ログのタイムスタンプとコンテナ名での99パーセンタイルの閾値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：システムで障害が発生した際，管理者はログを確認するために Kibana にアクセスし，障害の原因調査を行う．課題は，障害の原因調査でログを確認する際，専門知識やスキルが不足していると原因の特定や復旧に時間を要することである．提案手法では，タイムスタンプとコンテナ名をもとにログ件数をある時間間隔ごとにコンテナ名別に集計を行う．集計結果をコンテナ名ごとに昇順に並び替え，99 パーセンタイルの位置にあるログ件数を閾値とする．閾値を超える値を異常値として検出し，障害が発生しているコンテナ名を特定する．評価実験では，ユーザ ID を管理するサーバである STNS の builder コンテナで起きた障害をシナリオとして，検出の可否と提案ソフトウェアありの調査に要する時間の測定の 2 つを評価指標とする．検出の可否の測定は，1 週間分の検索期間と集計間隔を変更し，提案ソフトウェアで障害が発生したコンテナを特定できたか測定する．提案ソフトウェアありの調査に要する時間の測定は，被験者が提案ソフトウェアを使用して障害の発生箇所を特定するまでの時間を計測し，人が行う必要がある作業と自動化が可能な作業をそれぞれ測定した．検出の可否の測定は，2025 年 11 月 3 日の 14 時から 2025 年 11 月 3 日の 17 時 33 分に行った．検出の可否の測定では，集計間隔が 1 分と 5 分と 15 分の場合，各検索期間で検出でき，10 分と 30 分の場合，各検索期間で検出できなかった．提案ソフトウェアありの調査に要する時間の測定は，2025 年 11 月 5 日の 16 時 25 分から 2025 年 11 月 5 日の 16 時 59 分に行い，障害原因の全体の調査時間は 1892 秒であった．人が行う必要がある作業に要した時間は 774 秒であり，全体の作業時間の約 41%を占めている．一方，自動化が可能な作業に要した時間は 1118 秒であり，全体の作業時間の約 59%を占めている．この結果から，全体の作業時間の半分以上を自動化が可能な作業で占めていることがわかり，管理者が行う作業の一部を削減したことを確認できた．また，障害の原因調査における負担の軽減と調査全体の時間短縮により障害の原因調査の効率化に寄与している．

1. はじめに

背景

ログとは，システムに関する活動を時系列で記録しており，ソフトウェアシステムの開発と保守に用いられている．開発者や運用エンジニアはログを確認して，システムの挙動を追跡，分析する [1,2].

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 クラウド・分散システム研究室 (以後 CDSL と呼ぶ) では，サーバにアクセスする際，踏み台サーバにログインする必要がある．踏み台サーバにログインするには，GitHub に公開鍵を追加する必要があり，CDSL の学生である小山智之 (以後小山さんとする) が，公開鍵を登録したが，踏み台サーバにログインできない障害が 2025 年 5 月 16 日に発生した．小山さんは踏み台サーバの管理者である佐藤健斗 (以後佐藤さんとする) に障害の原因調査と原因対応を依頼した．

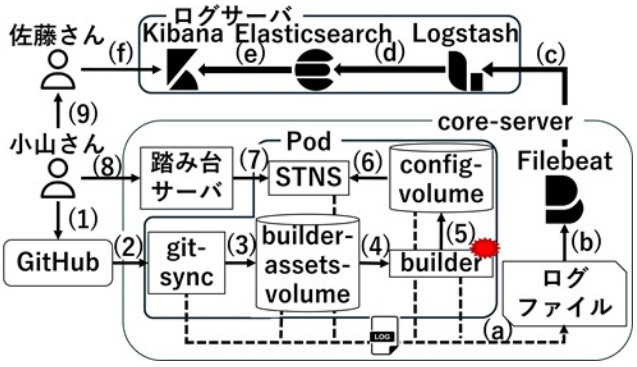


図 1　STNS の builder コンテナで起きた障害の概要

図 1 は，STNS の builder コンテナで起きた障害の概要を表している．図 1 は, GitHub, 小山さん, 佐藤さん, ログサー

[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

バ，Kibana，Elasticsearch，Logstash，core-server，Pod，git-sync，builder-assets-volume，builder，config-volume，STNS，踏み台サーバ，ログファイル，Filebeat で構成されている．GitHub は，公開鍵を登録するために使用しているウェブサービスである．小山さんは，CDSL の学生であり，障害が発生したことを管理者に報告した人物である．佐藤さんは，CDSL の学生であり，STNS サーバの管理者である．Kibana は，Elasticsearch に保存されたログを可視化する．Elasticsearch は，Logstash で構造化されたログを保存するオープンプラットフォームである．Logstash は，Filebeat で収集したログを構造化する．core-server は，Pod やログファイルが動作するサーバである．Pod は，コンテナやストレージを動作させるための実行環境である．git-sync は，core-server の Pod 内にあるコンテナの 1 つである．builder は，core-server の Pod 内にあるコンテナの 1 つであり，ユースケースで障害が発生した箇所である．config-volume は，core-server の Pod 内にあるストレージの 1 つである．STNS は，core-server の Pod 内にあるコンテナの 1 つである．踏み台サーバは，CDSL のサーバにアクセスする際の中継役のサーバである．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 [3]．図 1 では，core-server 内のログをファイル形式で記録している．システムの情報が記録され，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 [4]．Filebeat は，ログファイル内のログを収集する．

図 1 の流れについて説明する．(1) で小山さんが GitHub に公開鍵の登録を行う．(2) から (6) の順番で，GitHub に登録した公開鍵がダウンロードされ，STNS で管理される．(7) で小山さんが踏み台サーバに SSH 接続を行い，(8) で踏み台サーバが STNS に保存されている公開鍵を参照する．(9) で小山さんが踏み台サーバに SSH 接続できなかったことを佐藤さんに報告する．(a) で core-server の Pod 内にあるコンテナとストレージのログが，ログファイルに記録される．(b) でログファイルに記録されたログは Filebeat で収集される．(c) で Filebeat で収集されたログは Logstash に転送され構造化される．(d) で Logstash で構造化されたログが Elasticsearch に転送され，保存される．(e) で Elasticsearch に保存されたログが Kibana で可視化される．(f) で佐藤さんが Kibana でログを確認し，障害が起きた原因を調査する．

佐藤さんが Kibana でログを検索し，原因調査を行った結果，core-server の Pod 内にある builder コンテナで障害が発生していることがわかった．障害が発生した時に，3 つのログが頻繁に出力されており，以下に出力されていたログを記述する．

ログ 1 は，STNS の builder コンテナで障害が発生していた際に，連続で出力されていたエラーログの 1 つ目である．ログの内容は，GitHub のリポジトリのディレクトリの所有者と，現在 GitHub を操作しているユーザが異なっていることを表している．GitHub が，このディレクトリの所有権が疑わしいと判断し，安全のために操作を停止していることを意味する．

ログ 1 連続で出力されていたログの 1 つ目

```
fatal: detected dubious ownership in repository at
'/src/a97e5a809c465d2b2fad6a4ae33638d9259c73b0'
```

ログ 2 連続で出力されていたログの 2 つ目

```
To add an exception for this directory, call:
git config --global --add safe.directory /src/
a97e5a809c465d2b2fad6a4ae33638d9259c73b0
```

ログ 2 は，STNS の builder コンテナで障害が発生していた際に，連続で出力されていたログの 2 つ目である．ログの内容は，ログ 1 を解決するために GitHub が推奨しているコマンドを表す．「git config –global –add safe.directory /src/a97e5a809c465d2b2fad6a4ae33638d9259c73b0」を安全なディレクトリとして GitHub のグローバル設定ファイルに追加するように提示しており，追加することでユーザがディレクトリの所有権を気にせずに Git 操作を続行できることを表している．

ログ 3 連続で出力されていたログの 3 つ目

```
error: could not lock config file /root/.gitconfig:
 File exists
```

ログ 3 は，STNS の builder コンテナで障害が発生していた際に，連続で出力されていたログの 3 つ目である．ログの内容は，ログ 2 を実行しようとした際に発生したエラーログである．GitHub の設定ファイルを変更する際に，他のプロセスによる同時書き込みを防ぐために，一時的にロックファイルを作成することが書かれているが，ロックファイルが既に存在していることを示している．設定変更の処理が異常に終了し，ロックファイルが削除されずに残ってしまった可能性があり，残留したロックファイルが原因で新たな設定変更を実行できなかったことが示されている．

佐藤さんは，STNS の builder コンテナで障害が発生した際の原因調査と対応に約 238 分要していた．原因の調査で一番時間を要した項目がログでの調査で約 41 分であり，全体の作業時間の約 17.2%である．以下に佐藤さんが原因調査に時間を要していた原因を記述する．

- ログの内容がわからない

- どのログを見るか迷った
- 検索クエリに入力すべきことがわからない

佐藤さんが見ているログは正しかったが，これらにより原因調査に時間を要していた．

CDSL では，ログサーバを構築し，システムのログを管理している．図 2 は，CDSL のログサーバの構成を表す．図 2

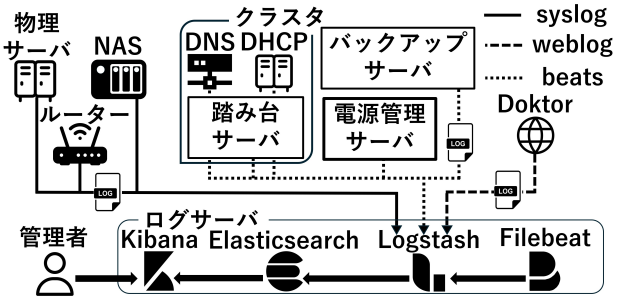


図 2　CDSL のログサーバの構成

は，管理者，ログサーバ，Elasticsearch，Logstash，Kibana，Filebeat，PV，NAS，物理サーバ，ルータ，Doktor，DNS，DHCP，踏み台サーバ，バックアップサーバ，電源管理サーバ，syslog，weblog，beats で構成される．管理者は，障害が発生した際に Kibana でログを確認する CDSL の学生を表す．ログサーバは，CDSL のシステムからログを管理するサーバを表す．Elasticsearch は，Logstash から送信された構造化されたログを保存するソフトウェアを表す．Logstash は，Filebeat により収集されたログを構造化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し，管理者がログを見られるようにするオープンソースのプラットフォームを表す．Filebeat は，CDSL のシステムからログを収集し，Logstash に送信するソフトウェアを表す．NAS は Network Attached Storage の略であり，ネットワーク経由で利用できるストレージである．物理サーバは，CDSL で使用している VMware ESXi をインストールしたサーバである．また，サーバ仮想化を実現するためのハイパーバイザを表す．ルータは，CDSL のネットワーク間のデータの通信を制御する機器である．Doktor は，CDSL の技術論文を検索およびダウンロードできるマイクロサービスで実装している Web サービスである [5]．DNS は Domain Name System の略であり，ドメイン名に対して IP アドレスをはじめとする情報を紐づける技術である．DHCP はネットワーク接続の際に必要な IP アドレスを自動で割り当てる．踏み台サーバは，CDSL のサーバにアクセスする際の中継役のサーバであり，Filebeat は踏み台サーバのログを収集し，Logstash に転送する．バックアップサーバは，データを複製して保管しておき，サーバが故障した際にデータを復旧するためのコンピュータである．電源管理サーバは，消費電力の削減を目的として，CDSL の学生が VMware ESXi を使用しない時間帯に電源を落とすサーバである．syslog は，ログを収集する際のデータ分類の単位の 1 つであり，CDSL では Doktor を対象としている．weblog は，ログを収集する際のデータ分類の単位の 1 つであり，CDSL では DNS，DHCP，踏み台サーバ，バックアップサーバ，電源管理サーバを対象としている．beats は，ログを収集する際のデータ分類の単位の 1 つであり，CDSL では NAS，VMware ESXi，ルータを対象としている．ログは，ELK Stack と Filebeat で管理する．ELK Stack とは，Elasticsearch，Logstash，Kibana を組み合わせたオープンソースアプリケーションである [6]．Elasticsearch とは，主にデータを整理し，容易にアクセスできるように設計されたリアルタイム分散型ソーシャル分析エンジンである [7]．Logstash とは，構造化データと非構造化データの両方を一元管理および解析するためのオープンソースフレームワークである [8]．Kibana とは，グラフや表の形式でログを視覚化するためのコンポーネントである [9]．Filebeat とは，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 [10]．

課題

運用の経験の浅いサーバ管理者は，次の要因でログを使った障害の原因調査に時間がかかっている．

- 検索クエリに何を入力すればいいかわからない
- ログのメッセージに書かれている内容がわからない
- 見ているログが正しいかわからない

このようなわからないことがあった際，生成 AI に質問したり，Web で検索したりする必要があり，調査に時間を要する．

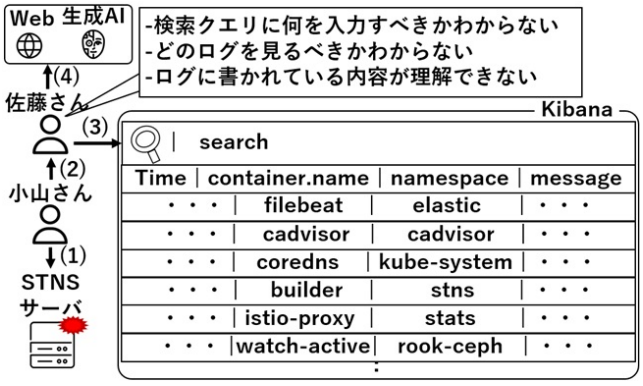


図 3　経験が少ない管理者がログで原因調査をする際のシナリオ

図 3 は，経験が少ない管理者がログで原因調査をする際のシナリオを表す．図 3 は，STNS サーバ，小山さん，佐藤さん，Kibana，Web，生成 AI で構成される．STNS サーバは，ユーザである小山さんが利用している時に障害

が発生したサーバを表す．小山さんは，利用していたシステムで障害が発生したことを認知し，佐藤さんに報告，調査と解決の依頼をする人を表す．佐藤さんは，障害が起きた際に Kibana で原因を調査する人を表す．Kibana は，佐藤さんが障害の原因調査をする際にログを検索するために使用するオープンソースのプラットフォームである．Web は，佐藤さんが Kibana でログを検索する際に，検索クエリに何を入力すべきかわからない時に調べるために使用するツールを表す．生成 AI は，佐藤さんが Kibana でどのコンテナログを見るべきかわからない時とログに何が書かれているのか，内容が理解できない時に調べるために使用するツールである．(1) で，システムで障害が発生し，利用していた小山さんが障害を認知し，(2) で佐藤さんに障害を報告し，原因調査と対応を依頼する．(3) で，障害が発生した際に佐藤さんが Kibana を使用して，障害が発生した原因を調査する．(4) で，佐藤さんがログを調査した際にわからないことがあれば，Web や生成 AI を使用してわからないことを検索する．(4) のように，わからないことがあると Web や生成 AI で検索する時間が生じてしまい，障害の原因調査や原因解決までに時間を要する要因となる．

22 種類の障害を対象に初期理解，環境設定，障害再現，障害特定，障害スコープ設定，障害箇所の特定，障害修正の 7 項目に対しログでの調査にどれくらい時間がかかるかを測定した論文がある [11]．マイクロサービスシステムに対して障害分析を行っており，分析の結果，初期理解，障害範囲の特定，障害箇所の特定の 3 項目でのログ分析に時間を要していた．ログ分析に時間を要する理由は 2 つある．1 つ目は開発者がログから断片的な情報をもとに，分散したコンテキストを再構築する点である．2 つ目は既存のデバッグ手法の限界である．複数のサービスにまたがる分散トレーシングのコンテキストが欠如している場合，非効率となり時間を要する．

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議論する．第 3 章では，課題に対する提案手法，ユースケースとシナリオについて説明する．第 4 章では，提案手法をもとに実装したソフトウェアについて説明する．第 5 章では，評価実験について説明する．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

マイクロサービスシステムでは複雑性と大規模性により，障害を避けることができず，膨大な数のメトリクスが存在するため根本原因の特定が困難であることが課題である．既存の手法は，メトリクス間の相関関係やメトリクスと障害間の相関関係にもとづいているが，マイクロサービスにおける主要なデータソースであるログを無視している．この課題に対し，ログの異常検出を組み込んで新たな根本原因のメトリクスを特定する手法を提案した論文がある [12]．この手法では，障害によって発生したシステムログの異常なスコアの変化に応じて，根本原因メトリクスも変化する点に注目しており，2 つの要素から構成されている．1 つ目は，ログ異常検出アルゴリズムによる異常スコアの収集である．2 つ目は，データ拡張を用いた正確で信頼性の高い相関分析による根本原因となるメトリクスの特定である．この 2 つの要素により短時間で根本原因の特定をするが，ログに目立った異常が出ない種類の障害には対応できない点で改善の余地がある．

大規模なソフトウェアシステムにおける，人間のオペレータや管理者による問題診断や根本原因の分析は，ログデータの規模が大きいため複雑で困難である．この課題に対し，ログデータのシステムの障害や異常と直接関連する部分に集中して，診断におけるプロセスの複雑さを軽減する手法を提案している研究がある [13]．この手法では，ユーザまたは自動プロセスによって設定された特定の分析目標と診断の仮説に従いログをフィルタリングするためのフレームワークを提案する．提案するフレームワークは，注釈付のゴールツリーを使用し，特定のシステムの機能を実現するための制約と条件をモデル化する．次に，変換プロセスによって制約と条件がクエリのコレクションにマッピングされる．そして，マッピングしたクエリをログデータを格納するリレーショナルデータベースに適用し，潜在的な意味インデックスを使用して，特定のクエリに最も関連性の高いログを特定する．この提案により診断の時間と複雑さが削減されるが，ゴールツリーの作成と維持に人の手作業が必要であり，構築と維持にコストを要するため，システムの変更に柔軟に対応できない点に改善の余地がある．

サーバの頻繁な障害は企業に対し多大な経済的な損失をもたらす．障害の根本原因の分析は運用や保守において重大な課題となっている．従来の根本原因の分析は KPI メトリクスや依存グラフ，相関分析，イベント因果グラフのような手法が提案されているが，粗い分析粒度や適応の難しさ，データ取得の制約が課題である [14]．この課題に対し，OS ログから障害伝搬チェーンを構築し，サーバ障害の根本原因の分析を自動化する手法を提案している研究がある．OS ログは汎用的かつ容易に取得でき，ソフトウェアの障害の解決に有効な粒度の情報を含んでいるため，サーバ障害の信頼性を向上させる．手法は 3 つの要素で構成されている．1 つ目はログの解析で，ログをテンプレートに変換し，分析を容易にする．2 つ目は障害イベントの識別で，対照学習ベースの階層的なマッチング手法を用いて，障害を特定する．3 つ目は障害伝搬チェーンの構築で，特定された障害イベントから障害伝搬チェーンを構築し，根本原

因を特定する．この提案によりサーバ障害の根本原因の分析を正確に行うことができるが，人間が関与するフィードバックメカニズムを導入しているため，大規模システムへの適用において効率性に改善の余地がある．

3. 提案

提案方式

提案手法では，曜日によってログの傾向が異なる点に着目する．休日は CDSL の活動がなく，平日と休日でログの出力の傾向が異なる．期間を 1 週間にすることで平日と休日の傾向を網羅できるため，使用するログの期間を 1 週間とする．障害の発生時刻から 1 週間前までの Elasticsearch に保存されているログを取得し，タイムスタンプの時間帯ごとコンテナ名別にログ件数を n 分ごとに集計する．ログ件数の集計結果を昇順に並び替える．本番環境のシステムでインシデントが発生する確率は 1%程度である [15]．そのため，提案手法では異常が発生する確率は 1%程度を想定しており，残りの 99%は正常であると前提を置いている．閾値の決定は，99 パーセンタイルを使用し，コンテナ名別に行う．決定した閾値と障害発生時刻のログ件数を比較することで，障害が発生しているコンテナを特定する．図 4 では，n の値は 1 である．

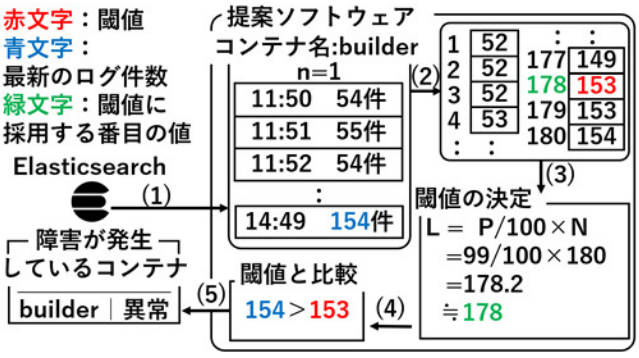


図 4 STNS の builder コンテナで発生した障害を例とした提案ソフトウェアの概要

図 4 は，STNS の builder コンテナで発生した障害を例とした提案ソフトウェアの概要を表している．図 4 は，Elasticsearch，提案ソフトウェア，障害が発生しているコンテナで構成されている．Elasticsearch は，提案ソフトウェアにログを転送する役割を担っている．提案ソフトウェアは，Elasticsearch からログを取得し，ログを集計，昇順に並び替える．99 パーセンタイルを使用して閾値を決定し，直前の時間帯のログ件数と閾値を比較して障害が発生しているコンテナを特定し，出力する．直前の時間帯のログ件数とは，最後に取得した時間帯のログ件数を指しており，図 4 では 14:49 の 154 件が最後に取得したログ件数であり，直前の時間帯のログ件数を表す．障害が発生しているコンテナは，提案ソフトウェアで特定した障害が発生しているコンテナを表示する．提案では，Elasticsearch に保存されている全てのコンテナログに対して処理を行う．図 4 では，処理の流れをわかりやすく示すために，builder コンテナを例としてログの取得から障害が発生しているコンテナを特定するまでの手順を示す．(1) では，提案ソフトウェアが Elasticsearch から，障害の発生時刻の 1 週間前から障害の発生時刻までのログを取得する．図 4 では，11:50 から 14:49 までのログを集計した際の結果を例としている．11:50 のログ件数が 54 件，11:51 は 55 件，11:52 は 54 件であり，14:49 は 154 件となっている．(2) では，n 分ごとに集計したログを昇順に並び替える．図 4 では，例として 1 分ごとに集計したログを昇順に並び替えている．並び替える前のログ件数は 54 件，55 件，54 件，154 件と時系列順に並んだ集計結果を昇順に並び替えている．並び替えた後のログ件数は，1 番目が 52 件，2 番目が 52 件，3 番目が 52 件，4 番目が 53 件，177 番目が 149 件，178 番目が 153 件，179 番目が 153 件，180 番目が 154 件のようにログ件数が小さい順に整列される．四角の枠内の数字がログ件数を表しており，その左隣の数字はログ件数を昇順に並べた際の順番を表している．(3) では，昇順に並び替えたログ件数から 99 パーセンタイルの値を取り，閾値を決定する．閾値を決定する際の式は実装で記述する．L は昇順に並べたデータにおける位置を表しており，図 4 では 178 が該当する．178 は，図 4 の閾値の決定で行った計算の結果である 178.2 を小数第一で四捨五入した値である．P は求めるパーセンタイルの値であり，図 4 では 99 が該当する．N は集計間隔の回数であり，図 4 では 180 が該当する．それぞれの値を式に当てはめ計算した際の解は 178 であり，178 番目の値を 99 パーセンタイルとして採用する．昇順に並べたデータにおける 178 番目の値は 153 であり，閾値は 153 に設定する．(4) では，決定した閾値と直前の時間帯のログ件数を比較し，障害が発生しているコンテナを特定する．直前の時間帯のログ件数が閾値より大きい場合を異常あり，直前の時間帯のログ件数が閾値より小さい場合を異常なしとする．図 4 では，直前の時間帯のログ件数を青文字で表しており，値は 154 である．閾値を赤文字で表しており，値は 153 である．(5) では，直前の時間帯のログ件数が閾値より大きいコンテナ名を障害が発生しているコンテナとして出力する．直前の時間帯のログ件数が 154 であり，閾値が 153 である．直前の時間帯のログ件数が閾値より大きいため，builder コンテナは障害が発生しているコンテナとして出力される．

ユースケース・シナリオ

本稿では，2025 年 5 月 16 日に STNS の builder コンテナで発生した障害をユースケースとする．図 5 は，GitHub，小山さん，佐藤さん，提案ソフトウェア，Elasticsearch，core-server, Pod, git-sync, builder-assets-volume, builder,

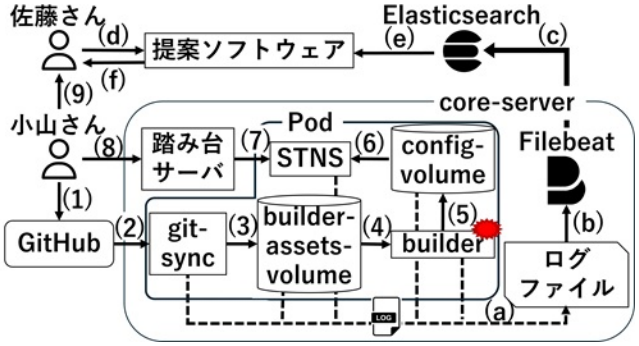


図 5 提案ソフトウェアを使用した際の原因調査の流れ

config-volume，STNS，踏み台サーバ，ログファイル，Filebeat で構成されている．GitHub は，公開鍵を登録するために使用しているウェブサービスである．小山さんは，CDSL の学生であり，障害が発生したことを管理者に報告した人物である．佐藤さんは，CDSL の学生であり，STNS サーバの管理者である．提案ソフトウェアは，障害の原因発生箇所を特定する際に使用するソフトウェアである．Elasticsearch は，Filebeat で収集したログを保存するオープンプラットフォームである．core-server は，Pod やログファイルが動作するサーバである．Pod は，コンテナやストレージを動作させるための実行環境である．git-sync は，core-server の Pod 内にあるコンテナの 1 つである．builder-assets-volume は，core-server の Pod 内にあるストレージの 1 つである．builder は，core-server の Pod 内にあるコンテナの 1 つであり，ユースケースで障害が発生した箇所である．config-volume は，core-server の Pod 内にあるストレージの 1 つである．STNS は，core-server の Pod 内にあるコンテナの 1 つである．踏み台サーバは，CDSL のサーバにアクセスする際の中継役のサーバである．ログファイルは，core-server 内のログをファイル形式で記録している．Filebeat は，ログファイル内のログを収集する．

図 5 の流れについて説明する．(1) で小山さんが GitHub に公開鍵の登録を行う．(2) から (6) の順番で，GitHub に登録した公開鍵がダウンロードされ，STNS で管理される．(7) で小山さんが踏み台サーバに SSH 接続を行い，(8) で踏み台サーバが STNS に保存されている公開鍵を参照する．(9) で小山さんが踏み台サーバに SSH 接続できなかったことを佐藤さんに報告する．(a) で core-server の Pod 内にあるコンテナとストレージのログが，ログファイルに記録される．(b) でログファイルに記録されたログは Filebeat で収集される．(c) で Filebeat で収集されたログは Elasticsearch に保存される．(d) で佐藤さんが，障害の原因箇所を特定するために提案ソフトウェアを実行する．(e) で提案ソフトウェアが Elasticsearch からログを取得する．(f) で提案ソフトウェアを実行し，出力された原因の発生箇所を佐藤さんが確認する．

4. 実装

提案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．ソフトウェアのライブラリと処理の過程を説明する．提案ソフトウェアでは，Elasticsearch ライブラリを用いてログを取得し，データ処理や分析のために Python の pandas，numpy，pytz，tabulate ライブラリをインストールして使用した．

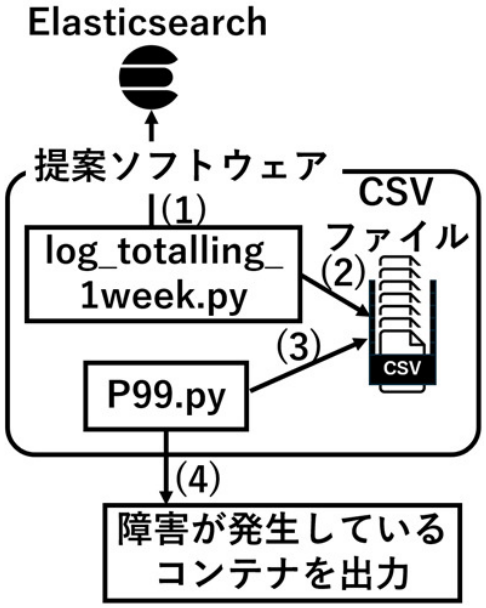


図 6 提案ソフトウェアの構成図

図 6 は，提案ソフトウェアの構成図を表している．図 6 は，Elasticsearch，提案ソフトウェアで構成されている．Elasticsearch は，CDSL のシステムのログを保存しているソフトウェアを表す．図 6 の流れについて説明する．提案ソフトウェアは，ログ件数を集計し，昇順に並び替え，99 パーセンタイルを用いて閾値を決定する．決定した閾値と直前の時間帯のログ件数を比較し，障害が発生しているコンテナを特定し，出力する．(1) で提案ソフトウェアが Elasticsearch からログを取得する．(2) で log_totalling_1week.py がログの集計を行い，集計結果を CSV ファイルで出力する．(3) で log_totalling_1week.py で出力された CSV ファイルをもとに P99.py で 99 パーセンタイルの閾値を決め，直前の時間帯のログ件数と閾値を比較し，障害が発生しているコンテナを特定する．(4) で P99.py で特定した障害が発生しているコンテナを出力する．

log_totalling_1week.py

log_totalling_1week.py では，Elasticsearch から取得したログを log_totalling_1week.py の process_and_export_chunk() 関数により 1 分間隔でコンテナ名を集計する．このプログラムでは，ログのタ

イムスタンプを JST に変換し，1 分ごとに時間帯を丸め，ログ件数を算出する．また，ログに含まれる kubernetes.container.name をもとにコンテナ名ごとに集計を行う．ログの取得は，get_logs_from_elasticsearch() 関数により実行し，取得する対象のログは Elasticsearch の beat-インデックスの全てのコンテナログである．ログの取得期間は，1 週間とし，_main_ブロック内の while ループで 1 日ごとに期間を分割して Elasticsearch からログを取得する．取得した日ごとのログを process_and_export_chunk() 関数で集計し，CSV ファイルで結果を出力する．CSV ファイルの出力は日ごとに分割されたログの終了時刻をファイル名とし，出力する．YYYYMMDD-HHMM の形式で整形された文字列で終了時刻を表し，終了時刻が 2025 年 1 月 1 日の 12 時 30 分の場合，20250101-1230 の形式で CSV ファイルが出力される．

P99.py

P99.py では，log_totalling_1week.py で出力した CSV ファイルの内容をもとにログ件数の並び替えと閾値の決定，直前の時間帯のログ件数と閾値を比較，障害が発生しているコンテナの出力を行う．まず，analyze_logs_for_anomaly() 関数で，ログ件数をコンテナごとに抽出し，ログ件数を昇順に並び替える．この処理は，99 パーセンタイルで閾値を決定する際に計算しやすくするために行う．データ件数が 10 未満の場合，警告メッセージを付与して計算結果の信頼性が低いことを出力する．

次に，並び替えの結果をもとに変数の past_counts から 99 パーセンタイルを計算し閾値を決定する．閾値を決定する際の式は式 (1) に示す．

$$L = \frac{99}{100} \times N \tag{1}$$

式 (1) の値について以下に記述する．

- L: 集計したログ件数を昇順に並べ替えた際に 99 パーセンタイルが何件目かを表す
- N: 集計間隔の回数であり，ログ件数の合計数である

式 (1) により，昇順に並び替えたデータの何番目の値を閾値にするか決定する．L は集計したログ件数を昇順に並べた際に位置する値である．L の値が整数でない場合，四捨五入して得られた値を 99 パーセンタイルの閾値とする．N は集計間隔の回数であり，ログ件数の合計数である．1 分間隔の集計をした際に全部で何個のデータがあったかを示す．

そして，直前の時間帯のログ件数と閾値を比較し，障害が発生しているコンテナを特定する．全ての取得したコンテナログに対し処理を行い，直前の時間帯のログ件数が閾値より大きい場合，異常ありと判断する．直前の時間帯のログ件数が閾値より小さい場合，異常なしと判断する．

最後に，異常ありと判断したコンテナログのコンテナ名を出力する．コンテナ名，直前の時間帯のログ件数，99 パーセンタイルの閾値，判定結果を見出しとし，表形式でコンソールに出力される．

5. 評価実験

実験環境

実験環境として，仮想マシン (以後 VM とする) を 1 台用意した．VM には Ubuntu24.04 をインストールした．スペックは，vCPU を 4Core，メモリを 8GB，ストレージを 30GB とした．実験で使用した Elasticsearch のバージョンは 8.5.1 である．図 7 に，評価実験の実験環境を示す．

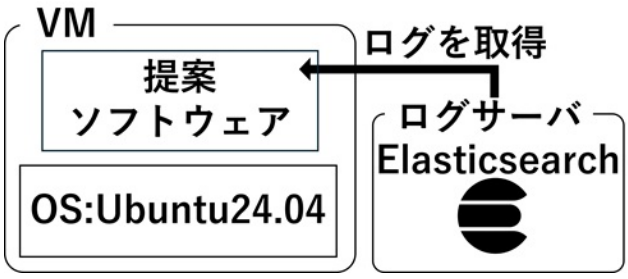


図 7 評価実験の実験環境

実験のデータセット

CDSL で運用している Elasticsearch から実験の内容に合わせて提案ソフトウェアでログを取得する．実験は 5 種類行い，表 1 に各実験で使用したログの検索期間を示す．

表 1 各実験で使用したログの検索期間

実験番号	検索期間	ログの件数
1,2	2025 年 5 月 9 日 14 時 45 分〜 2025 年 5 月 16 日 14 時 45 分	30,714,020 件
3	2025 年 5 月 8 日の 14 時 45 分〜 2025 年 5 月 15 日 14 時 45 分	30,405,005 件
4	2025 年 5 月 3 日の 13 時 38 分〜 2025 年 5 月 10 日 13 時 38 分	32,166,628 件
5	2025 年 5 月 16 日の 11 時 45 分〜 2025 年 5 月 16 日 14 時 45 分	622,317 件

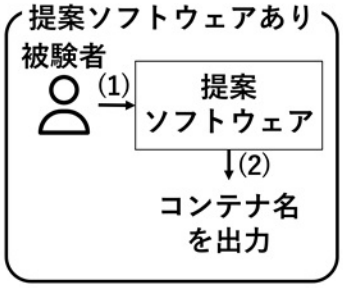
実験 1 と実験 2 では，STNS の builder コンテナで起きた障害をユースケースとしているため，2025 年 5 月 9 日 14 時 45 分から 2025 年 5 月 16 日 14 時 45 分のログを取得した．この期間に出力された 30,714,020 件のログを使用した．実験 3 では，誤検知の測定のため，STNS の builder コンテナで障害が発生していない期間である 2025 年 5 月 8 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分のログを取得した．この期間に出力された 30,405,005 件のログを使用した．実験 4 では，ログの集計の終了時刻を直前の時間帯のログ件数に設定し，ログ件数が閾値を上回った際

の結果を測定するため，2025 年 5 月 9 日の 13 時 38 分から 2025 年 5 月 16 日の 13 時 38 分のログを取得した．この期間に出力された 32,166,628 件のログを使用した．実験 5 では，ログの分布を測定するため，2025 年 5 月 16 日の 11 時 45 分から 14 時 45 分のログを取得した．この期間に出力された 622,317 件のログを使用した．Elasticsearch には Kubernetes クラスタ上で稼働する Pod 内のコンテナログが保存されている．例として，ceph-exporter や filebeat，jaeger，stns-server があり，これらは CDSL で共通で使用されている core サーバ上で動作しているコンテナである．

提案ソフトウェアを使用した際に障害が発生しているコンテナの検出の可否の測定と，調査に要する時間の測定で評価する．実験は 5 種類行い，主に障害が発生しているコンテナの検出の可否と障害の原因調査に要する時間を測定した．実験 1 では，提案ソフトウェアありで原因調査に要した時間を測定する．実験 2 では，実際に発生した障害を検出できたかを測定する．実験 3 では，実際に発生していない時に誤検知を起こさないかを測定する．実験 4 では，ログの件数が閾値を上回っていた時刻を終了時刻にした際の結果を測定する．実験 5 では，集計間隔を変更し，ログ件数の分布を測定する．

(実験 1) 実験の方法

実験 1 では，実験の被験者に手順書を渡し，障害の原因調査を行う際の情報を事前に共有して行った．図 8 は，実



- 特定した障害の発生箇所の検出の可否を測定
- 障害の原因特定までに要した時間を測定

図 8 実験 1 のシナリオ

験 1 のシナリオを表す．提案ソフトウェアありの実験は被験者 A と提案ソフトウェアで構成されている．被験者 A は，提案ソフトウェアを使用して障害の原因を特定する人を表す．提案ソフトウェアは，被験者 A が入力した検索日時をもとにログを集計し，障害の発生箇所を特定し，コンテナ名を出力する．提案ソフトウェアありで特定した障害の発生箇所の検出の可否と障害の原因特定までに要した時間を測定し，評価する．図 8 の手順について説明する．(1) で被験者 A が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の集計を行い，障害の発生箇所を特定し，コンテナ名を出力する．これらの手順を行い，被験者 A が特定した障害の発生箇所の検出の可否の測定と障害の発生箇所を特定するまでに要した時間の測定をする．

(実験 2，実験 3，実験 4，実験 5) 実験の方法

実験 2，実験 3，実験 4，実験 5 は，実験の内容に合わせて検索期間を変更し，提案ソフトウェアを実行させ，検出の可否や誤検知を測定する．実験 2，実験 3，実験 4，実験 5 では，提案ソフトウェアと Elasticsearch を使用する．提案ソフトウェアで，実験 3 と実験 4 と実験 5 で誤検知の測定や検出の可否を測定する．Elasticsearch には，提案ソフトウェアが誤検知や検出の可否を測定する際に，使用するログが保存されている．実験 2，実験 3，実験 4，実験 5 の手順について説明する．提案ソフトウェアが Elasticsearch に接続し，検索期間のログを取得する．取得した検索期間のログを提案ソフトウェアで集計し，誤検知や検出の可否の測定を行い，結果を出力する．

それぞれの実験の内容と，検索期間について説明する．実験 2 では，検出の可否を測定する．STNS の builder コンテナで障害が発生した 2025 年 5 月 16 日の 14 時 45 分が検索期間の終了時刻になるようにするため，2025 年 5 月 9 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分を検索期間とした．実験 3 では，誤検知の測定を行うため，STNS の builder コンテナで障害が発生していない期間である 2025 年 5 月 8 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分を検索期間とした．実験 4 では，ログの集計の終了時刻を直前の時間帯のログ件数に設定し，直前の時間帯のログ件数が閾値を上回った際の結果を測定するため，2025 年 5 月 9 日の 13 時 38 分から 2025 年 5 月 16 日の 13 時 38 分を検索期間とした．実験 5 では，ログの分布を測定するため，2025 年 5 月 16 日の 11 時 45 分から 14 時 45 分を検索期間とした．

(実験 1) 時間の測定

2025 年 5 月 16 日の 14 時 45 分に CDSL で起きた STNS の builder コンテナの障害をユースケースとして提案ソフトウェアありの障害特定までに要した時間を計測した．提案ソフトウェアありの実験を CDSL の学生である山崎拓海さん (以後山崎さんとする) を被験者として実施した．

提案ソフトウェアありの実験を 2025 年 11 月 5 日の 16 時 25 分から 16 時 59 分に行った．この実験では，提案ソフトウェアの検索期間を 1 週間，集計間隔を 1 分に指定して行った．この実験では，実際の障害が発生した状況に近づけるため手順書を渡し，内容の確認から実験を行ってもらった．手順書の内容を以下に示す．

- 2025 年 5 月 16 日 14 時 45 分に小山さんが佐藤さんに公開鍵を GitHub に登録したが，SSH 接続できない問

い合わせをした
- CDSL では踏み台サーバに SSH する際の公開鍵を GitHub に登録する
- STNS サーバの構成図
- GitHub リポジトリにある user.csv に公開鍵を追加することで，踏み台サーバに登録した公開鍵でログインできるようになる
- STNS サーバで公開鍵を管理している
- GitHub に登録された公開鍵は STNS に 1 分ごとにダウンロードされる
- 調査対象のサーバが STNS サーバであること
- 障害が発生しているコンテナの特定を行ってほしいこと
- 調査結果をまとめるため，調査中の画面を Zoom で録画をすること
- 調査の結果から障害が発生しているコンテナ，エラーログが出ていた原因，エラーログが発生していた日時を手順書にまとめること

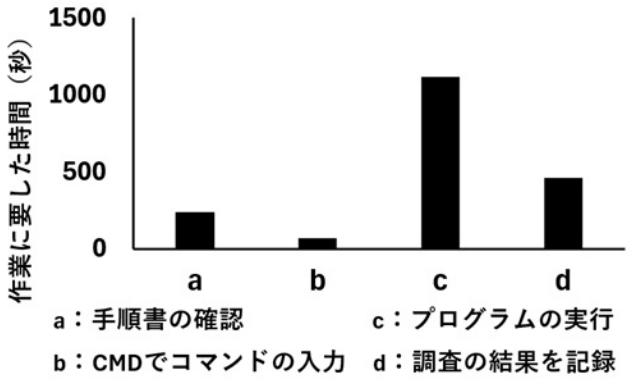


図 9 評価実験で各作業に要した時間の分類

図 9 は，評価実験で各作業に要した時間の分類を表す．Y 軸は，障害が発生しているコンテナを特定する際の各作業に要した時間を表し，単位は秒をとる．X 軸は，障害が発生しているコンテナを特定する際に行った作業内容を表す．a が手順書の確認，b が CMD でコマンドの入力，c がプログラムの実行，d が調査の結果を記録である．a の手順書の確認に 241 秒，b の CMD でコマンドの入力に 72 秒，c のプログラムの実行に 1118 秒，d の調査の結果を記録に 461 秒要していた．提案ソフトウェアありの評価実験で要した全体の作業時間は 241 + 72 + 1118 + 461 により 1892 秒である．この 1892 秒は，障害が発生しているコンテナを特定する際に行った作業内容の時間を全て合計した時間である．提案ソフトウェアを使用した場合，人が行う必要がある作業に要した時間は 241 + 72 + 461 により 774 秒であり，全体の作業時間の約 41%を占めている．一方，提案ソフトウェアありの自動化が可能な作業に要した時間は 1118 秒であり，全体の作業時間の約 59%を占めている．この結果から，プログラムの実行時間が長いことがわかる．これは，Elasticsearch から条件を満たすログを全て取得しているため，実行時間が長くなっている．事前にログの件数を集計して保存しておくことで，プログラムの実行時間の短縮が可能である．

(実験 2) 検出の可否

実験 2 では，障害が発生した期間に誤検知が発生しなかった．2025 年 5 月 16 日の 14 時 45 分に CDSL で起きた STNS の builder コンテナの障害をユースケースとする．検索期間と集計間隔を変更し，障害が発生しているコンテナを検出できたかを 2025 年 11 月 3 日の 14 時から 2025 年 11 月 3 日の 17 時 33 分に測定した．集計間隔を 1 分，5 分，10 分，15 分，30 分の 5 パターンで測定し，結果を表 2 で示す. 表の見出しは検索期間,1 分，5 分，10 分，15 分，30 分である．検索期間はログの検索対象の期間を表している．また，この期間内に障害が発生していたコンテナの数は 1 つであり，コンテナ名は builder である．1 日ごとの検索期間と集計間隔を変更し，対象の期間に発生している障害のコンテナ数に対し，検出数がいくつかを確かめ，検出できたか確認した．ログ件数は，検索期間とコンテナ名でフィルターした際のログ件数である．検索期間が 1 日の場合は，2025 年 5 月 15 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分であり，ログの件数は 77,817 件である．2 日の場合は，2025 年 5 月 14 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分であり，ログの件数は 155,678 件である．3 日の場合は，2025 年 5 月 13 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分であり，ログの件数は 233,799 件である．4 日の場合は，2025 年 5 月 12 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分であり，ログの件数は 311,288 件である．5 日の場合は，2025 年 5 月 11 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分であり，ログの件数は 388,991 件である．6 日の場合は，2025 年 5 月 10 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分であり，ログの件数は 466,688 件である．7 日の場合は，2025 年 5 月 9 日の 14 時 45 分から 2025 年 5 月 16 日の 14 時 45 分であり，ログの件数は 544,382 件である．1 分は，集計間隔を 1 分で実験を行った際の結果を示す．5 分は，集計間隔を 5 分で実験を行った際の結果を示す．10 分は，集計間隔を 10 分で実験を行った際の結果を示す．15 分は，集計間隔を 15 分で実験を行った際の結果を示す．30 分は，集計間隔を 30 分で実験を行った際の結果を示す．提案ソフトウェアで判断する際は以下の 4 つで判断する．

- TP(True Positive) は，実際に障害が発生している際に提案ソフトウェアが異常と判断したことを表す
- TN(True Negative) は，実際に障害が発生していない

際に提案ソフトウェアが正常と判断したことを表す

- FP(False Positive) は，実際に障害が発生していない際に提案ソフトウェアが異常と判断したことを表す
- FN(False Negative) は，実際に障害が発生している際に提案ソフトウェアが正常と判断したことを表す

表 2 実験 2 の結果と検索期間ごとのログ件数

検索期間	集計間隔					ログ件数
	1 分	5 分	10 分	15 分	30 分	
1 日	TP	TP	FN	TP	FN	77,817
2 日	TP	TP	FN	TP	FN	155,678
3 日	TP	TP	FN	TP	FN	233,799
4 日	TP	TP	FN	TP	FN	311,288
5 日	TP	TP	FN	TP	FN	388,991
6 日	TP	TP	FN	TP	FN	466,688
7 日	TP	TP	FN	TP	FN	544,382

表 3 に実験 2 の各集計間隔の閾値を示す．集計間隔が 1 分，5 分，15 分では閾値が同じであり，10 分と 30 分では検索期間によって閾値が異なった．閾値が同じ場合は 99 パーセンタイルで閾値を決定する際，特定のログ件数が高頻度で出力されているためである．例えば，集計間隔が 1 分の場合，検索期間を変更してもログ件数が 56 件であり，高頻度で出力されていたため，99 パーセンタイルに該当する値がどの検索期間でも同じとなった．閾値が異なる場合は検索期間を広げ，集計した際にログ件数の分布が変化するためである．提案ソフトウェアでは，ログ件数を一定の間隔で集計し，時間帯ごとにログ件数を丸める処理を行っている．そのため，検索期間と集計間隔が広がると，集計対象となるログの数がわずかに変化し，ログ件数の分布も変化する．この結果，99 パーセンタイルに該当する値が検索期間ごとに異なる．

表 3 実験 2 の各集計間隔の閾値

検索期間	集計間隔				
	1 分	5 分	10 分	15 分	30 分
1 日	56	276	548	822	1637
2 日	56	276	549	822	1639
3 日	56	276	549	822	1644
4 日	56	276	550	822	1645
5 日	56	276	550	822	1644
6 日	56	276	550	822	1644
7 日	56	276	550	822	1644

集計間隔が 1 分の場合，1 日から 7 日までの検索期間において結果が TP となった．集計間隔を 1 分とした場合の直前の時間帯は，14 時 44 分 0 秒から 14 時 44 分 59 秒である．表 4 に集計期間が 1 分の各検索期間の検出の可否の結果を示す．1 日から 7 日までの各検索期間において直前のログ件数はいずれも 154 件であり，閾値のログ件数はいずれも 56 件である．直前のログ件数の 154 件は閾値のログ件数の 56 件を上回るため，提案方式で異常と判定された．また，実際の障害は “発生した” であるため TP になる．提案方式の閾値の決定には 99 パーセンタイルを使用している．検索期間の 1 日から 7 日までのいずれでもログ件数の分布が変わらないため，99 パーセンタイルの値に変化がなかった．その結果，検索期間の 1 日から 7 日までのいずれでも閾値が同一であった．

表 4 集計期間が 1 分の各検索期間の検出の可否の結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	56	超える	異常	発生した	TP
2 日	56	超える	異常	発生した	TP
3 日	56	超える	異常	発生した	TP
4 日	56	超える	異常	発生した	TP
5 日	56	超える	異常	発生した	TP
6 日	56	超える	異常	発生した	TP
7 日	56	超える	異常	発生した	TP

集計間隔が 5 分の場合，1 日から 7 日までの検索期間において結果が TP となった．集計間隔を 5 分とした場合の直前の時間帯は，14 時 40 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 409 件である．閾値のログ件数は 276 件であり，1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を上回る結果だったため，提案ソフトウェアで TP と判断された．

集計間隔が 10 分の場合，1 日から 7 日までの検索期間において結果が FN となった．集計間隔を 10 分とした場合の直前の時間帯は，14 時 40 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 409 件である．提案ソフトウェアでは，開始時刻から集計間隔の時間ごとにログを集計している．実験 2 では検索期間の終了時刻が 14 時 45 分であり，直前の時間帯は集計間隔が 10 分よりも短い 5 分間の集計となったため，14 時 40 分 0 秒から 14 時 44 分 59 秒が直前の時間帯となった．表 5 に集計期間が 10 分の各検索期間の検出の可否の結果を示す．1 日から 7 日までの各検索期間において直前のログ件数はいずれも 409 件である．閾値のログ件数は 1 日では 548 件，2 日から 3 日では 549 件，4 日から 7 日では 550 件である．直前のログ件数の 409 件は閾値のログ件数の 550 件を下回るため，提案方式での異常判定は正常となった．また，実際の障害は “発生した” であるため FN になる．提案方式の閾値の決定には 99 パーセンタイルを使用している．検索期間の 1 日から 7 日までのいずれでもログ件数の分布が変わらないため，99 パーセンタイルの値に変化がなかった．その結果，検索期間の 1 日から 7 日までのいずれでも閾値が同一であった．

集計間隔が 15 分の場合，検索期間が 1 日から 7 日にお

表 5　集計期間が 10 分の各検索期間の検出の可否の結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	548	超えない	正常	発生した	FN
2 日	549	超えない	正常	発生した	FN
3 日	549	超えない	正常	発生した	FN
4 日	550	超えない	正常	発生した	FN
5 日	550	超えない	正常	発生した	FN
6 日	550	超えない	正常	発生した	FN
7 日	550	超えない	正常	発生した	FN

いて結果が TP となった．集計間隔を 15 分とした場合の直前の時間帯は，14 時 30 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 946 件である．閾値のログ件数は，検索期間が 1 日と 2 日の場合は 822 件，3 日から 7 日の場合は 824 件であった．検索期間が 1 日から 7 日で直前の時間帯のログ件数が閾値のログ件数を上回る結果だったため，提案ソフトウェアで TP と判断された．

集計間隔が 30 分の場合，1 日から 7 日までの検索期間において結果が FN となった．集計間隔を 30 分とした場合の直前の時間帯は，14 時 30 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 946 件である．検索期間の終了時刻が 14 時 45 分であり，提案ソフトウェアでは，開始時刻から集計間隔の時間ごとにログを集計している．そのため，直前の時間帯は集計間隔が 30 分よりも短い 15 分間の集計となり，14 時 30 分 0 秒から 14 時 44 分 59 秒が直前の時間帯となった．閾値のログ件数は，検索期間が 1 日の場合は 1637 件，2 日の場合は 1639 件，3 日と 5 日と 6 日と 7 日の場合は 1644 件，4 日の場合は 1645 件であった．1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を下回る結果だったため，提案ソフトウェアで正常と判断された．

集計間隔が 1 分と 5 分と 15 分では全て TP であり，実際に障害が発生している際に提案ソフトウェアが異常と判断した．集計間隔が 10 分と 30 分では全て FN であり，実際に障害が発生している際に提案ソフトウェアが正常と判断した．集計間隔によって結果が異なる理由は，集計間隔が変わることで閾値と直前のログの件数が変わるからである．検索期間が 1 日から 7 日のいずれでも集計間隔ごとの結果は変化しなかった．検索期間が長くなった場合にも直前の時間帯は変わらない．また，検索期間が長くなった場合に，閾値は表 4 の結果が示すとおり変化がほとんどない．

(実験 3) 障害が発生していない場合の誤検知

実験 3 では，障害が発生していない期間に誤検知が発生するかを検証する．STNS の builder コンテナにおいて障害が発生していない期間を対象とし，提案ソフトウェアにおける誤検知が発生するかの測定を行った．この実験では，実際に障害が発生していない際に提案ソフトウェアが異常と判断した場合を FP(False Positive) とし，実際に障害が発生していない際に提案ソフトウェアが正常と判断した場合を TN(True Negative) とする．検索期間の終了日時を 2025 年 5 月 15 日の 14 時 45 分に固定し，開始日時を終了日時から 1 日前から 7 日前までに変更して実験を行った．以下にそれぞれの検索期間の日時を記す．

- 1 日の場合は，2025 年 5 月 14 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分である．
- 2 日の場合は，2025 年 5 月 13 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分である．
- 3 日の場合は，2025 年 5 月 12 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分である．
- 4 日の場合は，2025 年 5 月 11 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分である．
- 5 日の場合は，2025 年 5 月 10 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分である．
- 6 日の場合は，2025 年 5 月 9 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分である．
- 7 日の場合は，2025 年 5 月 8 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分である．

判定結果は，誤検知が起きるかを取得する．検索期間が 1 日から 7 日までのそれぞれ集計間隔 (1 分，5 分，10 分，15 分，30 分) において誤検知が発生しなかった．

集計間隔が 1 分の場合，1 日から 7 日までの検索期間において誤検知は起きなかった．集計間隔を 1 分とした場合の直前の時間帯は，14 時 44 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 54 件である．閾値のログ件数は 56 件であり，1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を下回る結果だった．そのため，提案ソフトウェアで誤検知は起きなかった．

集計間隔が 5 分の場合，1 日から 7 日までの検索期間において誤検知は起きなかった．集計間隔を 5 分とした場合の直前の時間帯は，14 時 40 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 272 件である．閾値のログ件数は 276 件であり，1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を下回る結果だった．そのため，提案ソフトウェアで誤検知は起きなかった．

集計間隔が 10 分の場合，1 日から 7 日までの検索期間において誤検知は起きなかった．集計間隔を 10 分とした場合の直前の時間帯は，14 時 40 分 00 秒から 14 時 44 分 59 秒であり，ログ件数は 272 件である．実験 2 と同様に検索期間の終了時刻が 14 時 45 分であり，提案ソフトウェアでは，開始時刻から集計間隔の時間ごとにログを集計している．そのため，直前の時間帯は集計間隔が 10 分よりも短い 5 分間の集計となり，14 時 40 分 0 秒から 14 時 44 分 59 秒が直前の時間帯となった．閾値のログ件数は，検索期間

が 1 日の場合 549 件であり，2 日から 7 日の場合 550 件であった．1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値を下回る結果となり，提案ソフトウェアで誤検知は起きなかった．

集計間隔が 15 分の場合，1 日から 7 日までの検索期間において誤検知は起きなかった．集計間隔を 15 分とした場合の直前の時間帯は，14 時 30 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 811 件である．閾値のログ件数は，検索期間が 1 日の場合 822 件であり，2 日から 7 日の場合 824 件であった．1 日から 7 日までの検索期間で直前の時間帯のログ件数が閾値のログ件数を下回る結果だった．そのため，提案ソフトウェアで誤検知は起きなかった．

集計間隔が 30 分の場合，1 日から 7 日までの検索期間において誤検知は起きなかった．集計間隔を 30 分とした場合の直前の時間帯は，14 時 30 分 0 秒から 14 時 44 分 59 秒であり，ログ件数は 811 件である．検索期間の終了時刻が 14 時 45 分であり，提案ソフトウェアでは，開始時刻から集計間隔の時間ごとにログを集計している．そのため，直前の時間帯は集計間隔が 30 分よりも短い 15 分間の集計となり，14 時 30 分 0 秒から 14 時 44 分 59 秒が直前の時間帯となった．閾値のログ件数は，検索期間が 1 日の場合は 1639 件，2 日と 5 日と 6 日と 7 日の場合は 1644 件，3 日と 4 日の場合は 1645 件であった．1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を下回る結果だった．そのため，提案ソフトウェアで誤検知が起きなかった．

提案ソフトウェアが誤検知を起こさないかの測定では，集計間隔を 1 分，5 分，10 分，15 分，30 分に変更して行った．1 日から 7 日の検索期間とそれぞれの集計間隔で，直前の時間帯のログ件数が閾値を下回る結果であり，提案ソフトウェアで誤検知が起きなかった．これらの結果から，2025 年 5 月 8 日の 14 時 45 分から 2025 年 5 月 15 日の 14 時 45 分の 1 週間では提案ソフトウェアで誤検知が起きていないことが確認できる．

(実験 4) ログ件数が増える兆候の検知

ログ件数が閾値を上回った時刻を検索期間の終了時刻に設定し，ログ件数が増える兆候がみられた場合に提案ソフトウェアが異常を検知できるか測定した．対象のコンテナを builder とし，集計間隔が 1 分で検索期間が 2025 年 5 月 9 日の 13 時 38 分から 5 月 16 日の 13 時 38 分の時の閾値は 56 であった．集計間隔が 1 分の場合に，ログ件数が閾値を上回っていた時刻は 2025 年 5 月 10 日の 13 時 38 分であり，閾値は 60 であった．この時刻が直前の時間帯のログ件数となるように指定し，閾値を上回った際の測定結果を確かめるため，各集計間隔で測定を行った．検出の可否の測定と同じ条件にするため，検索期間を 2025 年 5 月 3 日の 13 時 38 分から 2025 年 5 月 10 日の 13 時 38 分の 1 週間とし，集計間隔は 1 分，5 分，10 分の 3 パターンで行った．判定結果は，FP(False Positive) と FN(False Negative) で判断する．対象の期間で障害に関係のないログが誤って異常と判断された場合を FP とし，障害に関係のないログが誤って異常だと判断されていない場合を FN とする．集計間隔が 1 分の場合に，検索期間の 1 日から 7 日のいずれでも FP となった．集計間隔が 5 分，10 分の場合には，検索期間が 1 日から 7 日のいずれでも FN となった．

集計間隔が 1 分の場合，1 日から 7 日までの検索期間において結果が FP となった．集計間隔を 1 分とした場合の直前の時間帯は，13 時 38 分 0 秒から 13 時 38 分 59 秒であり，ログ件数は 60 件である．閾値のログ件数は 56 件であり，1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を上回る結果だった．そのため，提案ソフトウェアで判定結果は FP と判定され，異常が起きていると判断した．

集計間隔が 5 分の場合，1 日から 7 日までの検索期間において結果が FN となった．集計間隔を 5 分とした場合の直前の時間帯は，13 時 35 分 0 秒から 13 時 38 分 59 秒であり，ログ件数は 223 件である．本来は実験で開始から終了までの集計間隔を 5 分で実験を計画していた．提案ソフトウェアでは，開始時刻から集計間隔 (例えば 5 分) ごとにログを集計している．開始時刻から集計間隔でログを分割した結果，直前の時間帯が集計間隔よりも短い 13 時 35 分 0 秒から 13 時 38 分 59 秒までの 4 分間となった．そのため，この実験では直前の時間帯の集計間隔が 4 分となっている．閾値のログ件数は，検索期間が 1 日から 5 日の場合 276 件であり，6 日と 7 日の閾値が 275 件であった．1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を下回る結果だった．そのため，提案ソフトウェアで判定結果は FN と判定され，異常が起きていないと判断した．

集計間隔が 10 分の場合，1 日から 7 日までの検索期間において結果が FN となった．集計間隔を 10 分とした場合の直前の時間帯は，13 時 30 分 0 秒から 13 時 38 分 59 秒であり，ログ件数は 495 件である．集計間隔が 5 分の場合と同様に，本来は実験で開始から終了までの集計間隔を 10 分で実験を計画していた．開始時刻から集計間隔でログを分割した結果，直前の時間帯が集計間隔よりも短い 13 時 30 分 0 秒から 13 時 38 分 59 秒までの 9 分間となった．そのため，この実験では直前の時間帯の集計間隔が 9 分となっている．閾値のログ件数は，検索期間が 1 日から 4 日の場合 550 件であり，5 日から 7 日の閾値が 549 件であった．1 日から 7 日までの検索期間で，直前の時間帯のログ件数が閾値のログ件数を下回る結果だった．そのため，提案ソフトウェアで判定結果は FN と判定され，異常が起きていないと判断した．

ログ件数が閾値を上回っていた時刻である 13 時 38 分を

終了時刻として測定した結果，集計間隔が 1 分の場合は異常ありと判定され，それ以外の 4 分，9 分の集計間隔では異常なしと判定された．提案方式では，集計間隔ごとにログ件数を集計しており，閾値の決定にも集計間隔の期間あたりに含まれるログ件数を使用している．そのため，5 分や 10 分のような直前の時間帯が集計間隔の時間よりも短い区間の場合，集計するログの件数が少なくなってしまい，異常なしと判定された．表 6 に直前の時間帯が集計間隔の時間よりも短い区間の場合の一部の時間帯の集計結果を示す．時間帯が 2025 年 5 月 10 日 13 時 10 分から 13 時 19 分

表 6 集計期間が 10 分の一部の時間帯の集計結果

時間帯	ログ件数 [件]
2025 年 5 月 10 日 13 時 10 分から 13 時 19 分	542
2025 年 5 月 10 日 13 時 20 分から 13 時 29 分	545
2025 年 5 月 10 日 13 時 30 分から 13 時 38 分	495

までの 10 分間でログ件数は 542 件，2025 年 5 月 10 日 13 時 20 分から 13 時 29 分までの 10 分間でログ件数は 545 件である．一方，2025 年 5 月 10 日 13 時 30 分から 13 時 38 分の 9 分間であり，ログ件数は 495 件である．このように，ログ件数の出力傾向が一定であっても，集計間隔よりも短い区間ではログ件数が少なくなる．提案ソフトウェアで閾値を決定する際，集計間隔ごとに算出したログ件数の一覧から 99 パーセンタイルを求めている．そのため，2025 年 5 月 10 日 13 時 30 分から 13 時 38 分のように集計間隔が短い区間では，ログ件数が増加している場合であっても閾値を超えず，障害を検知できない．

(実験 5) ログ件数の分布

実験 5 では，集計間隔の適切な値を検証するために，集計間隔を変えてログ件数の分布を求めた．2025 年 5 月 16 日に CDSL の STNS サーバの builder コンテナで起きた障害をユースケースとして，出力されるログ件数をコンテナ名ごとに時間帯別に集計した．ログの検索期間を 3 時間として，集計間隔を 1 分，5 分，10 分，15 分，30 分の 5 パターンで集計した．集計間隔を 1 分，5 分，10 分，15 分，30 分の 5 パターンで集計した理由を説明する．1 分，5 分，30 分は監視や分析の際に人が直感的に判断しやすい時間幅であり，区間を分ける際に利用される代表的な時間幅であるため，ログ件数の分布を集計する際に選定した．また，集計間隔が 1 分の場合は，突発的なログの増加を最も細かく捉えることができ，30 分の場合は，長時間の傾向を捉えられる．10 分，15 分は 5 分の 2 倍，3 倍の数字であり，5 分から 30 分の間の時間幅が空いてしまうため，その中間的な挙動を確認する目的で選定した．これにより，短期間から長期間にかけてのログ件数の分布を把握し，分析することができるため，集計間隔を 1 分，5 分，10 分，15 分，30 分の 5 パターンで集計した．

STNS サーバの builder コンテナで障害が発生した際に，エラーログが確認できた時間が 2025 年 5 月 16 日の 14 時 43 分であり，検索期間はこの時間帯を含むように設定した．また，この時間帯にログを出力していたコンテナは，author-app-container，author-mongo-container，builder，cadvisor，ceph-exporter，coredns，csi-snapshotter，expand-bluefs，filebeat，front-app-container，fulltext-app-container，istio-proxy，jaeger，kea，metrics，mgr，mon，osd，paper-app-container，paper-mongo-container，rook-ceph-operator，stats-app-container，stats-mongo-container，stns-server，watch-active である．これらのコンテナのログが時間帯別にどの程度出力されているかを確認し，コンテナごとにログの出力の分布や傾向の違いを捉え，明らかにした．集計の結果から，ログの出力の傾向には，一定量のログが継続的に出力されているパターンと，ログの出力に変動があるパターンの 2 種類があることが明らかとなった．各コンテナのログ件数は階級幅を 5 件ごとに区切って集計したが，折れ線グラフでは視認性を考慮し，階級の表示を 50 件刻みや 250 件刻みに変更している．また，一定量のログが継続的に出力されていた author-app-container を例とし，ログの出方に変動があった filebeat を例として，折れ線グラフの特徴を説明する．

集計間隔を 1 分とした場合の author-app-container のログの分布を図 10 に示す．図 10 は，検索期間が 3 時間で集計間隔が 1 分間の author-app-container の集計結果である．X 軸は階級であり，1 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．

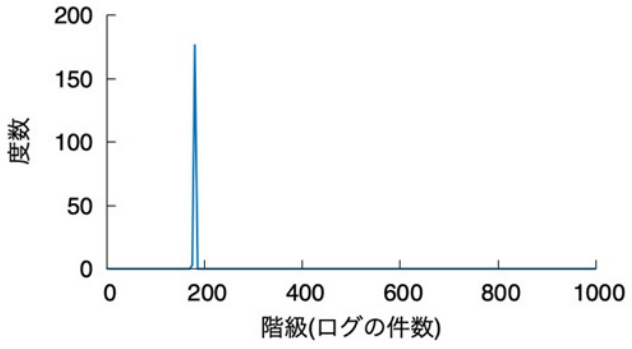


図 10 author-app-container(集計間隔：1 分間，検索期間：3 時間)

図 10 は，ログ件数の分布が一点に集中しており，グラフの山が 1 つしかない特徴があり，一定の件数が出力されていることが読み取れる．このことから，1 分間隔で集計した際の author-app-container では，グラフの山が 1 つにまとまっており，短期間で大きな変動が発生していないこ

とがわかる．

集計間隔を 1 分とした場合の filebeat のログの分布を図 11 に示す．図 11 は，検索期間が 3 時間で集計間隔が 1 分間の filebeat の集計結果である．X 軸は階級であり，1 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 11 は，ログ件数の分布が複数にばらついており，

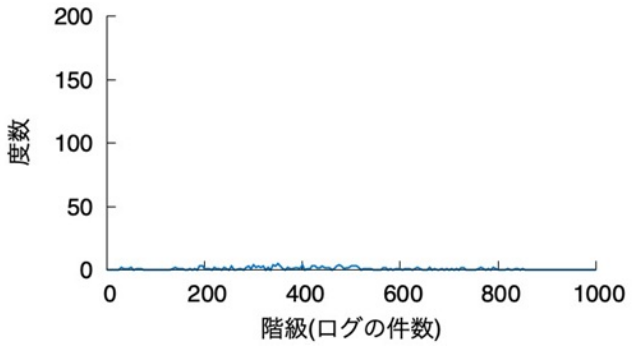


図 11 filebeat(集計間隔：1 分間，検索期間：3 時間)

複数のグラフの山が存在する特徴があり，ログの出力が分散していることが読み取れる．このことから，1 分間隔で集計した際の filebeat では，複数のグラフの山が存在し，ログの出力が一定ではなく，短期間でログの出力に変動があることがわかる．

集計間隔を 5 分とした場合の author-app-container のログの分布を図 12 に示す．図 12 は，検索期間が 3 時間で集計間隔が 5 分間の author-app-container の集計結果である．X 軸は階級であり，5 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．

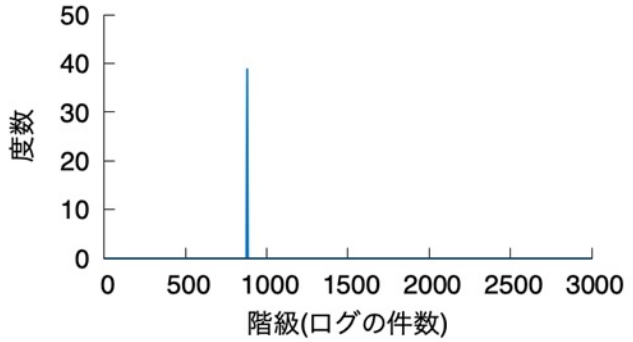


図 12 author-app-container(集計間隔：5 分間，検索期間：3 時間)

図 12 は，ログ件数の分布が一点に集中しており，グラフの山が 1 つにまとまっている特徴がある．5 分間隔にしても分布の形は変わらず，一定の件数でログが出力されていることが読み取れる．このことから，author-app-container では，時間幅が 5 分でもログ出力の傾向は安定しており，短期的な変動の影響を受けにくいことがわかる．

集計間隔を 5 分とした場合の filebeat のログの分布を図 13 に示す．図 13 は，検索期間が 3 時間で集計間隔が 5 分間の filebeat の集計結果である．X 軸は階級であり，5 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 13 は，ログ件数の分布が複数の階級に広がって

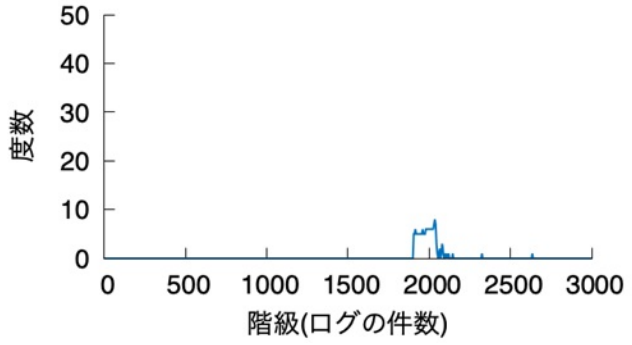


図 13 filebeat(集計間隔：5 分間，検索期間：3 時間)

おり，複数のグラフの山が確認できる特徴がある．5 分間隔で集計した場合でも，ログ件数にばらつきが残っており，一度に出力されるログ量が安定していないことが読み取れる．このことから，filebeat では，時間幅が 5 分でもログの変動が続いており，短期間では一定の傾向に収束しないことがわかる．

集計間隔を 10 分とした場合の author-app-container のログの分布を図 14 に示す．図 14 は，検索期間が 3 時間で集計間隔が 10 分間の author-app-container の集計結果である．X 軸は階級であり，10 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 14 は，ログ件

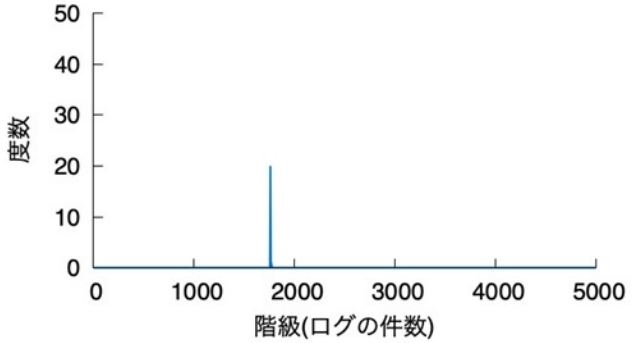


図 14 author-app-container(集計間隔：10 分間，検索期間：3 時間)

数の分布が一点に集中しており，グラフの山が 1 つにまとまっている特徴がある．10 分間隔にしても分布の形は変わらず，一定の件数でログが出力されていることが読み取れる．このことから，author-app-container では，時間幅が 10 分でもログ出力の傾向は安定していることがわかる．

集計間隔を 10 分とした場合の filebeat のログの分布を図

15 に示す．図 15 は，検索期間が 3 時間で集計間隔が 10 分間の filebeat の集計結果である．X 軸は階級であり，10 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 15 は，ログ件数が複数の階級に分散しており，グ

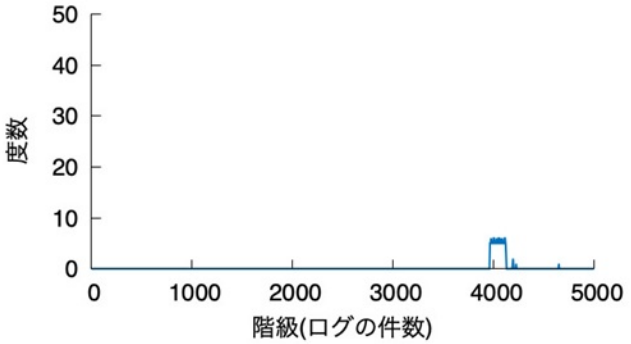


図 15　filebeat(集計間隔：10 分間，検索期間：3 時間)

ラフ上に複数の山が存在する特徴がある．10 分間隔にすることで細かい変動は平滑化されているが，ログの出力量に幅が見られる．このことから，filebeat では，10 分間隔でもログの出力は一定ではなく，変動が続いていることがわかる．

集計間隔を 15 分とした場合の author-app-container のログの分布を図 16 に示す．図 16 は，検索期間が 3 時間で集計間隔が 15 分間の author-app-container の集計結果である．X 軸は階級であり，15 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 16 は，ロ

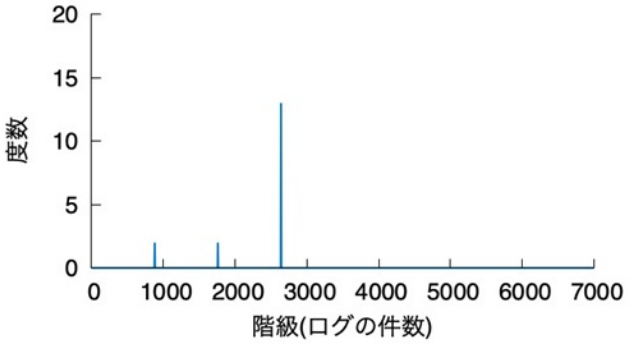


図 16　author-app-container(集計間隔：15 分間，検索期間：3 時間)

グ件数の分布が一点に集中しており，グラフの山がまとまっている．15 分間隔にしてもログ出力の傾向に大きな変化はなく，安定した挙動が読み取れる．このことから，author-app-container では，広めの時間幅である 15 分でもログの出方に大きな変動がないことが確認できる．

集計間隔を 15 分とした場合の filebeat のログの分布を図 17 に示す．図 17 は，検索期間が 3 時間で集計間隔が 15 分間の filebeat の集計結果である．X 軸は階級であり，15 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 17 は，ログ件数が複数の階級に広がっており，複

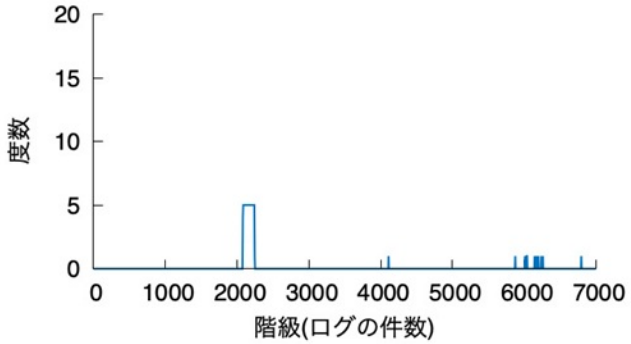


図 17　filebeat(集計間隔：15 分間，検索期間：3 時間)

数の山が存在する分布になっている．15 分間隔でもばらつきが残っており，ログの出力量に変動が続いていることが読み取れる．このことから，filebeat では，広めの時間幅である 15 分でもログ出力の安定性が低く，挙動が不規則であることがわかる．

集計間隔を 30 分とした場合の author-app-container のログの分布を図 18 に示す．図 18 は，検索期間が 3 時間で集計間隔が 30 分間の author-app-container の集計結果である．X 軸は階級であり，30 分間ごとに集計した際の一度に出力されたログ件数を表す．Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 18 は，ロ

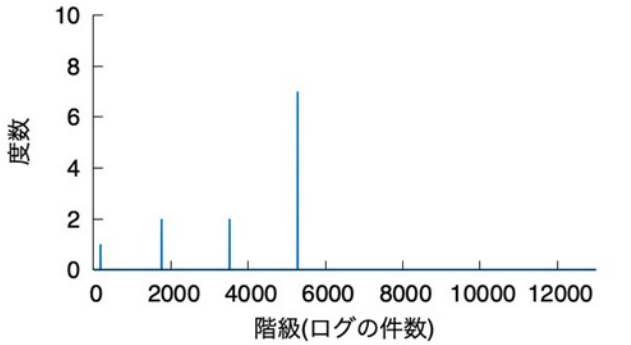


図 18　author-app-container(集計間隔：30 分間，検索期間：3 時間)

グ件数の分布が 1 つの山にまとまっており，30 分間隔でも変動がほとんど見られない．大きな時間幅である 30 分にしても分布が広がらないことから，長期にわたって安定したログ出力であることが読み取れる．このことから，author-app-container は，短期・中期・長期いずれの時間幅でも安定していることがわかる．

集計間隔を 30 分とした場合の filebeat のログの分布を図 19 に示す．図 19 は，検索期間が 3 時間で集計間隔が 30 分間の filebeat の集計結果である．X 軸は階級であり，30 分間ごとに集計した際の一度に出力されたログ件数を表す．

Y 軸は度数であり，X 軸で定義した階級が何件あったかを表す．図 19 は，ログ件数の分布が複数の階級に広がって

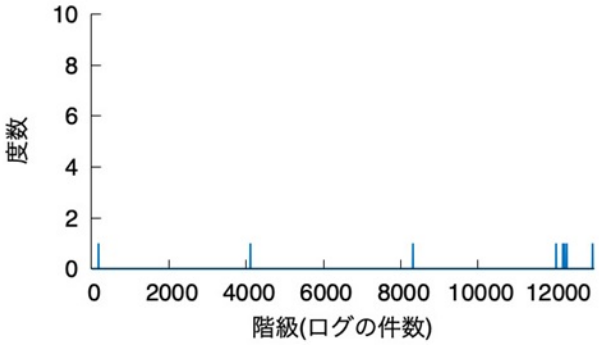


図 19 filebeat(集計間隔：30 分間，検索期間：3 時間)

おり，30 分間隔にしても山が複数存在する特徴がある．時間幅を長くして平滑化してもばらつきが残り，ログ出力が安定しないことが読み取れる．このことから，filebeat では，大きな時間幅である 30 分の集計でも出力にばらつきがあり，コンテナの動作が時間的に不規則であることがわかる．

一定量のログが継続的に出力されているパターンでは，集計期間を変更しても階級が同じ値に集中する結果となった．author-app-container の他に，折れ線グラフで集計した際に，一定量のログが継続的に出力されていたコンテナ名は 18 種類あり builder，cadvisor，ceph-exporter，coredns，csi-snapshotter，front-app-container，fulltext-app-container，istio-proxy，jaeger，metrics，mgr，paper-app-container，paper-mongo-container，rook-ceph-operator，stats-app-container，stats-mongo-container，stns-server，watch-active であった．

ログの出力に変動があるパターンでは，集計期間を変更しても階級にばらつきがある結果となった．filebeat の他に，折れ線グラフで集計した際に，ログの出力に変動があったコンテナ名は 5 種類あり author-mongo-container，expand-bluefs，kea，mon，osd であった．

ログ件数を集計した結果，1 分あたりに出力されるログ件数はばらつきがあるため，特徴や傾向が表れなかった．また，ログ件数の分布は正規分布や二項分布になっておらず，中央値や平均値を代表値として使用できない．そのため，本稿では正規分布に従う必要のないノンパラメトリック手法の基本的な手法である 99 パーセンタイルを採用した．ログの出力は正規分布しないことが確認でき，ノンパラメトリック手法の中でも基本的な手法である 99 パーセンタイルを採用する．99 パーセンタイルは上位 1%の値を異常とするため，ログ件数の変動が小さい場合や安定してログが出力された場合でも，上位 1%に位置する値は異常と検出される．また，コンテナの停止のようにログ件数が増加しない障害の場合には，99 パーセンタイルを使用して障害を検出することができない．

6. 議論

本稿の提案ソフトウェアでは，各コンテナのログ件数に対し 99 パーセンタイルで閾値を設定し，直前の時間帯のログ件数が閾値を超えた場合に異常コンテナを検知する．しかし，この手法ではデータの上位 1%が必ず閾値を超えるため，システムが正常であっても一部のコンテナが異常と判定され，短期間の一時的なログ件数の増加による誤検知が生じる可能性がある．これは，ログ件数だけでなく，ログメッセージの長さや内容の類似度を指標とすることで改善できる．ログ内容をもとにクラスタリングを行い，各クラスタごとにログ件数を集計して比較することで，特定のログ件数が急増した場合のみ異常として判断できる．例えば k-means 法を適用し，ログのタイムスタンプやコンテナ名，ログメッセージをクラスタリングすることで，通常時の安定したパターンと異常時の急増したパターンを区別でき，短期的な変動による誤検知を抑制できる．

本稿の提案ソフトウェアでは，99 パーセンタイルの閾値を直前の時間帯のログ件数が超えた場合，異常が発生したコンテナを検知する．しかし，Kubernetes クラスタの障害や Pod 内のコンテナが終了状態になった場合は，ログが出力されないため，本稿の提案ソフトウェアでは，異常が発生したコンテナを特定することができない．これは，kubectl describe コマンドの events の件数を数えることで改善できる．

7. おわりに

課題は，障害の原因調査でログを確認する際，専門知識やスキルが不足していると原因の特定や復旧に時間を要することである．提案手法では，タイムスタンプとコンテナ名をもとにログ件数をある時間間隔ごとにコンテナ名別に集計を行った．集計結果をコンテナ名ごとに昇順に並び替え，99 パーセンタイルの位置にあるログ件数を閾値とした．閾値を超える値を異常値として検出し，障害が発生しているコンテナ名を特定した．評価実験では，ユーザ ID を管理するサーバである STNS の builder コンテナで起きた障害をシナリオとして，検出の可否と提案ソフトウェアありの調査に要する時間の測定の 2 つを評価指標とした．検出の可否の測定は，1 週間分の検索期間と集計間隔を変更し，提案ソフトウェアで障害が発生したコンテナを特定できたか測定した．提案ソフトウェアありの調査に要する時間の測定は，被験者が提案ソフトウェアを使用して障害の発生箇所を特定するまでの時間を計測し，人が行う必要がある作業と自動化が可能な作業をそれぞれ測定した．検出の可否の測定は，2025 年 11 月 3 日の 14 時から 2025 年 11 月 3 日の 17 時 33 分に行った．検出の可否の測定では，集計間隔が 1 分と 5 分と 15 分の場合，各検索期間で検出

でき，10 分と 30 分の場合，各検索期間で検出できなかった．提案ソフトウェアありの調査に要する時間の測定は，2025 年 11 月 5 日の 16 時 25 分から 2025 年 11 月 5 日の 16 時 59 分に行い，障害原因の全体の調査時間は 1892 秒であった．人が行う必要がある作業に要した時間は 774 秒であり，全体の作業時間の約 41%を占めていた．一方，自動化が可能な作業に要した時間は 1118 秒であり，全体の作業時間の約 59%を占めていた．この結果から，全体の作業時間の半分以上を自動化が可能な作業で占めていることがわかり，管理者が行う作業の一部を削減したことを確認できた．また，障害の原因調査における負担の軽減と調査全体の時間短縮により障害の原因調査の効率化に寄与した．

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に御礼申し上げます．また，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手塚雄星さん，山崎拓海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] 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).

[3] 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).

[4] 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).

[5] Koyama, T. and Kushida, T.: Log message with JSON item count for root cause analysis in microservices, *2023 6th Conference on Cloud and Internet of Things (CIoT)*, pp. 55–61 (online), DOI: 10.1109/CIoT57267.2023.10084901 (2023).

[6] 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).

[7] Gupta, S. and Rani, R.: A comparative study of elasticsearch and CouchDB document oriented databases, *2016 International Conference on Inventive Computation Technologies (ICICT)*, Vol. 1, pp. 1–4 (online), DOI: 10.1109/INVENTIVE.2016.7823252 (2016).

[8] Doan, D. N. and Iuhasz, G.: Tuning Logstash Garbage Collection for High Throughput in a Monitoring Platform, *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 359–365 (online), DOI: 10.1109/SYNASC.2016.063 (2016).

[9] 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).

[10] 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).

[11] Zhou, X., Peng, X., Xie, T., Sun, J., Ji, C., Li, W. and Ding, D.: Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study, *IEEE Transactions on Software Engineering*, Vol. 47, No. 2, pp. 243–260 (online), DOI: 10.1109/TSE.2018.2887384 (2021).

[12] Wang, L., Zhao, N., Chen, J., Li, P., Zhang, W. and Sui, K.: Root-Cause Metric Location for Microservice Systems via Log Anomaly Detection, *2020 IEEE International Conference on Web Services (ICWS)*, pp. 142–150 (online), DOI: 10.1109/ICWS49710.2020.00026 (2020).

[13] 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).

[14] 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).

[15] Kapel, E., Cruz, L., Spinellis, D. and Van Deursen, A.: On the Difficulty of Identifying Incident-Inducing Changes, *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '24, New York, NY, USA, Association for Computing Machinery, p. 36–46 (online), DOI: 10.1145/3639477.3639755 (2024).

付録

表 7 (実験 2) 集計期間が 5 分の各検索期間の検出の可否の結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	276	超える	異常	発生した	TP
2 日	276	超える	異常	発生した	TP
3 日	276	超える	異常	発生した	TP
4 日	276	超える	異常	発生した	TP
5 日	276	超える	異常	発生した	TP
6 日	276	超える	異常	発生した	TP
7 日	276	超える	異常	発生した	TP

表 8 (実験 2) 集計期間が 15 分の各検索期間の検出の可否の結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	822	超える	異常	発生した	TP
2 日	822	超える	異常	発生した	TP
3 日	822	超える	異常	発生した	TP
4 日	822	超える	異常	発生した	TP
5 日	822	超える	異常	発生した	TP
6 日	822	超える	異常	発生した	TP
7 日	822	超える	異常	発生した	TP

表 9 (実験 2) 集計期間が 30 分の各検索期間の検出の可否の結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	1637	超えない	正常	発生した	FN
2 日	1639	超えない	正常	発生した	FN
3 日	1644	超えない	正常	発生した	FN
4 日	1645	超えない	正常	発生した	FN
5 日	1644	超えない	正常	発生した	FN
6 日	1644	超えない	正常	発生した	FN
7 日	1644	超えない	正常	発生した	FN

表 10 (実験 3) 集計期間が 1 分での誤検知の測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	56	超えない	正常	発生していない	TN
2 日	56	超えない	正常	発生していない	TN
3 日	56	超えない	正常	発生していない	TN
4 日	56	超えない	正常	発生していない	TN
5 日	56	超えない	正常	発生していない	TN
6 日	56	超えない	正常	発生していない	TN
7 日	56	超えない	正常	発生していない	TN

表 11 (実験 3) 集計期間が 5 分での誤検知の測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	276	超えない	正常	発生していない	TN
2 日	276	超えない	正常	発生していない	TN
3 日	276	超えない	正常	発生していない	TN
4 日	276	超えない	正常	発生していない	TN
5 日	276	超えない	正常	発生していない	TN
6 日	276	超えない	正常	発生していない	TN
7 日	276	超えない	正常	発生していない	TN

表 12 (実験 3) 集計期間が 10 分での誤検知の測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	549	超えない	正常	発生していない	TN
2 日	550	超えない	正常	発生していない	TN
3 日	550	超えない	正常	発生していない	TN
4 日	550	超えない	正常	発生していない	TN
5 日	550	超えない	正常	発生していない	TN
6 日	550	超えない	正常	発生していない	TN
7 日	550	超えない	正常	発生していない	TN

表 13 (実験 3) 集計期間が 15 分での誤検知の測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	822	超えない	正常	発生していない	TN
2 日	824	超えない	正常	発生していない	TN
3 日	824	超えない	正常	発生していない	TN
4 日	824	超えない	正常	発生していない	TN
5 日	824	超えない	正常	発生していない	TN
6 日	824	超えない	正常	発生していない	TN
7 日	824	超えない	正常	発生していない	TN

表 14 (実験 3) 集計期間が 30 分での誤検知の測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	1639	超えない	正常	発生していない	TN
2 日	1644	超えない	正常	発生していない	TN
3 日	1645	超えない	正常	発生していない	TN
4 日	1645	超えない	正常	発生していない	TN
5 日	1644	超えない	正常	発生していない	TN
6 日	1644	超えない	正常	発生していない	TN
7 日	1644	超えない	正常	発生していない	TN

表 15 (実験 4) 集計期間が 1 分での測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	56	超える	異常	発生していない	FP
2 日	56	超える	異常	発生していない	FP
3 日	56	超える	異常	発生していない	FP
4 日	56	超える	異常	発生していない	FP
5 日	56	超える	異常	発生していない	FP
6 日	56	超える	異常	発生していない	FP
7 日	56	超える	異常	発生していない	FP

表 16 (実験 4) 集計期間が 4 分での測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	276	超えない	正常	発生していない	TN
2 日	276	超えない	正常	発生していない	TN
3 日	276	超えない	正常	発生していない	TN
4 日	276	超えない	正常	発生していない	TN
5 日	276	超えない	正常	発生していない	TN
6 日	276	超えない	正常	発生していない	TN
7 日	276	超えない	正常	発生していない	TN

表 17 (実験 4) 集計期間が 9 分での測定結果

検索期間	閾値 [件]	直前のログ件数が閾値を超えるか	提案方式での異常判定	実際の障害	判定結果
1 日	550	超えない	正常	発生していない	TN
2 日	550	超えない	正常	発生していない	TN
3 日	550	超えない	正常	発生していない	TN
4 日	550	超えない	正常	発生していない	TN
5 日	549	超えない	正常	発生していない	TN
6 日	549	超えない	正常	発生していない	TN
7 日	549	超えない	正常	発生していない	TN