ログファイル内でのステータスコード500番台の有無にもとづく非圧縮によるシステム障害の検索時間の短縮

金子 拓磨[1]　大野 有樹[2]　串田 高幸[1]

概要：Web サイトでは，アクセス情報やシステムの実行情報がログという形式で記録されており，ログはシステム稼働状況の確認や障害発生時の対処に用いられるデータである．そのため出来る限り残しておく必要があり，圧縮して保存することにより容量の削減が出来る．しかし圧縮して保存すると，圧縮せずに保存しているログに比べて検索時間が長くなる．課題はログを圧縮または非圧縮で保存する際の基準をどのように決定するかである．システム障害の発生原因に関するログのみを非圧縮で保存することにより，検索時間の短縮と容量の確保が可能である．本稿では，ログのテキストに記述されているステータスコードの 500 番台を基準として 500 番台が含まれている場合は非圧縮ファイルとして保存し，ステータスコードの 500 番台が含まれていない場合は圧縮して別のストレージに保存する．それによりシステム管理者がログを確認するとき対象のファイルのみが非圧縮ファイルとして保存されているためすべてのログを検索するより検索時間が短縮する．結果，1998 年のサッカーワールドカップの予約 Web サイトのログを対象とし 100 回の平均を計測したとき，提案前の検索時間の平均が約 966.26 秒かかり，提案後の検索時間の平均が約 0.59 秒かかることとなり，提案を用いたときシステム管理者が検索するときの時間を 965.67 秒短縮することが出来た．

1. はじめに

背景

Web サイトやソフトウェアでは通常，サイトへのアクセス情報やシステムに関する実行情報がログという一定の形式で記録されている [1]．特に大規模な企業であれば，毎日大量のログが生成される [2]．ログはシステムが正常に稼働しているかを確認することのできる現状唯一のデータで，サービスプロバイダやユーザによるサービスの管理に用いられており，その用途にはバグの修正や異常検出，テスト結果の分析，システム監視といったソフトウェア開発のプロセスで使用されることがある [3]. ログの幅広い用途，有用性により開発者はシステムのソースコードの中に多くのログを記述するように設定している [4]．上記のことを常に確認できるようにするため，ログは削除をせず保存しておくことは重要である．

ログサーバとはネットワーク機器やサーバ機器が出力したログを収集，集約するログ専用のサーバで，大規模な企業ほどアプリケーションやソフトウェアのサーバと分けるために使用される[*1] [5, 6].

ログの種類にはアクセスログがある．アクセスログとは，いつ，どこから，どの主体がアクセスしたのかを記録するログであり，マーケティングの統計を取ることやシステム障害時の確認作業に用いることがある．ファイルが大量に保存されることによってストレージが圧迫される状況で対処する方法の中にファイルの圧縮がある[*2]．圧縮はデータで表している情報を実質的に内容を変更せず，より短いデータに置き換える処理であり，拡張子に「.gz」や「.zip」というものが存在する[*3].

課題

本稿ではログの保存先のストレージの容量を削減しつつログ検索時間も削減することを目的としている．ログは圧縮して保存するとデータ容量が少なくなるため保存する際にストレージ容量の削減が可能になる．しかし圧縮ファイルのままログを検索に掛けると非圧縮ファイルでのログ検索と比較して検索に時間がかかる．以前基礎実験で，圧縮ファイルと非圧縮ファイルの検索時間を比較する実験をした [7]．結果として，圧縮ファイルが最小で約 42.40 秒，最

[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

[*1] https://www.amiya.co.jp/column/10496/
[*2] https://x.gd/U1GxN
[*3] https://x.gd/jL7vj

大で約 44.96 秒，非圧縮ファイルが最小で約 22.80 秒，最大で約 23.86 秒であった．平均としては，圧縮ファイルが約 42.96 秒，非圧縮ファイルが約 23.20 秒であった．このことから非圧縮ファイルが圧縮ファイルよりも検索時間が短縮されることが示された．圧縮ファイルだけでは検索に時間がかかってしまうため，検索の対象を絞りその範囲のみ非圧縮で保存することにより検索全体の時間を短縮することとする．前述から，図 1 のようにどのファイルを対象にするか判断が出来ないため，本稿では，検索の対象を選択することが困難であることが課題である．

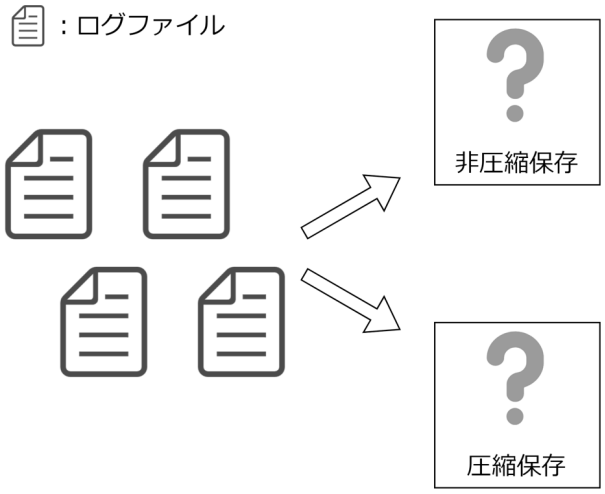

図 1 圧縮・非圧縮の判定基準

各章の概要

第 2 章では関連研究について述べる．第 3 章では本稿の提案方式の説明とユースケースの説明する．第 4 章では今回の提案に対する実験の方法について述べる．第 5 章では評価方法と分析手法について述べる．第 6 章では本稿の議論について述べる．第 7 章では本稿のまとめを述べる．

2. 関連研究

ログを圧縮または解凍することによってクエリ時間の短縮を図っている研究がある [8]．ログを溜めておくために保存し続けるとそれだけで大量の領域が占有されてしまうことを課題として挙げている．著者はこの課題に対して，ログを複数の列に分割しその列単位で異なるモデルごとに圧縮する方法を提案をしている．本稿とは提案のログを圧縮する基準や方法の違いがある．

ログを前処理するアルゴリズムの提案によってログの圧縮率と圧縮時間の向上を図っている研究がある [9]．ログは繰り返しアーカイブに入力されるため大量発生することを課題としている．著者はこの課題に対して，前処理と一般的な圧縮を用いて可逆圧縮とし，圧縮時間の向上を図る提案をしている．ログの検索に関する項目は考慮していない．

トラブルシューティングや問題の診断のため，分散システムによって生成されたシステムログが使用される．しかし，分散システムの規模と複雑さが増大しているため，手動でシステムログを検査して異常を検出することは現実的ではない．著者は異常検知の為の非構造ログ分析手法を提案している [10]．本稿とは課題がログの検索という点は酷似しているが，検索の時間ではなく異常の検出に焦点を当てている点が異なっている．

ログメッセージの保存や検索対象のサイズ増加を抑制するため，分散システムによる検索手法を提案している研究がある [11]．ネットワーク管理者はサーバ，ネットワーク，セキュリティ，アプライアンスによって生成されたログを収集して保存している．これによりネットワークのトラブルやセキュリティインシデントが発生したときにログの内容を調査して問題の原因特定を行う．しかし，管理対象が大きくなるほどそのログメッセージの保存や検索にかかるコストが大きくなることが問題となる．そこで，「はやぶさ」という各サーバに複数のワーカープロセスのある，単純な分散システムをシステムを提案している．この研究では分散システムによって検索の時間の短縮を行っているが，もう一つの問題である大量にログが存在している場合のストレージの容量の確保に改善点がある．

3. 提案

本稿では以下に記述しているユースケースに従って，ログが 1 日毎のファイルに分割されていることとする．

提案方式

本稿の提案方式では，ログのテキストに記載されているステータスコードから圧縮するログの対象を決定する．今回のケースでは Web サーバで発生したログをファイルとして保存する際，1 日毎に保存し，その中でも正常ログと異常ログに分けて保存する．正常，異常の判断はステータスコードを基に行い，正常ログはステータスコードが 200 番台，300 番台のログで，異常ログは 400 番台，500 番台のログになる．図 3 に圧縮と判断する基準を示す．ステータスコードの 500 番台を基準にログの圧縮を判断し，500 番台が含まれているファイルを非圧縮で保存する．500 番台とした理由としては，ユースケースに用いた事例において発生する障害がサーバ障害であるため，サーバ側のエラーを示すステータスコード 500 番台とした．

図 2 はシーケンス図であり，本稿の提案方式のログの流れを説明している．ライフラインとして Web サーバ，検索ディレクトリ，提案ソフトウェア，圧縮ログディレクトリが存在し，検索ディレクトリ，提案ソフトウェア，圧縮ログディレクトリはログサーバに置かれている．以下にそれぞれのライフラインで何を実行するかを説明する．最初に

Web サーバにてアクセスログが発生する．発生したログをログサーバ内にある検索ディレクトリに送信する．実行が開始されるのは日を跨いだときである．そのため直近一日のログは全て非圧縮のまま保存されている．提案ソフトウェアが検索ディレクトリに対してログの確認をし，取得した中にステータスコードが 500 番台のログの有無を確認する．500 番台のログがある場合はエラーログである 400 番台と 500 番台を取り出し，非圧縮でログを検索ディレクトリに保存したままにし，元々のログは提案ソフトが圧縮して圧縮ログディレクトリに保存する．500 番台のログがないログファイルである場合は提案ソフトがそのログファイルを圧縮して，同じくログサーバ内にある圧縮ログディレクトリに保存する．

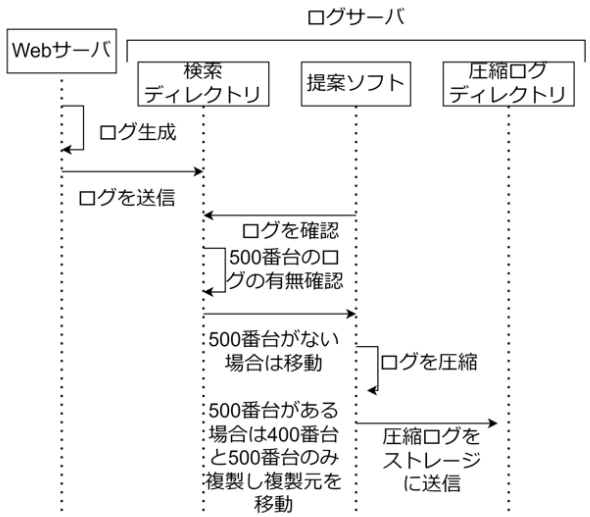

図 2　提案方式のシーケンス図

図 3 は非圧縮で保存するログファイルの選定の基準を示した図である．各ファイルごとに保存されたログの中身を確認して圧縮か非圧縮かを判断をする．その基準として，前述したようにステータスコードの 500 番台かを判断の対象とすることとした．日毎に分割し保存されているファイルのログテキストにあるステータスコードを参照し，サーバエラーである 500 番台が含まれていた場合は検索対象のログとし，エラーログである，400 番台と 500 番台を圧縮せず非圧縮のまま保存し続け管理者が確認するときの時間を短縮させる．図 3 の場合は HTTP の 404，503，504，505 が対象になる．500 番台のログが含まれていなかった場合は検索対象外のログとし，ファイルを圧縮してから圧縮ログディレクトリに転送して保存する．

ユースケース・シナリオ

本稿では，1998 年に開催されたサッカーワールドカップ

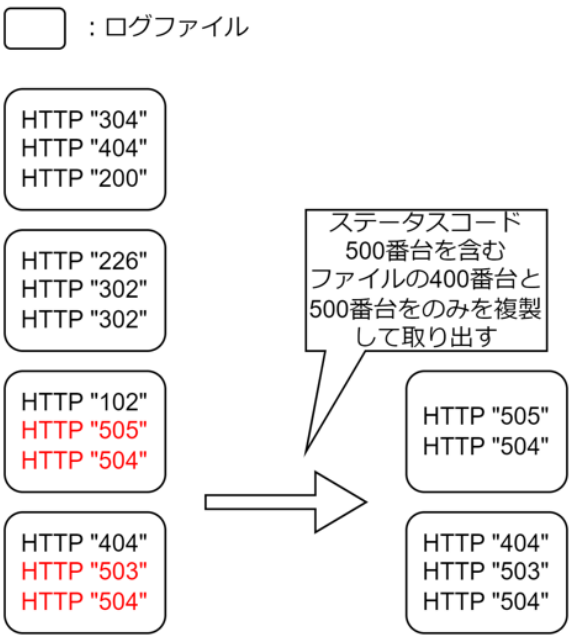

図 3　非圧縮で保存する対象の選定基準

のチケット予約の Web サイトを想定している[*4]．対象とした Web サイトでは約 3 か月にわたってログの収集が行われ，期間中に約 13 億 5000 万件のリクエストを受け取った [12]．対象の Web サイトは 1 日ごとにファイルを分割して保存している．

図 4 は本稿のユースケースシナリオを示している．システム管理者は Web サーバとログサーバを管理している．ログファイルは Web サーバからログサーバの検索ディレクトリに送られ，提案ソフトウェアによって分割される．システム管理者が必要に応じて検索をかけるときは，非圧縮でログが保存されている検索ディレクトリに向けて行う．

ソースコード 1 はユースケースに用いた予約サイトで実際に発生したログを一部抜粋したものとなる．提案で提示したステータスコードはバージョンを示す「HTTP1.0」の後ろに記述されている「200」「304」「500」の 3 桁の数字のことである．

4.　実装

本稿の実装では提案方式を用いてログを分割し，分割したファイルごとにストレージに保存する．本稿ではユースケースにしている 1998 年サッカーワールドカップチケット予約サイトで過去に実際に発生したログを使用する．図 5 は実装における構成要素とその動きの図である．構成要素として，Web サーバとログサーバ内の検索ディレクトリ，提案ソフトウェア，圧縮ログディレクトリがある．以下に実装の図の説明を示す．

[*4] https://www.fifa.com/tournaments/mens/worldcup/1998france

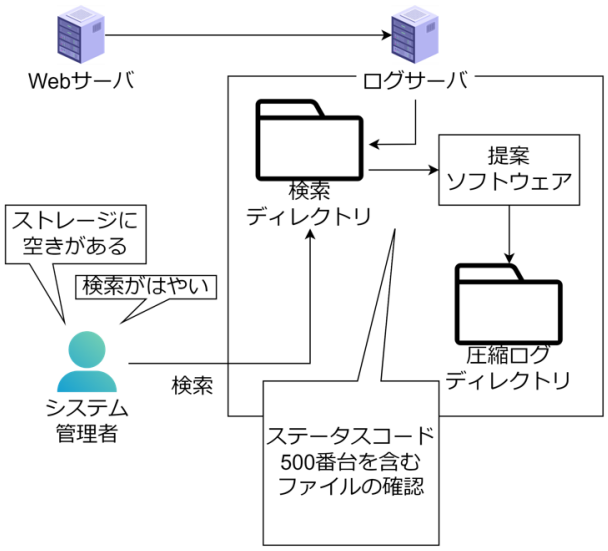

図 4　ユースケースシナリオ

①提案ソフトウェアが検索ディレクトリ内にあるログをファイルごとに確認し，サーバエラーである 500 番台のログが含まれている場合，対象のログファイルからエラーログである 400 番台と 500 番台のログを取り出し，取り出したログを検索ディレクトリ内に保存し続ける．

②提案ソフトウェアが検索ディレクトリから対象のログファイルを移動し，検索ディレクトリに残ったログファイルを削除する．取得したログファイルは圧縮する．

③圧縮したログファイルは今まで保存していたところとは違う圧縮ログディレクトリに転送して保存する．

ソースコード 1　WorldCup 予約サイトのログの抜粋

```
198.7.1.0 - - [04/May/1998:23:02:31 +0000] "GET␣/
    english/help/site.html␣HTTP/1.0" 200 7697
14.8.1.0 - - [04/May/1998:23:02:31 +0000] "GET␣/
    images/case5.gif␣HTTP/1.0" 200 1362
14.8.1.0 - - [04/May/1998:23:02:31 +0000] "GET␣/
    images/bord_stories01.gif␣HTTP/1.0" 304 -
203.85.0.0 - - [04/May/1998:23:02:31 +0000] "GET␣
    /spanish/␣HTTP/1.0" 500 305
15.230.1.0 - - [04/May/1998:23:02:31 +0000] "GET␣
    /french/images/teams_bu_group_on.gif␣HTTP
    /1.0" 200 838
15.230.1.0 - - [04/May/1998:23:02:32 +0000] "GET␣
    /french/images/team_hm_header_shad.gif␣
    HTTP/1.0" 200 1589
```

5.　実験

本稿では対象の Web サイトから取得したアクセスログを用いて，ステータスコード 500 番台を対象に実験を行う．本実験の目的は，提案を用いてログファイルが分割され，ログの検索の時間が短縮されるか確認するためである．検索する範囲は 3 か月のログの集計期間全体である．理由としてはアクセス障害が起きたときの原因の範囲が不明なためである．

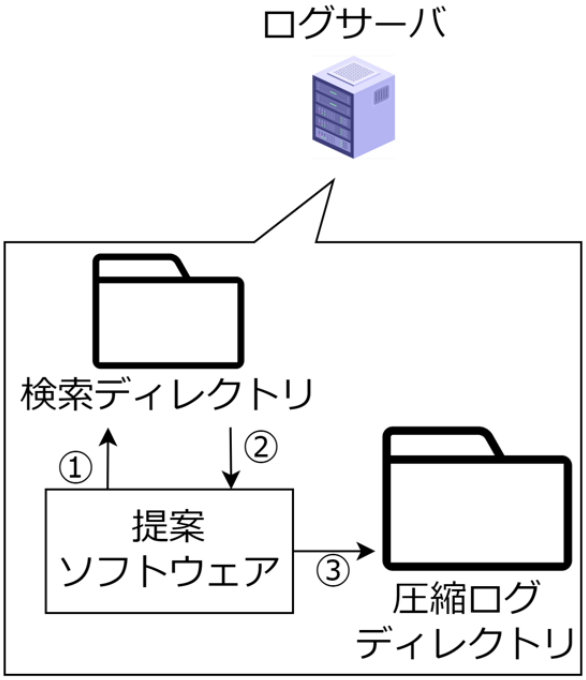

図 5　実装するソフトウェアの図

実験環境

実験環境は ESXi の VM を用いる．使用する VM は OS が Ubuntu22.04 であり，vCPU が 4 コア，RAM メモリが 8GB，HDD が 512GB に設定する．

以下に実験に用いる VM の構成要素を示す．

VM の構成要素

- OS：Ubuntu 22.04 LTS
- vCPU：4 コア
- RAM：8GB
- HDD：512GB

今回実験するに当たって使用するログファイルの保存の際の名称がソースコード 2 のように記述されていることとする．

ソースコード 2　ログファイルの保存形式

```
wc_day10_1_err.log
wc_day11_1_err.log
wc_day11_2_err.log
wc_day12_1_err.log
```

今回ファイルの検索にはソースコード 3 のように grep コマンドを用いて行い，検索の時間を測るために/usr/bin/-time を使用する．

今回実験で使用した検索の条件は，ユースケースよりア

クセス過多によってサーバの障害が起きてしまったとき，システム管理者が Web サイトのアクセス状況の確認するためにアクセスログのサーバエラーに関する HTTP のステータスコード 500 番台とする．

ソースコード 3　検索で使用したコマンド

```
$ /usr/bin/time grep "" [5][0-9][0-9]"␣" wc_day
    *.log
```

実験結果と分析

提案を用いる前のログファイルと提案方式を用いた後のログファイルの検索時間の比較を行い，検索時間を評価とする．

検索対象のログはユースケースで記述した 1998 年ワールドカップのチケット予約サイトのアクセスログを使用する．検索はアクセス障害の原因を確認するために行い，ログの件数は全数で 13 億 5385 万 7139 件であり，提案後の件数は 869 万 95 件である．

図 6 に提案前と提案後の平均の検索時間を示している．図 6 は結果の差が大きかったため対数グラフで表すこととする．

検索は各 100 回ずつ行い，提案前の検索時間は平均で約 966.26 秒かかっており，提案後の検索時間は平均で約 0.59 秒かかっている．結果として検索時間の平均の時間差が約 965.67 秒となった．検索の結果 500 番台は 3 万 6395 件であることが分かった．

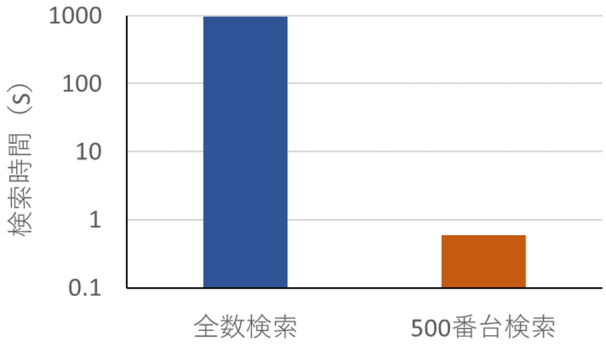

図 6　全数検索と 500 番台検索の平均検索時間の比較

6. 議論

本稿の議論は提案方式によって非圧縮で保存されているログが検索されなかった場合，そのログをいつまで非圧縮で保存し続けるのかという問題である．今回のユースケースを基にした実験では，過去に保存されたログを使用したため検索されることが決まっているが，実際にリアルタイムでログの集計をした場合，この問題が発生する．解決方法として，今回のユースケースを基準に置いた場合は予約のワールドカップの試合が終了した場合に実行することが挙げられる．試合前に行ってしまうとトラブルが発生したときの対処が遅くなる恐れがあるためである．

7. おわりに

ログは Web サイトやアプリケーションシステムにおいて障害が起きた時に使用される．そのためログを出来るだけ保存する必要があり，ストレージに長くログを溜めるためにログを圧縮することでその容量を削減することが出来る．しかし，圧縮状態のログでは非圧縮状態のログより検索に時間がかかってしまう．そこでログの保存先のストレージの容量を確保しつつログの検索時間も削減することを目的とし，どの範囲までを圧縮にするかを課題として置いた．本稿では，ログに記述されているステータスコード 500 番台のログを対象に圧縮と非圧縮の判断をすることとした．実装において Web サーバで出力されたログをシステム管理者が検索する前に提案方式を用いてログを分割しておく．実験では提案前の事前分割をしなかった場合の検索と提案後の検索を比較した．その結果，提案前の検索時間が平均で約 966.26 秒で，提案後の検索時間が平均で約 0.59 秒となり，平均の時間差で約 965.67 秒の削減が出来た．

謝辞　本テクニカルレポートを執筆にあたりご指導いただきました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河竹純一さん，東京工科大学コンピュータサイエンス学部先進情報専攻の三上智徳さん，西嶌知良さん，増田和範さん，平尾真斗さんに御礼申し上げます．

参考文献

[1] He, P., Zhu, J., Zheng, Z. and Lyu, M. R.: Drain: An online log parsing approach with fixed depth tree, *2017 IEEE international conference on web services (ICWS)*, IEEE, pp. 33–40 (2017).

[2] Yao, K., Sayagh, M., Shang, W. and Hassan, A. E.: Improving state-of-the-art compression techniques for log management tools, *IEEE Transactions on Software Engineering*, Vol. 48, No. 8, pp. 2748–2760 (2021).

[3] Hassani, M., Shang, W., Shihab, E. and Tsantalis, N.: Studying and detecting log-related issues, *Empirical Software Engineering*, Vol. 23, pp. 3248–3280 (2018).

[4] Zhang, L., Xie, X., Xie, K., Wang, Z., Lu, Y. and Zhang, Y.: An efficient log parsing algorithm based on heuristic rules, *Advanced Parallel Processing Technologies: 13th International Symposium, APPT 2019, Tianjin, China, August 15–16, 2019, Proceedings 13*, Springer, pp. 123–134 (2019).

[5] Anusooya, R., Rajan, J. and SatyaMurty, S.: Importance of centralized log server and log analyzer software for an organization, *International Research Journal of Engineering and Technology (IRJET)*, Vol. 2, No. 3, pp. 2244–2249 (2015).

[6] Dou, H., Chen, P. and Zheng, Z.: Hdconfigor: automatically tuning high dimensional configuration parameters for log search engines, *IEEE Access*, Vol. 8, pp. 80638–80653 (2020).
[7] 金子拓磨，高橋風太，大野有樹，串田高幸：サーバダウン時におけるログファイルの非圧縮による検索時間の短縮，技術報告 CDSL-TR-135，Tokyo University of Technology CDSL Technical Report, (online: https://ja.takcslab.org/tech-report) (Jan.18, 2023).
[8] Lin, H., Zhou, J., Yao, B., Guo, M. and Li, J.: Cowic: A column-wise independent compression for log stream analysis, *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE, pp. 21–30 (2015).
[9] Balakrishnan, R. and Sahoo, R. K.: Lossless compression for large scale cluster logs, *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, IEEE, pp. 7–pp (2006).
[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*, IEEE, pp. 149–158 (2009).
[11] Abe, H., Shima, K., Miyamoto, D., Sekiya, Y., Ishihara, T., Okada, K., Nakamura, R. and Matsuura, S.: Distributed hayabusa: Scalable syslog search engine optimized for time-dimensional search, *Proceedings of the 15th Asian Internet Engineering Conference*, pp. 9–16 (2019).
[12] Arlitt, M. and Jin, T.: A workload characterization study of the 1998 world cup web site, *IEEE network*, Vol. 14, No. 3, pp. 30–37 (2000).

