# 事前ソートによるWebアクセスログの検索時間の削減

大野有樹[1]　小山智之[2]　串田 高幸[1]

**概要**：Web サービスにおいてシステム障害が起きたとき，システム管理者はアクセスログを検索することで障害の原因の絞り込みや発生時刻を特定している．障害の原因特定を早くする方法はログの検索の応答時間を削減させることである．システム管理者が障害を解消するためのログ検索における課題は複数のアプリケーションノードからログを集める場合，検索結果のソートで検索に遅延が生じることである．本提案は，あらかじめログをアクセス時刻を基準にソートして，検索対象を絞り込むためにステータスコードごとに分割して保存することでログ検索の応答時間を削減する．実験はログをランダムに生成して検索時にソートしたときと保存時にソートしたときの検索の応答時間を比較した．検索の対象のログは約 600,000 件のログから約 8,000 件のログを検索結果として出力する．約 3,000,000 件のログを検索した時，検索時にソートは 1,400[ms] で保存時にソートは 10[ms] であった．

## 1. はじめに

### 背景

ログはシステム実行中の動作の詳細を記録した唯一のデータである [1]．本研究は Web サービスにおけるアクセスログを対象とする．Web サービスは HTTP のアクセスログを出すアプリケーションを Web サービスと置く．本研究は 1 分間に 1 万のリクエストが来る Web サイトを想定する．これは 1998 年のサッカーワールドカップの Web サイトを平均すると 1 分間に 1 万リクエストがあった事例を基準にしている [2]．ログは 1 リクエストにつき 1 件生成されるものとする．本研究で扱うシステム障害は Web サービスの利用者が Web サービスにアクセスできない状態，HTTP レスポンスステータスコード 503 エラーが発生する状態と定義する．

システム管理者はシステム障害を解消するためにログを検索する．システム管理者はログを検索することでシステム障害の原因の絞り込みや発生時刻を特定している．ログ検索の応答時間はシステム管理者がログサーバーに検索クエリを発行してから、検索結果がシステム管理者へ返ってくるまでの時間とする．検索の応答時間はシステム障害が解消するまでの時間に含まれる．したがって，検索の応答時間の増加はシステム障害が解消するまでの時間の増加である．

[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

システム障害解消までの時間が増加することはシステムを管理している組織の経済的損失をもたらす [3]．例えば SLA の違約金の発生における損失である．SLA の保証する水準にはサービス稼働時間があげられる．サービス稼働時間を 99.5%[*1]とすると，1 ヶ月で 3.6 時間以内にシステム障害を収める必要がある．

障害対応の内，ログ検索の時間を用いる時間がどれだけ含まれているかの例を示す．今回、例としてあげる状況は「プログラム不備によりデータ不整合が発生しオンラインサービスでシステムエラーが発生した．暫定対応としてデータパッチを実施して解消した.」場合である[*2]．障害発生から恒久対応を除く暫定対応完了までの間を以下の 4 ステップに分ける．4 ステップは事象確認，調査，暫定対応，暫定対応内容確認とする．事象確認は障害の検知、サービスへの影響が有るか無いかの確認である．調査は障害の影響調査、対象サーバのログ取得・調査である．暫定対応は調査を基にした暫定対応内容検討、暫定対応実施である．暫定対応内容確認は暫定対応後の確認である．そのうち，ログ検索が含まれる対応のステップは調査と暫定対応内容確認である．システム障害の対応時間の全体である 94 分のうち，調査が 8 分で暫定対応の内容確認が 6 分と計 14 分であり，全体の約 15%をログ検索が含まれる対応のステップが占める．したがって，ログ検索の時間を削減することでシステム障害解消までの時間を削減できる．

検索の応答時間を向上させる方法のひとつは，ログを分散配置して並列に検索する手法である [4]．並列化手法のひ

[*1] https://moneyforward.com/pages/premium
[*2] https://atmarkit.itmedia.co.jp/ait/articles/1705/09/news009.html

とつである分散手法はひとつのノードにログを配置して検索するのではなく，複数ノードにログを配置して並列で検索する手法である．分散手法の具体的な例は scatter-gather がある．scatter-gather は分散を制御するルートノードと，検索をするリーフノードの 2 種類のノードを使ったリクエスト処理の並列化手法である．

システム管理者はシステム障害を解消するためにログを検索して原因を特定する．このときログはアクセス時刻を基準に並んでいる必要がある．システム管理者は，過去のログを遡ることでシステム障害の根本原因やきっかけを辿ることができる．システム障害は前後関係があり、順序だてて起こる．システム管理者はアクセス時刻の順番に並んでいるログを参照してシステム障害の原因を特定する．

### 課題

システム障害発生時のログ検索における課題は，複数のアプリケーションノードからログを集めることで検索以外の処理であるアクセス時刻を基準にしたソートが，検索の応答時間に含まれることである．ソートが含まれることで，ログ検索の応答時間がその分増加する．また，検索の対象となるログが増えるほど検索の応答時間が増加する．

実際にソートによってどれだけ時間がかかるか予備実験をした．予備実験に用いたログは WordPress を使用したブログサイト[*3]のアクセスログである．ログの期間は 13 日，件数は 24,784 件である．予備実験はログ件数とソートの時間の関係性を調べるため，ログ件数を増やしてソートにかかる時間を計測した．図 1 は，予備実験で使用したログの複製方法と予備実験の手順を示している．ソートの対象となるログはアクセスログ 24,784 件を複製して増やした．測定した結果は 3 つであり，アクセスログ 1 つのみをソートした結果，もととなるアクセスログと複製したログを結合してソートした結果，もととなるアクセスログと複製したログ 2 つを結合してソートした結果である．ソートはアクセス時刻を基準に行った．ソート時間はログをソートしはじめてからソートし終わるまでとする．ソート時間はソート処理を 10 回行った平均値から算出した．ソートのアルゴリズムは，Linux の sort コマンドのアルゴリズムである，マージソートを使用した[*4]．計測は Linux の time コマンドを使用した．

図 2 は予備実験の結果を示している．図 2 より，ログ件数とソート時間は比例関係を示していた．ソート時間は検索の応答時間の中に含まれる．すなわち，ログ件数が増えるほどログ検索の応答時間がソートによって増えることを示している．


**図 1**　基礎実験の手順

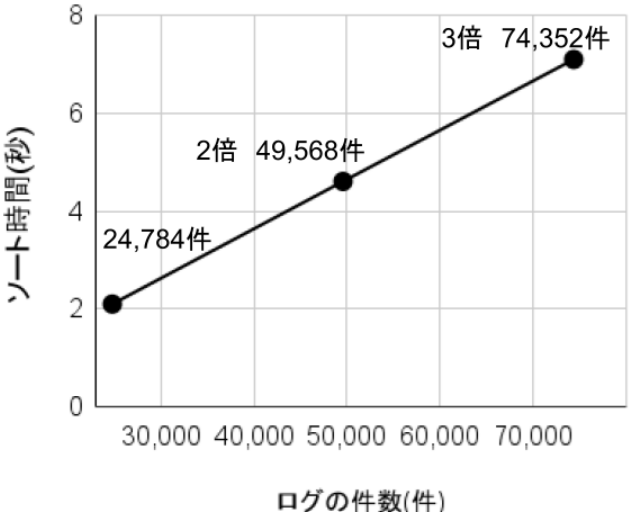


**図 2**　予備実験におけるログの件数とソートの時間

### 各章の概要

2 章では関連する既存研究を述べる．3 章では提案手法を説明する．4 章では実装方法と実験方法を述べる．5 章は，評価と分析の手法を説明する．6 章では提案手法の議論をする．最後，7 章は全体を簡潔にまとめている．

## 2. 関連研究

ある産業の売上量といった，特徴量を視覚的に色を用いて示す MAP を作成するアプリケーションを実装した研究がある [5]．563,000 以上のレコードがあるデータセットを場所，業界タイプ，主要製品タイプ，売上高，従業員を

*3 https://ja.tak-cslab.org/
*4 http://vkundeti.blogspot.com/2008/03/tech-algorithmic-details-of-unix-sort.html

キューブ化して分類している．この研究はデータセットをキューブ化してシステムが取り出しやすい形にしている．このキューブ化し，分類する考えはログ検索にも取り入れられる．

scatter-gather パターンを情報のアクセスツールとして捉えた研究がある [6]．この研究は scatter-gather におけるドキュメントクラスタリングは効果的な情報アクセスツールとなることを示している．ログ検索をする際に検索条件ごとのログファイルの塊を作ってアクセスする方法を用いれば scatter-gather パターンを使用したログ検索の高速化ができる．

## 3. 提案方式

### 提案方式

本提案は，複数のアプリケーションノードから集められたアクセスログをあらかじめアクセス時刻を基準にソートして，ステータスコードごとにログを分けて保存することでログ検索の応答時間を削減する．これによりログを検索するときは，ログをソートした状態でリーフノードに保存してあるため，リーフノードから集めた検索結果をルートノードでソートする時間を削減できる．また，ログをステータスコードごとに分けて保存しているため，検索の対象となるログを削減できる．ソート時間と検索対象を削減することは，検索の応答時間の削減に繋がる．アクセスログは，HTTP レスポンスステータスコードとアクセス時刻が入っている．また，システム管理者が発行する検索クエリは，HTTP レスポンスステータスコードとアクセス時刻順にソートする条件が付いているものとする．

図 3 はログの保存と検索のアーキテクチャを示している．はじめにログの保存について説明する．ユーザーのアクセスによって生成されたログはアプリケーションノードからルートノードへ送信する．アプリケーションノードから送信されたログはルートノード内で結合し，HTTP レスポンスステータスコードと時刻で分割される．分けられたログはアクセス時刻を基準にソートをした後ブロック化しリーフノードを通してディスクに保存される．検索条件であるステータスコードを基準に分割することで，検索の対象となるログを削減できる．ステータスコードは障害対応のための異常か正常かの判断基準として利用できる．次に検索する場合を説明する．システム管理者はクライアントを通してルートノードに検索クエリを発行する．ルートノードは各リーフノードに検索を要求し返答を受け取る．このとき，ルートノードはリーフノードからの結果を結合するだけでソート処理をすることなく検索結果を出力できる．検索時間はソート処理が不要なため削減できる．

図 4 はログの保存方法を示している．ログの保存は検索時にソート処理をしないためにあらかじめソートする．ルートノードで行う動作は 4 つに分けられる．

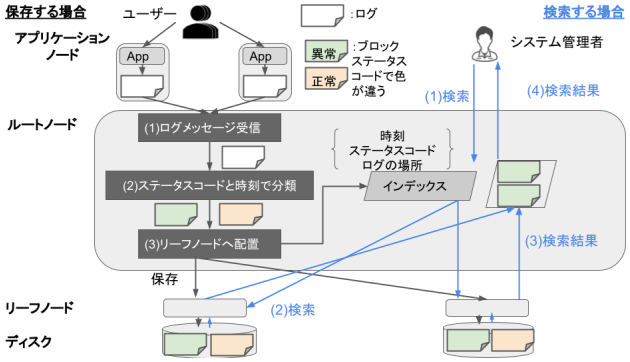


図 3　提案の全体図

(1) ログ結合
(2) ソート
(3) 属性分割
(4) ブロック化

まずログ結合はアプリケーションノードから集められたログを結合する．次にソートは結合されたログを HTTP レスポンスステータスコードと時刻でソートをする．ソートによってログは HTTP レスポンスステータスが同じものでまとまり，同じ HTTP レスポンスステータスコードの中でも古いログから順番に並ぶ．また，属性分割は同じ HTTP レスポンスステータスコードのログをまとめて分割する．最後に，ブロック化は分割したログをアクセス時刻が時間と分まで一致するログごとに分ける．分ごとに分ける理由は，システム管理者が検索条件に時間帯を指定するとき，分単位で指定することを想定しているからである．ブロック化したログはリーフノードへ順番に振り分け，ディスクに保存される．

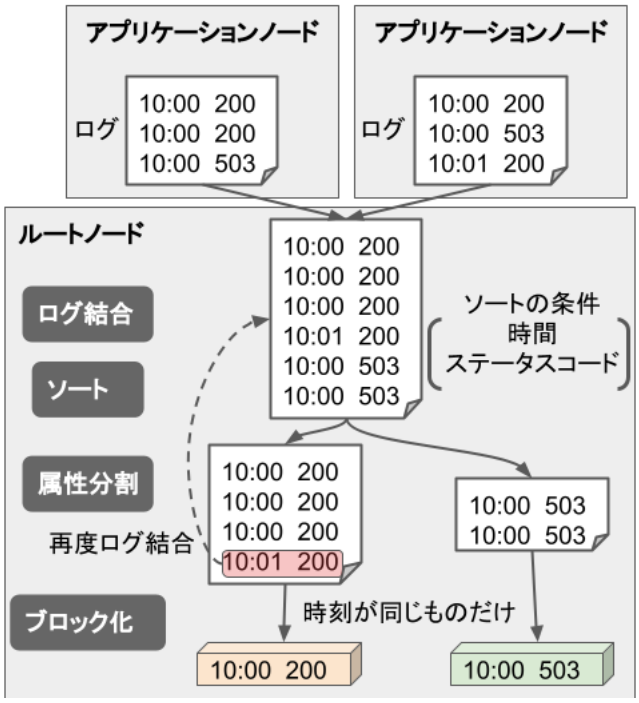


図 4　ログの保存方法

次に，システム管理者は HTTP レスポンスステータスコード 503 のログを検索するときの流れを例として説明する．図 5 はログの検索方法を示している．ログは保存するときにソートされているため，ルートノードはログを順番に結合するだけで古いログから順番に並んだ検索結果が得られる．検索の流れを以下に説明する．まずシステム管理者はクライアントを通してルートノードに検索クエリを送信する．ルートノードは各リーフノードに検索クエリを発行する．各リーフノードはルートノードから受け取った検索クエリに従い検索する．リーフノードの検索結果はルートノードに送信され結合される．結合した検索結果はクライアントに送信されて出力される．最後にシステム管理者はクライアントに出力された検索結果を確認する．

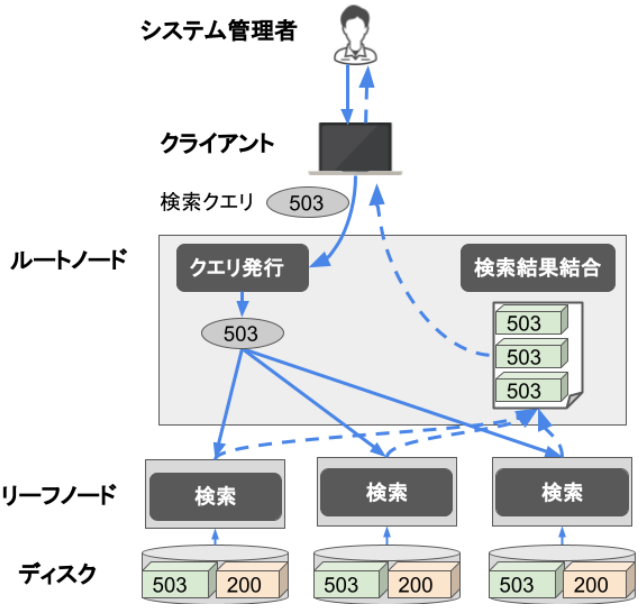


**図 5**　ログの検索方法

### ユースケース・シナリオ

図 6 は WordPress における HTTP レスポンスステータスコード 503 が発生したときのユースケース・シナリオを示している．検索する目的はシステム管理者が障害の原因を特定することである．保存されるログは WordPress のアクセスログとする．(1)，(2) は Web サービス利用者のコメント投稿でエラーが発生することを示している．(3) は Web サービスがログをルートノードに送信することを示している．(4) は Web サービス利用者がカスタマサポートに問い合わせすることを示している．(5) はカスタマサポートがシステム管理者へ調査を依頼することを示している．(6) はシステム管理者が原因調査のため，ログをルートノードに検索することを示している．また，システム管理者は検索クエリの条件を HTTP レスポンスステータスコードとアクセス時刻を基準にソート，時間帯を分単位で指定するものとする．

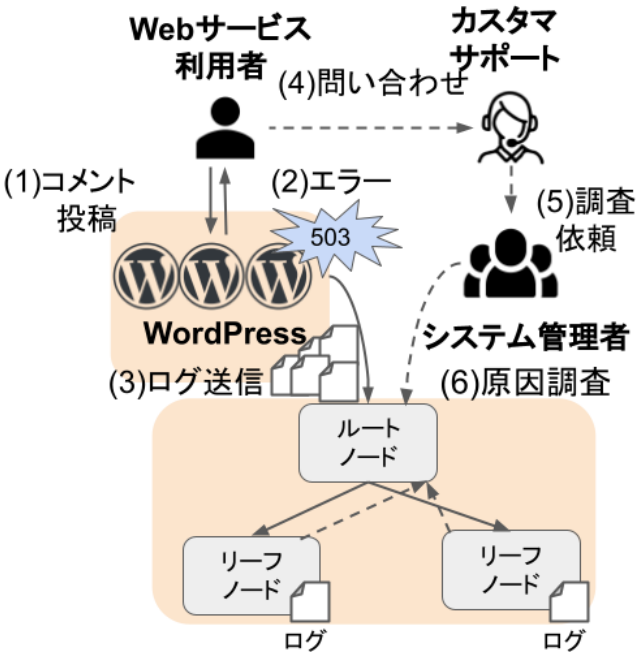


**図 6**　ユースケース・シナリオ

## 4.　実装と実験方法

提案方式をもとに開発したソフトウェアやハードウェアの実装と実験方法について説明する．

### 実装

図 7 と図 8 は，ルートノードのソフトウェア構成図を示している．ソフトウェアの構成を以下に示す．

(1) ログメッセージ受信
(2) ステータスコードとアクセス時刻を取得
(3) ログファイルに追記
(4) アクセス時刻順にソート
(5) インデックス保存
(6) リーフノードへ送信

(1) のログメッセージ受信は，ログメッセージをアプリケーションノードから受け取り，ログメッセージを 1 件ごとにステータスコードと時刻で分割へ送信する．

図 7 は提案の章でも述べたステータスコードとアクセス時刻で分割を示している．(2) のステータスコードとアクセス時刻を取得は，ログメッセージからステータスコードと時刻を読み取る．(3) のログをファイルに追記は，ステータスコードと時刻の分まで一致するログをファイルに追記して分類する．

図 8 は，提案の章で述べたリーフノードへ配置を示している．ログを 1 分単位で分割するため，(4) から (6) は 1 分おきに実行する．1 分単位で分割する理由は，検索条件に合わせて検索範囲を限定するためである．システム管理者が検索条件を時刻でつけるとき，秒単位ではなく分単位で指定することを想定してログを分割している．(4) のア

クセス時刻を基準にソートは，同じステータスコードと時刻を持つログをアクセス時刻順にソートする．(5) のインデックス保存は，ソートしたログの配置場所とアクセス時刻とステータスコードを関連付けて保存する．(6) のリーフノードへ配置は，リーフノードの通し番号の順番にログを転送する．

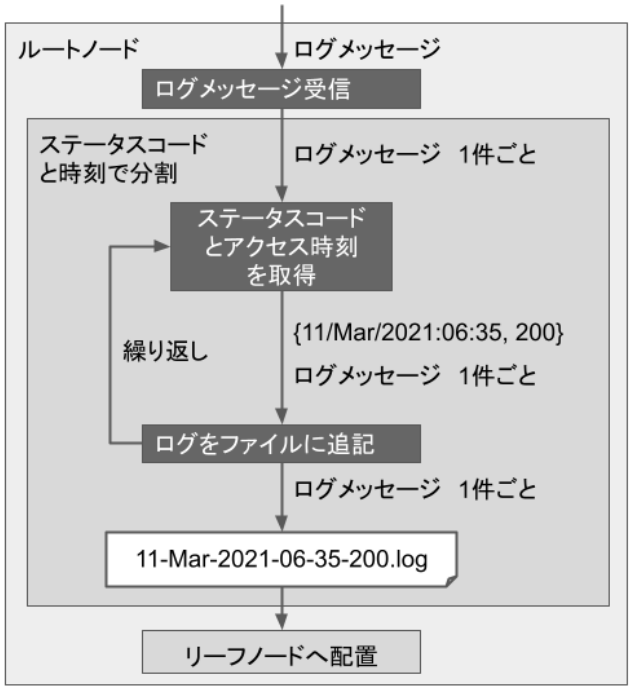


**図 7**　ルートノードのステータスコードで分割のソフトウェア構成図

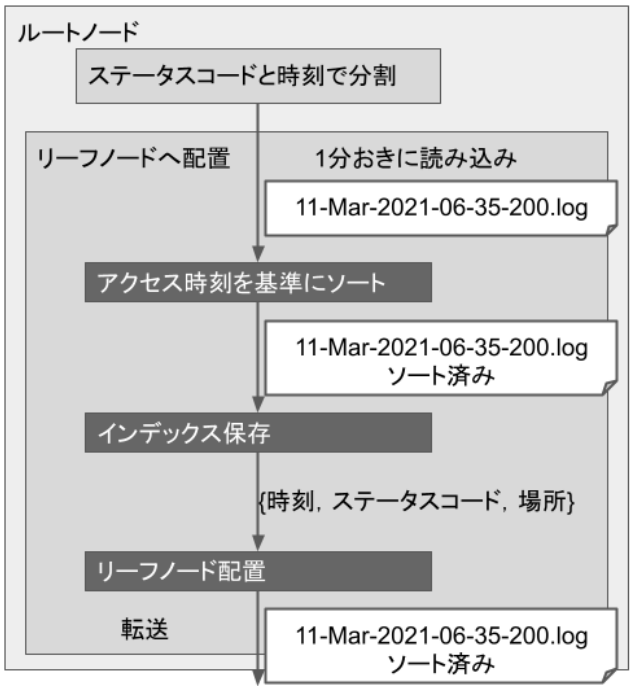


**図 8**　ルートノードのリーフノードへ配置のソフトウェア構成図

図 9 は，リーフノードのソフトウェア構成図を示している．リーフノードはルートノードからブロックを受け取り保存する．保存したブロックは HTTP レスポンスステータスコードとアクセス時刻の 2 階層になっている．検索時に検索範囲を限定して検索の応答時間を削減するために，ディレクトリは 2 階層に分けている．

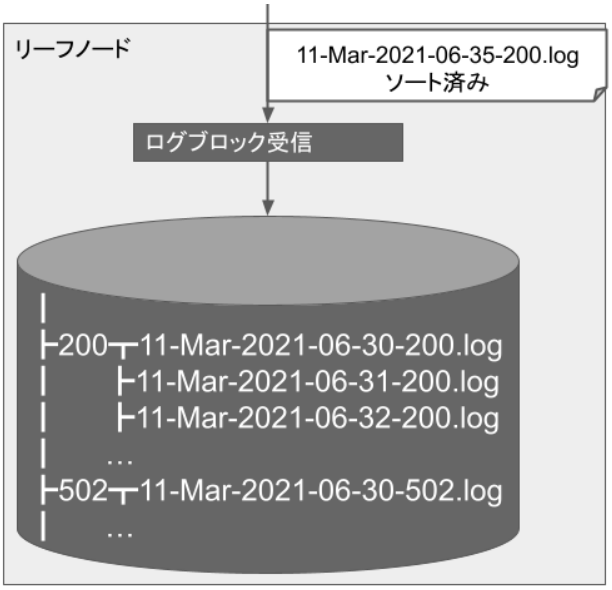


**図 9**　リーフノードのソフトウェア構成とディレクトリ構造

**実験環境**

実験はハードウェア上にインストールしたハイパーバイザー (VMware ESXi) で行う．ハイパーバイザーに配置する仮想マシン (Ubuntu 20.04.2 LTS) はハードウェア性能 (CPU: 1[コア]，RAM: 4[GB]，Storage: 32[GB]) を持つ．

## 5.　評価手法と分析手法

図 10 は，検索時にソートしたときと保存時にソートしたときの検索の応答時間を示している．ログをランダムに生成して検索の応答時間を比較した．想定は 1 分あたり 10,000 リクエストとし，2 つのアプリケーションノードからログが集められるものとする．1 時間分である約 600,000 件のログから約 8,000 件のログを基準に検索の応答時間を比較した．評価時に用いた検索条件は HTTP レスポンスステータスコード 502 とした．既存手法は，2 つのログファイルから検索条件に当てはまるログを見つけ出し，ログをアクセス時刻順にソートした．検索の応答時間とソート時間の合計を測定した．提案手法はあらかじめ 2 つのログファイルをアクセス時刻順に並び替える．並び替えたログファイルを検索の応答時間を測定した．時間の測定方法は Linux の time コマンドを使用し，検索には grep コマンドを使用した．約 3,000,000 件のログを検索した時，検索時にソートは 1,400[ms] で保存時にソートは 10[ms] であった．上記の分析により，検索の応答時間の中で検索の時間

よりもソートの時間が多くの時間を占めていると実験結果が得られた．

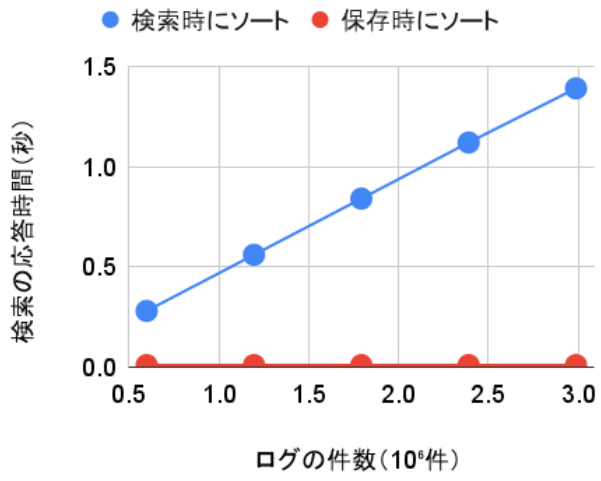


**図 10** 検索の応答時間の比較

## 6. 議論

評価において，実験はノードひとつでしている．本提案は分散検索手法であるため，複数ノードにログを分散した検索の応答時間を評価する必要があった．

本提案において，ログをリーフノードへ保存した後に，アプリケーションノードからログがルートノードへ送信される場合がある．このときリーフノードへ保存されたログは，ログがアクセス時刻を基準に並んでいない．アクセス時刻を基準に並んでいる状態でリーフノードへ保存しなければ，ログを検索するときに検索結果はアクセス時刻を基準に並んでいないことになる．したがって，ログがアクセス時刻を基準に並んでいる状態を維持する手法を提案する．ログがリーフノードへ保存した後にルートノードへ送信された場合，本提案と同じようにログを結合してソートして属性分割した後，ブロック化を行わず，リーフノードへ保存してあるブロックへログを送信する．後から来たログをリーフノードで結合し，ソートすることでブロック内のログがアクセス時刻を基準に並んでいる状態を維持する．

## 7. おわりに

課題は，検索結果をソートすることで検索結果の件数が増えるほどログ検索の応答時間が増加することである．本提案は，検索をするときに，ソート時間を削減したログの配置をする．本提案は検索条件として HTTP レスポンスステータスコードに着目してログを分割した．分割したログは時刻順にソートして保存することで，検索時にソートの時間を削減したログのブロックを作成した．ブロックはリーフノードへルートノードから振り分ける．検索時はブロックをルートノードに送信し，結合することでソートの時間を削減したログ検索システムを実現した．実験はログをランダムに生成して仮想マシン上にログを配置して検索時にソートしたときと保存時にソートしたときの検索の応答時間を比較した．検索の対象のログは約 600,000 件のログから約 8,000 件のログを検索結果として出力する．約 3,000,000 件のログを検索した時，検索時にソートは 1,400[ms] で保存時にソートは 10[ms] であった．本提案は，障害が解消するまでの時間に含まれるログ検索の応答時間の削減できた．

## 参考文献

[1] He, P., Zhu, J., He, S., Li, J. and Lyu, M. R.: Towards Automated Log Parsing for Large-Scale Log Data Analysis, *IEEE Transactions on Dependable and Secure Computing*, Vol. 15, No. 6, pp. 931–944 (online), DOI: 10.1109/TDSC.2017.2762673 (2018).

[2] 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 (online), DOI: 10.1109/65.844498 (2000).

[3] Ali, S. K. and Sadik, S.: Web Services Monitoring, Analysis for Future Usage and Failure Prediction.

[4] Hamadi, Y.: Distributed interleaved parallel and cooperative search in constraint satisfaction networks, *Proc. IAT*, Vol. 1 (2001).

[5] Guo, D., Chen, J., MacEachren, A. and Liao, K.: A Visualization System for Space-Time and Multivariate Patterns (VIS-STAMP), *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 6, pp. 1461–1474 (online), DOI: 10.1109/TVCG.2006.84 (2006).

[6] Cutting, D. R., Karger, D. R., Pedersen, J. O. and Tukey, J. W.: Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections, *SIGIR Forum*, Vol. 51, No. 2, pp. 148 – 159 (online), DOI: 10.1145/3130348.3130362 (2017).