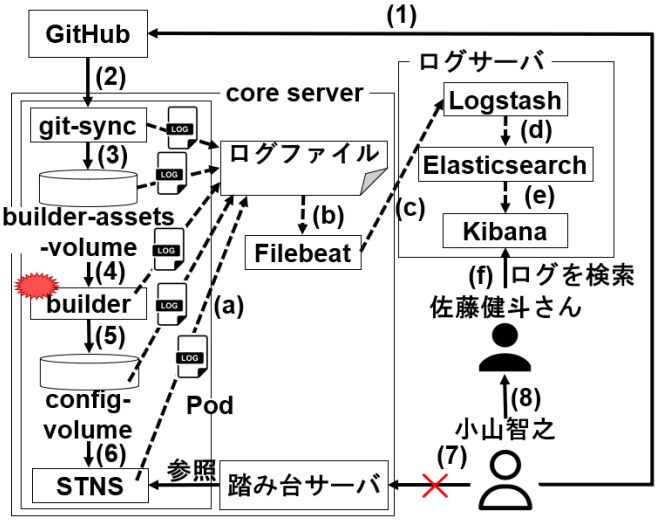

図 8 公開鍵を登録したが ssh 接続できない障害について調査するまでの過程

図 8 は，公開鍵を登録したが ssh 接続できず障害が発生し調査するまでの過程を表している．小山智之 (以後

小山とする) は，CDSL の学生で公開鍵を GitHub に登録し，踏み台サーバに ssh 接続しようとしたができなかった人を表す．佐藤さんは，CDSL の学生で STNS サーバと踏み台サーバの管理を担当している．GitHub は，公開鍵を登録するためのウェブサービスを表す．core server は，Pod やログファイルが動作するサーバを表す．Pod は，コンテナやストレージが動作するための実行環境を表す．git-sync は，core server の Pod 内にあるコンテナを表す．builder-assets-volume は，core server の Pod 内にあるストレージを表す．builder は，core server の Pod 内にあるコンテナを表す．config-volume は，core server の Pod 内にあるストレージを表す．STNS は，core server の Pod 内にあるコンテナを表す．踏み台サーバは，STNS サーバにログインするための中継サーバを表す．ログファイルは，core server の Pod 内にあるコンテナやストレージのログをファイル形式で記録していることを表す．Filebeat は，core server のログファイルからログを収集するソフトウェアを表す．ログサーバは，ソフトウェアが動作するサーバを表す．Logstash は，Filebeat に収集されたログを構造化するソフトウェアを表す．Elasticsearch は，Logstash で構造化されたログを保存するソフトウェアを表す．Kibana は，Elasticsearch に保存されたログを可視化させるオープンソースのプラットフォームを表す．公開鍵を登録したが ssh 接続できない障害について調査するまでの過程を以下に記述する．

(1) 小山が，GitHub に公開鍵を登録する．
(2) GitHub に登録された公開鍵が core server にある git-sync にダウンロードする．
(3) git-sync にダウンロードされた公開鍵は builder-assets-volume にダウンロードする．
(4) builder-assets-volume にダウンロードされた公開鍵が builder にダウンロードする．
(5) builder にダウンロードされた公開鍵が config-volume にダウンロードする．
(6) config-volume にダウンロードされた公開鍵が STNS にダウンロードされ，管理する．
(7) 小山が踏み台サーバに ssh 接続しようとしたができなかった．
(8) STNS サーバと踏み台サーバの管理者である佐藤さんに小山が問い合わせる．

管理者が障害の原因調査を行うまでの過程を以下に記述する．

(a) Pod 内のコンテナとストレージのログがログファイルに記録する．
(b) ログファイルに記録されたログが Filebeat で収集する．
(c) Filebeat に収集したログを Logstash に送信し，構造化する．
(d) Logstash で構造化されたログを Elasticsearch に送信し，保存する．
(e) Elasticsearch に保存されたログを Kibana で可視化する．
(f) 佐藤さんが Kibana でログを検索し，踏み台サーバに ssh 接続できなかった原因を調査する．

小山が公開鍵を GitHub に登録する．GitHub に登録された公開鍵は core server の Pod 内にあるコンテナ（git-sync, builder, STNS），ストレージ（builder-assets-volume, config-volume）に図 8 の矢印の順にダウンロードされる．GitHub に登録された最終的に STNS にダウンロードされ，管理される．小山が踏み台サーバにアクセスしたが，アクセスできなかったため，STNS と踏み台サーバの管理者である佐藤さんに問い合わせをした．コンテナとストレージのログはファイル形式で core server のログファイルで記録される．ログファイルに記録されたログが Filebeat に収集される．Filebeat に収集されたログは，ログサーバの Logstash に送信され，構造化する．Logstash で構造化されたログは Elasticsearch に送信され，保存される．Elasticsearch に保存されたログを Kibana で可視化する．佐藤さんは Kibana でログを検索し，小山が踏み台サーバにアクセスできなかった原因を調査する．

ログ 2 連続して出ていたエラーログ

```
1 fatal: detected dubious ownership in
  repository at '/src/
  a97e5a809c465d2b2fad6a4ae33638d9259c73b0'
2 To add an exception for this directory, call:
3 git config --global --add safe.directory /src/
  a97e5a809c465d2b2fad6a4ae33638d9259c73b0
4 error: could not lock config file /root/.
  gitconfig: File exists
```

ログ 2 は佐藤さんが Kibana でログを検索，調査した際に連続して出ていたエラーログである．ログ 2 の 1 行目は，Git がリポジトリが現在のユーザーが所有しているものではないと判断し，セキュリティ上の理由で操作を拒否していることを表す．ログ 2 の 2 行目は，リポジトリの使用を許可したい場合は，次のコマンドを実行して信頼できるディレクトリとして登録する必要があることを表す．ログ 2 の 3 行目は，ログ 2 の 2 行目の説明にあった次のコマンドを表す．ログ 2 の 4 行目は，Git の設定ファイルを変更しようとしたがロックファイルが既に存在しているため書き込みに失敗したことを表す．

コード 1 は，CDSL で管理している GitHub にある build.sh の一部である．ログ 2 の 2 行目と 3 行目の内容から，コード 1 を変更する必要があることがわかる．障害の

コード 1 build.sh の中身の一部

```
1 git config --global --add safe.
  directory /src/$(ls ../ | head -1)
```

対応をした佐藤さんは，build.sh をコード 2 に変更した．コード 2 は，ログ 2 の内容から変更後の build.sh の中身の一部である．

コード 2 変更後の build.sh の中身の一部

```
1 git config --global --add safe.directory /src/
  a97e5a809c465d2b2fad6a4ae33638d9259c73b0
```

4. 実装

ソフトウェアの作成には，開発言語である Python 3.12.3 を使用した．開発ソフトウェアのモジュールと処理の過程について説明する．

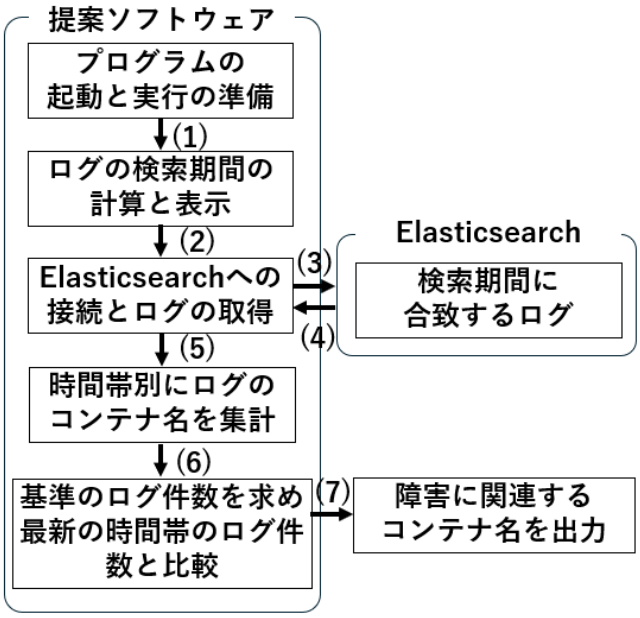

図 9 ソフトウェアの構成図

図 9 は，ソフトウェアの構成図を表している．図 9 は，Elasticsearch，提案ソフトウェアで構成されている．Elasticsearch は，ログが保存されているソフトウェアを表す．提案ソフトウェアはログからタイムスタンプとコンテナ名を取得して時間帯ごとに集計し，中央値と最新の時間帯を比較して障害に関連するコンテナ名を出力する．(1) コマンドを実行するとプログラムの起動と実行の準備が行われる．(2) ログの検索期間を計算し，ターミナルに表示される．(3) 提案ソフトウェアが Elasticsearch への接続を行い，ログを取得する．(4)Elasticsearch から検索期間に合致するログを提案ソフトウェアに入力する．(5) 入力されたログを時間帯とコンテナ名別に集計を行う．(6) 集計の結果から中央値を求め最新の時間帯のログ件数と比較する．(7) 比較した結果から障害に関連するコンテナ名を出力する．

プログラムの起動と実行の準備

提案ソフトウェアのプログラムの起動と実行の準備を import 文を使用して行う．Elasticsearch に保存されたログを日本時間で集計する．ログを集計するために必要なモジュールは，datetime，timedelta，timezone，collections，os，elasticsearch，scan，pytz である．datetime は，日付と時刻を扱うためのモジュールである．timedelta は時間と時間の差を計算するためのモジュールである．timezone は，Python の標準モジュールで，UTC や決まった時差を扱うためのモジュールである．collectons は，カウントやグループ化を行うためのモジュールである．os は，コンピュータのオペレーティングシステムと連携するためのモジュールである．elasticsearch は Elasticsearch に接続し，データの検索を行うためのモジュールである．scan は，Elasticsearch のログをメモリを圧迫せずに，抜けや重複なしで全て取得するためのモジュールである．pytz は，タイムゾーンを正確に扱うためのモジュールであり．日本時間を正確に処理するために使用する．これらのモジュールにより，ログの取得，時刻の変換，集計を実行する．

ログの検索期間の計算と表示

ログの検索期間の計算とターミナルに検索期間の表示を行う．関数を使って，Elasticsearch からログを取得する期間を，現在時刻を基準にして，日本時間で計算する．取得した現在時刻の 3 時間前を検索期間の開始とし，現在時刻を終了として，検索期間を決定する．決定された検索期間は print 文でターミナルに表示する．

Elasticsearch への接続とログの取得

Elasticsearch への接続とログの取得を行う．決定した検索期間を引数として，Elasticsearch が利用している世界標準時刻に変換する．次に変換された時間を使って開始時刻から終了時刻までのログを Elasticsearch にリクエストする．Elasticsearch がリクエストを受け取ると，該当するログを返信する．プログラムでログを 1 件ずつ受け取り，ログからタイムスタンプとコンテナ名を抜き出す．最後に取得したログの件数を print 文でターミナルに表示する．

時間帯別にログのコンテナ名を集計

時間帯別にログのコンテナ名を集計する．ログのタイムスタンプを日本時間に変換し，15 単位で集計する．例として，14 時 34 分のログは 14 時 30 分として扱われ，14 時 46 分のログは 14 時 45 分として扱われます．集計したタイムスタンプとコンテナ名を組み合わせたものをキーとして，

同じキーを持つログの件数を数える．

基準のログ件数を求め最新の時間帯のログ件数と比較

基準のログ件数を求め最新の時間帯のログ件数と比較する．基準のログ件数の集計の結果から，基準のログ件数を求め最新の時間帯のログ件数と比較する．最新の時間帯のログ件数が基準のログ件数より大きい場合は，異常有りと判断する．基準のログ件数が最新の時間帯のログ件数より大きい場合，異常無しと判断する．

障害に関連するコンテナ名を出力

障害に関連するコンテナ名を出力する．基準のログ件数を求め最新の時間帯と比較の結果，異常有りの場合は障害に関連するコンテナ名を出力する．例として Container1 がコンテナ名のコンテナで基準のログ件数が 100 件．最新の時間帯のログ件数が 50 件だった場合，基準のログ件数の方が最新の時間帯のログ件数より多いため，Container1 を障害に関連するコンテナ名として出力する．

5. 評価実験

評価実験では，STNS に関する障害と doktor 環境で発生した Rook-Ceph の障害の 2 種類のシナリオを対象に行う．各シナリオにおいて，提案ソフトウェアでログの件数を出力し，障害の原因となるコンテナ名を特定できるかを評価の基準とし，検出率で評価する．

実験環境

実験環境として，以下の VM を 1 台用意した．

- vCPU：2Core
- メモリ：2GB
- ストレージ：30GB
- OS：Ubuntu24.04

STNS で起きた障害と Rook-Ceph の障害を対象に，管理者が特定した原因と提案ソフトウェアで特定した原因を比較する．管理者が特定した原因を正解として，提案ソフトウェアで正解となる原因を特定できたかを評価する．

図 10 は，2 種類の障害を想定した提案ソフトウェア使用時の検出率の評価方法を表している．図 10 は，管理者，提案ソフトウェア，障害 a，障害 b で構成されている．管理者は提案ソフトウェアを使用して，障害の原因の特定を行う人を表す．提案ソフトウェアは，管理者によってプログラムが実行されるとログの件数を求めて障害が起きているコンテナを見つけることを表す．障害 a と障害 b は想定した 2 種類の障害を表す．全体の障害のうち提案手法が検出した障害の件数を n とする．図 10 では，障害 a で起きた障害に対して，障害の原因を特定できたが，障害 b に対して障害の原因を特定できなかったことを表す．この場合, 2 種類の障害のうち，n 種類を障害として特定できているため検出率は n/2 となる．

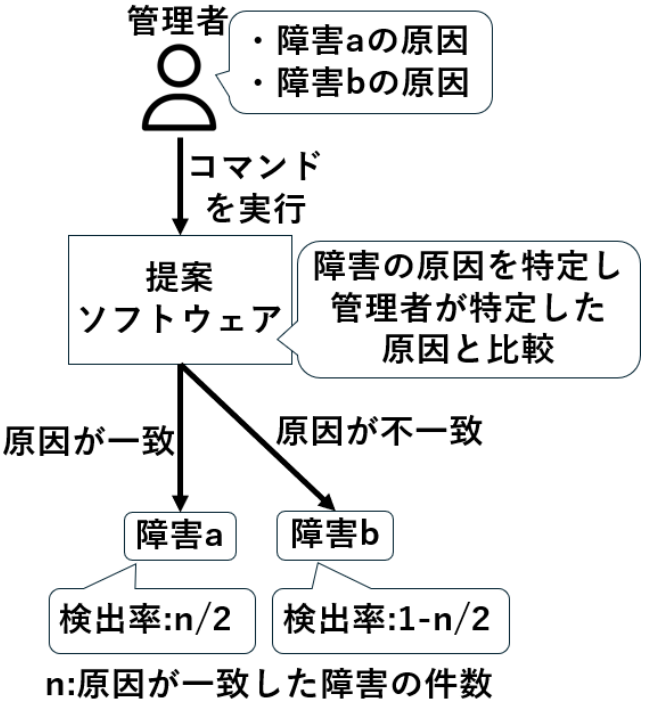

図 10 2 種類の障害を想定した提案ソフトウェア使用時の検出率の評価方法

実験のデータセット

CDSL で運用している Elasticsearch から 30 日分のログを取得する．Elasticsearch には Kubernetes クラスタのコンテナのログが保存されている．例えば，Rook-Ceph や PowerDNS，STNS，Kea DHCP，OpenSSH のログが保存されており，これらは CDSL で共通で使用している core サーバで動作しているコンテナである．

6. 議論

本稿の提案では，Elasticsearch のログからタイムスタンプとコンテナ名を取得し，基準のログ件数を最新のログ件数と比較することで原因を特定する．しかし，コンテナ内で動いているアプリケーションの実装によりログ件数が変わり，エラーが起きてもログ件数が増加しないアプリケーションがある．その際，本稿の提案では原因のログを見つけられない．これは，別のメトリクスと組み合わせることで原因のログを見つける．提案ソフトウェアの入力にメトリクスを追加することでコンテナの中で動いているアプリケーションでエラーが起きても障害の原因となるコンテナ名を検出できる．

本稿の提案では，ログの件数の増加を兆候として捉えている．しかし，システムによってはログの件数の減少も兆候として捉えられる．例えば，約 1000 件のログが出力されているシステムで異常が発生してダウンした場合，ログは

出力されなくなり，件数は 0 件になる．また，アプリケーションによってはログにエラーの原因が出てこない場合がある．これは，提案ソフトウェアでログの件数だけでなくメッセージの内容を分析することで，障害の原因となるコンテナ名を検出できる．

7. おわりに

課題は，障害が発生してログを調査する際にログの検索クエリの作成にスキルやログのメッセージを理解するための専門知識が必要である．そのため，スキルや経験が不足していると調査による原因の特定や解決に時間がかかることである．基礎実験で，STNS の障害を対象に原因調査の際にログの調査にかかるかを計測した．ログでの調査にかかった時間は約 41 分であり，調査全体で 2 番目に時間がかかっていた．提案手法は，Elasticsearch のログのタイムスタンプをもとに基準のログ件数を時間帯別にコンテナ名で集計する．集計した結果から基準のログ件数を求め最新の時間帯のログ件数と比較を行い，基準のログ件数より最新の時間帯のログ件数が多い場合，異常有りと判断することで障害に関連するコンテナ名を特定する．評価実験では，STNS の障害と Rook-Ceph で起きた障害の 2 種類のシナリオを対象に，管理者が特定した原因と提案ソフトウェアで特定した原因を比較し，各シナリオにおいて障害に関連するコンテナ名を検出できたかの検出率で評価する．

参考文献

[1] Zhu, J., He, S., He, P., Liu, J. and Lyu, M. R.: Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics, *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 355–366 (online), DOI: 10.1109/ISSRE59848.2023.00071 (2023).

[2] Sankar, P., George, D. E. and S, A. S. N.: Social media monitoring using ELK Stack, *2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, Vol. 1, pp. 231–235 (online), DOI: 10.1109/SPICES52834.2022.9774273 (2022).

[3] 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 Science (ICECOS)*, pp. 338–342 (online), DOI: 10.1109/ICECOS47637.2019.8984494 (2019).

[4] 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).

[5] 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).

[6] 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).

[7] Zawawy, H., Kontogiannis, K. and Mylopoulos, J.: Log filtering and interpretation for root cause analysis, *2010 IEEE International Conference on Software Maintenance*, pp. 1–5 (online), DOI: 10.1109/ICSM.2010.5609556 (2010).

[8] Shi, J., Jiang, S., Xu, B. and Xiao, Y.: ServerRCA: Root Cause Analysis for Server Failure using Operating System Logs, *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 486–496 (online), DOI: 10.1109/ISSRE59848.2023.00083 (2023).

[9] Chuah, E., Kuo, S.-h., Hiew, P., Tjhi, W.-C., Lee, G., Hammond, J., Michalewicz, M. T., Hung, T. and Browne, J. C.: Diagnosing the root-causes of failures from cluster log files, *2010 International Conference on High Performance Computing*, pp. 1–10 (online), DOI: 10.1109/HIPC.2010.5713159 (2010).