ネットワーク管理におけるイベントのリアルタイム識別の実現のためのログ要約手法の提案と評価

長尾真宏[†] 北形元[†,††]
菅沼拓夫[†,††] 白鳥則郎[†,††]

本論文では，既存の異常検出手法により検出されたイベントの識別を支援する「ログ要約手法」を提案する．提案手法の特徴は，段階的なログ情報の絞り込みにより，リアルタイム性を維持しつつ高い精度でイベントの識別に必要なログ情報のみを抽出できることである．これより，大量のログ情報から，イベントの識別に必要な情報のみを抽出してリアルタイムに管理者に提供することが可能であり，管理者のイベントへの対応時間の短縮および作業負担の軽減に大きく貢献できる．さらに実証実験を通じ，提案手法は従来手法と比較して，ログのレコード数を 6 割から 9 割以上削減でき，かつイベントの識別に必要なレコードが削減されないこと，および多数のレコードを含むログに対してリアルタイムに処理可能であることを示す．

Log Summarizing Method for Real-time Event Recognition in Network Management and Its Evaluation

Masahiro Nagao,[†] Gen Kitagata,[†,††] Takuo Suganuma[†,††] and Norio Shiratori[†,††]

In this paper, we propose a *log summarizing method* to help in network management to recognize a network event. Our method can accurately extract only necessary information to recognize the network event in real-time by introducing the stepwise information extraction. This feature can help the network manager to handle the event quickly with less management cost. With our experimental results, we show that our method has the ability to reduce more than 60%–90% unnecessary records compared to traditional method, without losing any necessary information. Additionally, our method can process a log, containing massive records in real-time.

1. はじめに

ネットワーク管理者は，適切な運用管理をするうえで重要な障害，不正アクセスなどのイベントに対し，迅速に「検出/識別/応急処置」を行うことを求められている．従来,「検出」の手法は多く研究されており，なかでも異常検出手法は，これまで検出が困難であったイベントを，高精度で検出可能とする成果をあげている[3)~6)]．異常検出手法では，検出されたイベントの内容を識別する作業がより重要となるが,「識別」の研究は少ないため，異常検出手法により検出されたイベントの内容把握に時間を要し，全体として迅速なイベント処理が実現されていなかった．

そこで本論文では，既存の異常検出手法により「検出」されたイベントを対象に，リアルタイムでのイベントの「識別」の実現へ向けて，対応時間の短縮および作業負担の軽減を可能とする「ログ要約手法」を提案する．ログ要約とは，膨大な数のレコードを含むすべてのログの中から，イベントの識別に必要なログのレコードのみを抽出することである．管理者は，このようなログ要約を用いることで，短い作業時間かつ少ない作業負担でリアルタイムに，すなわち，与えられた制約時間以内に，イベントの識別作業を行うことが可能となる．

提案のログ要約手法を実現するための課題として，1) リアルタイム性を確保し，2) 1) の制約下でできるだけ高精度に，膨大なログから必要ログのみを抽出すること，という 2 点があげられる．本提案手法の特徴

† 東北大学電気通信研究所/情報科学研究科
Research Institute of Electrical Communication/ Graduate School of Information Science, Tohoku University
†† 情報通信研究機構東北リサーチセンター
NICT Tohoku Research Center

は，これらの問題を解決するため，段階的な情報の絞り込みを行うことにある．ここで，提案手法のリアルタイム性とは，ある時間間隔を設定し，ある期間中に生成されたログを，次の期間中に処理できることを示す．さらに，実装したプロトタイプを用いた実環境での実験を通じ，有効性を示す．

実験の結果，ログ要約により提供されるログのレコードは，イベントの識別に必要なレコードを残しつつ，不要なレコードの大部分を削減することに成功した．単純にイベントの発生時刻を用いてログを切り出す従来手法と比較して，少なくとも 6 割程度，多ければ 9 割以上のレコードを削減することができた．また，従来人手で数十分から数時間かけて行われていたイベントの識別作業を，提案手法を用いることで，早ければ数十秒，遅くとも 5 分以内に短縮することができ，リアルタイムでのイベントの識別が実現できる．これにより，管理者がログを解析する負担を大幅に削減できることを示す．

2. 関 連 研 究

2.1 不正検出手法によるイベントの検出と識別

イベントの種類は様々であり，検出の方法も異なる．たとえば，リンクダウンやサーバダウンというイベントの検出方法は，定期的なポーリングによる死活監視が一般的である．このように，イベント発生時の特徴が明確な場合には，その特徴を検出条件として与え，イベントを検出すればよい．これは不正検出と呼ばれる手法である．この手法でイベントを検出した場合，検出条件とイベントの特徴が密接に関係しているため，検出したイベントの識別は容易に行える．その関係を用いて，イベントの識別やその後の対応作業の支援を行い，管理負担を軽減する手法も提案されている[1),2)]．

2.2 異常検出手法によるイベントの検出と識別

一方，たとえばワームの感染というイベントの場合，症状が固定的でなく新種が日々出現する．ゆえに，そのたびに検出条件の追加が必要となるため，不正検出手法では非効率的であり，対応しきれない場合がある．そこで，異常検出と呼ばれる手法により，トラフィックパターンの変化からイベントを検出する研究が行われてきた[3)〜6)]．それらは，正常状態/異常状態を抽象化し，イベントの検出を一般化・汎用化することで，新種のイベントを検出可能とした．しかし，検出後に「識別/応急処置」を行う際，抽象化された「異常」という判定だけでは情報が著しく不足するため，管理者はイベントの具体的な情報を得るために大量のログを解析し，関連する情報を探し出し，整理する必要がある．これは高度な経験・知識を必要とし，また，システムの大規模化にともない迅速な対応が困難となり，作業負担が非常に大きくなる．

2.3 異常検出手法におけるイベントの識別の課題

上述したように，異常検出手法は不正検出手法では検出できない，より多種のイベントを検出可能であるが，その反面，検出された異常の識別に手間がかかるというデメリットがある．そのため，異常検出手法に適用可能な，イベントの識別を効果的に支援する手法が大きく期待されている．

3. ログ要約手法の提案

3.1 提案手法の概要

本論文では，上述の異常検出手法におけるイベントの識別の課題を解決し，迅速なイベント処理を実現するため，リアルタイムという制約の下，できるだけ高い精度でログ要約を行うための 3.2，3.3，3.4 節で述べる段階的な情報の絞り込みを行うログ要約手法を提案する．**図 1** に，提案手法によるログ要約結果の具体例を示す．図 1 上段はイベントに関連性の高い代表的なレコードを，下段に異常検出時刻付近のレコード全体の統計値を示している．次節より，以下の用語を用いて提案手法における各段階の処理手順とアイデアについて述べる．

イベント：ネットワーク管理者が管理上重要と見なす出来事である．たとえば，リンク/サーバ障害，不正アクセス，アクセス過多などがあげられる．

異常：異常検出手法において，観測したトラフィックが正常の値の範囲を超えた状態を異常とする．たとえば，送信パケット数が多すぎる，などの異常がある．

ログとレコード：アプリケーションが動作履歴として出力する情報の集合をログとし，その動作履歴の最小単位をレコードとする．本論文では，テキスト形式のログを対象とし，これをログファイルと呼ぶ．また，ログファイルの 1 行分をレコードとして扱う．

3.2 異常検出結果に基づく注目ログ・注目情報の絞り込み

第 1 段階として，後続の処理における解析量を削減してリアルタイム性を確保し，また解析精度を向上させるため，処理対象となる情報を限定する．処理対象となる情報とは，具体的には次の (A)，(B) の 2 点，すなわち (A) どのログを解析するか，(B) ログ内のどの記述に注目するか，である．たとえば，HTTP トラフィックにおける異常検出の場合，(A) は HTTP サーバのログとなり，(B) は HTTP サーバのログにおける転送量やリクエスト文字列などの記述となる．

```
2005/06/05 (Sun) 06:19:17 - 06:19:26: [count: 79 / 95, freqency: 0.0388888888889 -> 15.8 /min]
  61.100.180.13 - - [05/Jun/2005:06:19:17 +0900] "GET // adm/awstats/awstats.pl HTTP/1.1" 400 377 "-" "-"
  61.100.180.13 - - [05/Jun/2005:06:19:18 +0900] "GET // awstats.pl HTTP/1.1" 400 377 "-" "-"
  61.100.180.13 - - [05/Jun/2005:06:19:26 +0900] "GET //webstats/awstats.pl HTTP/1.1" 404 304 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)"

total sample count: 95
new cluster: 0
frequency: 7.41111111111 -> 19 /min
```

図 1 ログ要約結果の例
Fig. 1 An exapmle of log summarizing.

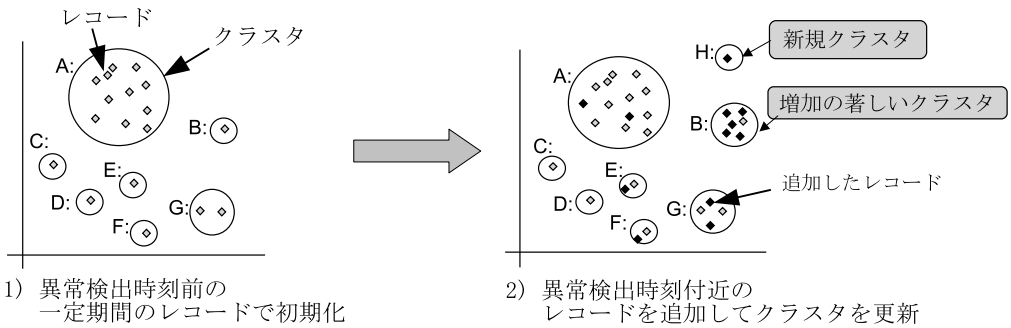


図 2 クラスタリングによるレコードの分類手順と重要クラスタの選択の例
Fig. 2 An example of categorizing records by clustering and selecting significant clusters.

3.3 クラスタリングによるログのレコードの分類

第 2 段階として，対象ログそれぞれついて，対象ログごとにログ内のレコードを個別にクラスタリングにより分類する．これは以下の 2 つの手順に分けられる．

3.3.1 ログの各レコードに対応する特徴ベクトルの生成

ログのレコードの記述を，単独で意味を持つ「フィールド」単位に分割し，各フィールドを一定の規則に従い数値化して特徴量とする．この特徴量をまとめ，特徴ベクトルを生成する．これにより，ログの各レコードがベクトルで表現され，レコードの分類を特徴ベクトルのクラスタリングとして実現できる．なお，特徴ベクトルの生成の際は，つねに全フィールドを利用するのではなく，3.2 節で得られた注目情報に基づき必要なフィールドを絞り込む．これによりリアルタイム性を確保しつつ，高い精度でクラスタリングを行うことが可能となる．

3.3.2 特徴ベクトルに基づくレコードのクラスタリング

特徴ベクトルの生成後，そのベクトルの集合に対してクラスタリングアルゴリズムを適用する．**図 2** に，クラスタリングによるレコードの分類手順の例を示す．具体的なクラスタリングの手順は以下のようになる．

(1) 異常と判定される直前の一定期間のレコードから生成した特徴ベクトルを用いてクラスタリングを行い，初期クラスタを生成する．
(2) 異常検出時刻付近のレコードを追加しクラスタリングを行い，初期クラスタを更新する．

(1) の結果は，対象ログの通常時の振舞いを表し，(1) と (2) の結果の差異は，通常時のログの振舞いと，異常検出時のログの振舞いの差異を表している．

なお，本手法では，誤判定を最小限に抑えるため，類似度の非常に高い特徴ベクトルのみを同一クラスタとするように，同一クラスタと見なす類似度の閾値を設定する．このような閾値の設定は，クラスタ数を増大させ処理を非効率にする可能性があるが，提案手法では，前段階の 3.2 節の情報の絞り込みにより注目する情報が限定されているため，クラスタ数を少なく抑え，処理を効率的に行うことが可能である．

3.4 クラスタリング結果に基づく重要クラスタの抽出

第 3 段階として，クラスタリング結果からイベントを特徴づける重要クラスタを抽出する．重要クラスタとは，前節の (1) と (2) で大きな差があるクラスタであり，たとえば大きくなったクラスタや，異常検出時刻付近で新規に発生したクラスタなどである．図 2 の例では，B:の増加の著しいクラスタ，および H:の新規クラスタを重要クラスタとしてあげている．

最後に，抽出した重要クラスタについて，要素数などのクラスタの特徴と，代表値としてクラスタ内の任意の数個のレコードを記述したログ要約結果を出力する．その理由は，3.3 節で述べたとおり，類似度の非常に高いレコードのみ同一クラスタに含まれており，クラスタの全レコードを見なくても任意の数個のレコー

ドによりクラスタの特徴が把握できるためである．

4. 提案手法に基づくログ要約システムの設計と実装

4.1 提案手法に基づくログ要約システムの概要

図 3 に，提案手法に基づくログ要約システムの概要を示す．異常検出モジュールは，トラフィックを利用し，異常検出手法により，(1) 異常検出時刻と，(2) 異常検出手法が判断に使用した時間範囲，および (3) トラフィックデータの種類，の 3 種の情報を異常発生情報として出力する．本論文では，トラフィックを用いた異常検出を対象とするが，(3) データの種類として他の情報を用いることも可能である．

ログ要約モジュールは，ログと異常検出モジュールの出力する異常発生情報を用いて，3 章で提案したログ要約処理を実行し，ログ要約結果をネットワーク管理者に提供する．**図 4** に，ログ要約モジュールの概要を示す．ログ要約モジュールは，1) 注目ログ・注目情報の絞り込み機能，2) レコードの分類機能，3) 重要クラスタ抽出機能，の 3 つの機能を持つ．以下，それぞれの機能の設計について述べる．

4.2 注目ログ・注目情報の絞り込み機能の設計

トラフィックデータの種類を次の 2 つの観点で考え，各観点から第 1 段階の異常検出結果に基づく情報の絞り込み処理を行う．

(1) 異常検出に用いたトラフィックのプロトコルの種別

(2) バイト数，パケット数，アクセス数など異なる注目情報での統計値

(1) は 3.2 節 (A) の対象ログの絞り込みに使用する．たとえば，HTTP トラフィックでの異常検出時は，HTTP サーバのログを解析する．一方，一般に DNS トラフィックの異常検出時には，DNS サーバのログだけでなく，メールサーバのログにも影響が見られる[7]．

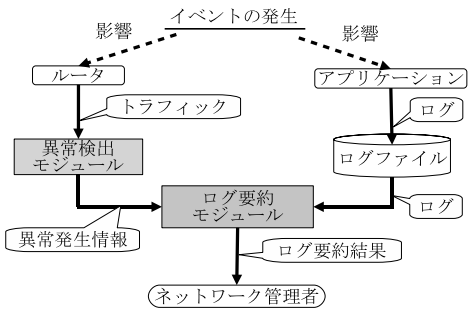


図 3 提案手法に基づくログ要約システムの概要
Fig. 3 An overview of the log summarizing system based on proposed method.

このようなプロトコルとログの関連を明示的に定義しておくことにより，解析対象ログを選択する．

(2) は 3.2 節 (B) の関連の深い記述の絞り込みに利用する．たとえば，バイト数において異常が検出された場合には，転送ファイルサイズやファイル名などの，コンテンツの情報に注目する．また，アクセス数において異常が検出された場合には，リモートホストやセッションの頻度などの，リクエストの特徴に注目する．

4.3 レコードの分類機能の設計

4.3.1 ログの各レコードに対応する特徴ベクトルの生成

特徴ベクトルの生成は，次の 3 種類の手順で行う．

(a) 離散値や単語などで表現され，値のバリエーションが少ないフィールドは，各値に数値を割り振る．

(b) 連続値をとる数値フィールドはその数値を用いる．

(c) 自由記述の文字列フィールドは，あらかじめランダムに選択した基準レコードの当該フィールドとの間で，それぞれの部分文字列から Levenshtein 距離[8] を求め数値化する．また，文字列長も 1 つの特徴量として用いる．

たとえば HTTP サーバのアクセス履歴であれば，(a) は HTTP ステータスコード，(b) は転送ファイルサイズ，(c) はリクエストパスなどがある．

4.3.2 特徴ベクトルに基づくレコードのクラスタリング

クラスタリングのアルゴリズムとして単一パス法[9] を採用する．単一パス法は処理を 1 パスで完了するため，ベクトル数すなわちレコード数の増加に対してスケーラビリティがある．精度については，ベクトル間の類似度の尺度および閾値の影響が大きい．本手法では，ベクトル間の類似度として，文章比較などに用い

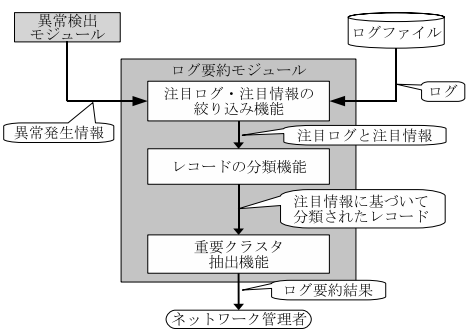


図 4 ログ要約モジュールの概要
Fig. 4 An overview of the log summarizing module.

られるコサイン尺度を用いた．ベクトル $\boldsymbol{v_i}$ と，重心ベクトル $\boldsymbol{C_j}$ を持つクラスタとの類似度は

$$\cos(\boldsymbol{v_i}, \boldsymbol{C_j}) = \frac{\boldsymbol{v_i} \cdot \boldsymbol{C_j}}{||\boldsymbol{v_i}||\,||\boldsymbol{C_j}||}$$

と表される．この尺度は，ベクトルの各次元のスケールの差を吸収でき，計算量が少ないため，様々なフィールド値から構成される特徴ベクトルを，リアルタイムに処理する必要のある本手法に適している．なお，閾値は 3.3.2 項で述べたとおり，非常に高い類似度を持つレコードのみが同一クラスタとなるように設定する．

4.4 重要クラスタ抽出機能の設計

異常検出時にログに現れる影響として，(1) レコード数に特徴がある場合，(2) レコード数ではなくレコード内のフィールドに特徴がある場合，の 2 つがあり，(1) はさらにレコード数が多い場合とレコード数が少ない場合に分けられる．

(1) の場合，レコード数の増減により重要クラスタを判別可能である．一方，(2) の場合，レコード数の増減で重要クラスタを判別できないため，別の基準を用いる必要がある．そこで，レコードの割合が 0 または小さいクラスタが，異常検出時にある程度大きい割合に変化した場合にそれを重要クラスタとする．また，異常と関係するレコード数が少ない場合が考えられるので，レコード数の非常に少ないクラスタも重要クラスタとする．本論文では，これらの基準を用いて 3.4 節で述べた機能を実現するが，より適切な判定方法については，今後の検討課題とする．以下に重要クラスタの判別条件をまとめる．

図 5 に重要クラスタの決定条件に用いるパラメータを示す．初期クラスタの生成に使用する異常検出前の一定期間の時間範囲を T_1，その長さを τ_1 とし，異常検出手法が判断に使用した時間範囲を T_2，その長さを τ_2 とする．クラスタ i についての T_1 のレコード数を n_{1_i}，T_2 のレコード数を n_{2_i} とし，全レコードについての T_1 のレコード数を N_1，T_2 のレコード数を N_2 とする．また，クラスタ i の平常時のレコード密度を $d_{1_i} = n_{1_i}/\tau_1$，異常時のレコード密度を $d_{2_i} = n_{2_i}/\tau_2$ とし，全レコードについての平常時のレコード密度を $D_1 = N_1/\tau_1$，異常時のレコード密度を $D_2 = N_2/\tau_2$ とする．α_x は定数の閾値とする．

(1) レコード数が多い異常の場合
異常時において，全体に占める割合が大きいクラスタ，すなわち

$$\begin{cases} n_{1_i} > 0 \text{ かつ } n_{2_i}/N_2 \geq \alpha_{n_1} \\ n_{1_i} = 0 \text{ かつ } n_{2_i}/N_2 \geq \alpha_{n_2} \end{cases}$$

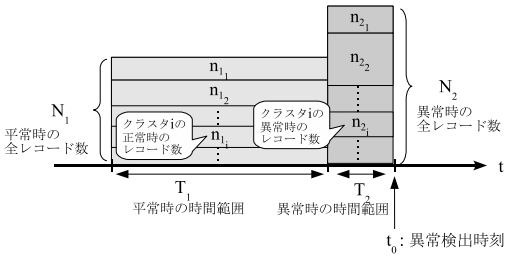


図 5 重要クラスタの決定条件に用いるパラメータ
Fig. 5 Conditional parameters for selecting significant clusters.

を満たすクラスタを重要クラスタとする．これはレコード数が多くなった原因の多数のレコードを含むクラスタである．

(2) 内容の特殊なレコードが現れる異常の場合
異常時において，全体に占める割合が小さいクラスタ，すなわち

$$n_{2_i}/N_2 \leq \alpha_{n_3}$$

を満たすクラスタ，またはレコード総数にかかわらず平常時に対する異常時のレコード数の増加が大きいクラスタ，すなわち

$$D_2/D_1 \geq \alpha_D \text{ かつ } d_{2_i}/d_{1_i} \geq \alpha_d$$

を満たすクラスタを重要クラスタとする．前者は記録されることが珍しいレコードからなるクラスタで，後者は平常時に比べ，異常時のレコード数の割合が非常に大きくなったクラスタである．

なお，レコード数が少ない場合については，検証環境で定常的なアクセスが多く見込めないため，本論文では扱わないこととした．

これらの条件の下，重要クラスタを抽出し，さらに 3.4 節で述べたとおり，それらの重要クラスタに含まれるレコード数，レコードの時間範囲，任意の数個のレコードを代表値として記述し，ログ要約を完了する．

4.5 提案手法に基づくログ要約システムの実装

ログ要約システムのプロトタイプを，Linux 2.6.13 上で，PHP，Perl およびシェルスクリプトを用いて実装した．プログラムの総行数は約 1,300 行である．

5. 実験および評価

5.1 実験の目的と概要

5.1.1 目　　的

提案手法により，(1) 管理者が解析する必要のあるログの量を大きく削減でき，かつ必要な情報を提供可能であること，(2) リアルタイムで実現可能であることを検証し，提案手法が管理負担の軽減に大きく貢献

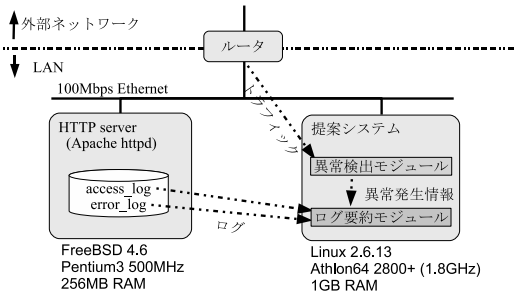


図 6 実験環境
Fig. 6 Experimental environment.

できることを示すため，提案手法に基づくログ要約システムのプロトタイプを実装し，このプロトタイプを用いて実際に運用しているネットワークにおいて実験を行った．実験項目は，(1) 不要なログのレコードの削減量の計測，(2) アルゴリズムの計算時間の計測，の 2 つである．

5.1.2 諸　　元

図 6 に実験環境を示す．LAN 内に配置された HTTP サーバ（apache httpd [10]）は普段より運用されているものであり，一般利用者からのアクセスが存在する．そこに新たに提案システムを配置して実験を行った．提案システムに入力するログは，HTTP サーバから得られるアクセス履歴ログ（access_log），エラー履歴ログ（error_log）の 2 つを用いた．

異常検出モジュールでは，ルータを通過する HTTP トラフィックの，バイト数/パケット数の入出力およびアクセス数の計 5 種類を，それぞれ 5 分ごとの合計値でサンプリングした値を入力として用い，異常検出アルゴリズムとして deviation score [4] を用いて異常発生情報を出力した．

ログ要約モジュールでは，クラスタリングの閾値として 0.99 を用い，4.4 節で述べた重要クラスタ判定条件の定数値は，$\tau_1 = 3$ 時間，$\tau_2 = 5$ 分間，$\alpha_{n_1} = 0.1$，$\alpha_{n_2} = 0.05$，$\alpha_{n_3} = 0.05$，$\alpha_D = 2$，$\alpha_d = 10$ とした．重要クラスタの抽出後，代表値として抽出するレコードは，時系列順で先頭 2 つおよび末尾 1 つとなる計 3 レコードとした．これらのパラメータについては，トラフィック計測が 5 分単位であることを考慮し，実験的に決定したが，最適なパラメータを求めることは今後の課題とする．

リアルタイムとは与えられた制約時間以内に処理を完了することである．本実験では，入力トラフィックのサンプリング間隔以内にログ要約処理を完了できる場合にリアルタイム性を実現しているとするが，トラフィック計測や異常検出が対象とする時間スケールに応じて，実用的な制約時間を定めればよい．

5.2 ログ要約処理の正確性の評価

5.2.1 HTTP アクセス数の異常検出時における不要レコードの削減の正確性

表 1 に，HTTP アクセス数において異常が検出された場合の access_log，error_log のレコード数の削減結果を示す．計 6 回の異常検出時における提案手法の実行時において，(A)〜(D) に提案手法の各段階のレコード数を，(E)，(F) にはクラスタリングと重要クラスタ判定の誤りから生ずる (E) false positive，(F) false negative のレコード数を示している．それぞれのケースのアクセスの内容としては，1，3，5，6 はサーバ内の多数のファイルに短時間にアクセスするクロール行為であり，2 は AWStats [11] というアクセス解析 cgi へのアタック，4 は同一ファイルへの短期間で高頻度なアクセスで，DoS ツールのようなものと思われる．いずれも実際の運用の中で観測されたアクセスである．また，**図 7**，**図 8** に，この 6 回の結果にそれぞれついて，(B) のレコード数を 100%としたときの (C)，(E)，(F) のレコード数の割合を示した．

HTTP アクセス数の異常検出時は，イベントに関連するレコードが大量に記録されるため，(C) の段階では削減量は多くないが，類似度の高いレコードが大量に現れるため，同一クラスタに含まれるレコードが多く，クラスタ数が少なくなるため，(D) の段階において大幅な削減が可能である．(E) false positive，(F) false negative については，(E) が多数存在しても作業負担への影響は小さいが，(F) が多数存在するとイベントの識別に支障をきたす可能性が高く，よって (F) が少ないことが望ましい．実験結果より，ケース 2，3，4，5 において (F) は 0 である．また，(F) が 0 でないケース，すなわち，ケース 1，6 においても，(F) の値はそれぞれ 32，49 であり，(C) の重要クラスタに含まれたレコード数 499，832 に比べ，十分小さい．このような観点から，提案手法は精度良く不要なレコードを削減できているといえる．

以上より，HTTP アクセス数の異常検出時において，提案手法は高い精度で不必要なレコードを削減することができ，管理者がログを解析する負担を大幅に減らせることを確認した．

5.2.2 HTTP バイト数・パケット数の異常検出時における不要レコードの削減の正確性

表 2 に HTTP バイト数・パケット数から異常が検出された場合の access_log，error_log のレコード数の削減結果を，**図 9** に表 2 (B) を 100%とした場合の (C)，(E)，(F) のレコード数の割合を示す．それ

表 1 HTTP アクセス数の異常検出時の access_log，error_log のレコード数の削減結果
Table 1 The reduction in the number of records of access_log and error_log when anomalies related to the number of HTTP access were detected.

対象ログ	access_log						error_log					
ケース ID	1	2	3	4	5	6	1	2	3	4	5	6
(A) クラスタリングに使用した数	4,537	1,450	4,692	3,480	4,442	2,986	581	179	350	272	245	230
(B) 異常検出時刻付近の数	543	116	683	355	382	966	214	98	56	19	12	93
(C) 重要クラスタに含まれる数	499	79	492	295	332	832	183	95	47	16	10	77
(D) 重要クラスタの代表値とした数	6	3	3	3	3	3	6	12	3	3	3	6
(E) (C) に含まれる必要のない数	0	0	33	11	0	0	31	16	47	16	0	0
(F) (C) に必要だが含まれない数	32	0	0	0	0	49	26	0	0	0	0	0

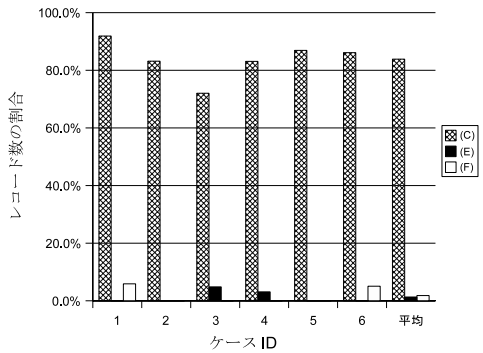


図 7 表 1 (B) を 100%とした access_log の各種レコード数の割合
Fig. 7 The ratio of remaining records of access_log when (B) in Table 1 is assumed to be 100%.

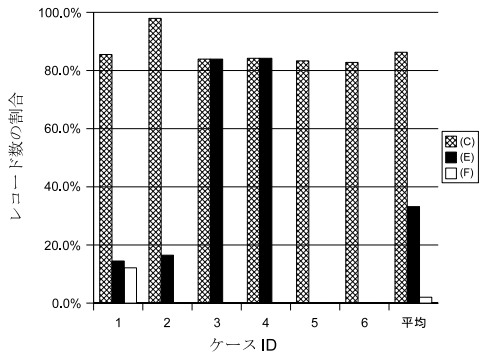


図 8 表 1 (B) を 100%とした error_log の各種レコード数の割合
Fig. 8 The ratio of remaining records of error_log when (B) in Table 1 is assumed to be 100%.

ぞれのケースのアクセスの内容としては，1–4，6 は 50 MB〜150 MB 程度の非常に大容量のファイルを 1 個または数個程度ダウンロードしたもの，5 はそれよりは小さい 3 MB〜5 MB 程度のファイルを短時間に複数個（21 個）ダウンロードしたものである．3，5，6 は実際の運用の中で観測されたアクセスであり，1，2，4 はそれを模して人為的にリクエストを生成した．

HTTP バイト数・パケット数の異常検出時には，HTTP アクセス数の異常検出時と異なり，レコードの件数は大きく増加せず，単一あるいは少数の特異なレコードがイベントに関連している．3.3.2 項で述べた特徴ベクトルに基づくクラスタリングの結果，(C) の段階で，特異でない一般的なレコードが多数削除されるが，3.4 節で述べた重要クラスタを判断するアルゴリズムはレコード数の少ない多数のクラスタを重要クラスタとして選択するため，(D) ではレコード数が大きく減っていない．また (E) の false positive となるレコードが多く残るという結果になっている．この結果，false positive を削除する作業負担が管理者に残される．しかしながら，元のレコード数と比較して 10%から 30%程度に削減できているため，作業負担の軽減には効果があるといえる．

以上より，HTTP バイト数・パケット数の異常検出時において，提案手法は高い精度で不必要なレコードのみを削減することができ，管理者がログを解析する負担を大幅に減らせることが確認された．

5.3 アルゴリズムのリアルタイム性の評価

図 10 に，クラスタリングの実行時間とレコード数の関係を実測した結果を示す．横軸はレコード数，縦軸はクラスタリングの実行時間 [sec] である．また，**図 11** に，クラスタリング終了時の総クラスタ数とレコード数の関係を実測した結果を示す．横軸はレコード数，縦軸はクラスタ数である．図 10，図 11 の A，B のカーブは，3.2 節で述べたように使用するフィールドをそれぞれ 4 次元，9 次元に限定した場合の結果，C のカーブはフィールドを限定せず，18 次元のベクトルを使用した場合の結果を示している．

クラスタリングの対象レコードは，実験に用いたサーバで得られた access_log から，任意にレコードを切り出したもので，特定のイベントや異常検出の状況にフォーカスしたものではない．これを用いて実行時間とクラスタ数を測定し，さらにレコードを追加し，再び実行時間とクラスタ数を測定し，またレコードを追加する，というサイクルを繰り返してデータを計測した．なお，B は 5.2.1 項で用いた特徴ベクトルであ

表 2 HTTP バイト数・パケット数の異常検出時の access_log，error_log のレコード数の削減結果
Table 2 The reduction in the number of records of access_log and error_log when anomalies related to the number of HTTP bytes and packets were detected.

対象ログ	access_log						error_log					
ケース ID	1	2	3	4	5	6	1	2	3	4	5	6
(A) クラスタリングに使用した数	6,530	5,056	2,516	6,850	2,352	9,975	570	353	190	407	106	897
(B) 異常検出時刻付近の数	149	115	212	185	123	241	3	4	13	18	2	30
(C) 重要クラスタに含まれる数	23	16	26	16	38	19	0	0	0	0	0	2
(D) 重要クラスタの代表値とした数	14	14	16	11	17	15	0	0	0	0	0	2
(E) (C) に含まれる必要のない数	16	15	25	15	17	17	0	0	0	0	0	2
(F) (C) に必要だが含まれない数	0	0	0	0	0	0	0	0	0	0	0	0

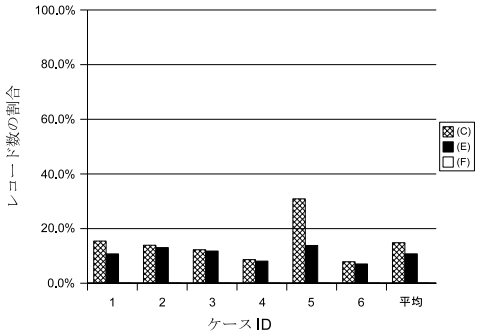


図 9 表 2 (B) を 100%とした access_log の各種レコード数の割合
Fig. 9 The ratio of remaining records of access_log when (B) in Table 2 is assumed to be 100%.

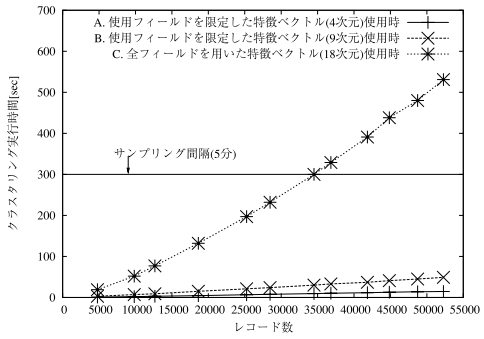


図 10 クラスタリングのレコード数と実行時間の関係
Fig. 10 Relation between a number of records and clustering time.

り，リクエストの特徴を重視して使用するフィールドを限定した．A は比較のため B からさらに 5 次元減らしたベクトルである．これらのフィールドの取捨選択方法については，現在のところ管理者の経験・知識に基づく判断が必要である．ログの学習や統計処理に基づき，自動で適切な特徴ベクトルを構成することが望ましいが，それは今後の検討課題とする．

提案手法は，レコード数 n，クラスタ数 m のとき，

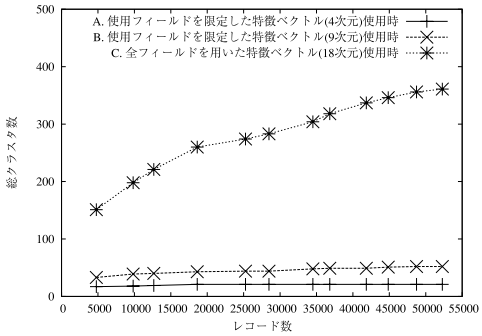


図 11 クラスタリングのレコード数とクラスタ数の関係
Fig. 11 Relation between a number of records and clusters.

n 個のレコードそれぞれが m 個のクラスタとの比較により自身の所属クラスタを決定するため，計算時間は $O(mn)$ になる．m は変数であるから，m が n に比べて十分小さい場合には $O(n)$ に近付き，逆に m と n が近くなると計算時間は $O(n^2)$ に近付く．これは図 10，図 11 の結果と一致する．図 10，図 11 の C. に示されるように，ログ内の全フィールドを用いて特徴ベクトルを構成した場合には，クラスタが細分化しやすく，m が増大するため，レコード数の増加とともに計算時間が急激に増加している．一方，図 10，図 11 の A.，B. に示されるように，必要なフィールドを絞り込み，特徴ベクトルを構成した場合には，レコード数の増加と比してクラスタ数の増加は小さく抑えられており，計算時間の増加も線形に近い形に抑えられ，早ければ数十秒，遅くともサンプリング間隔（5 分）以内でクラスタリングを終了できている．

以上より，3.3 節で述べた本提案手法の特徴である段階的な情報の絞り込みにより特徴ベクトルに用いるフィールドを絞り込み，m を一定値以下とすることで，多数のレコードからなるログに対しても，サンプリング間隔である 5 分以内で処理を終了でき，すなわちリアルタイム性を実現可能であることを確認した．

5.4 提案手法の汎用性・有効範囲に関する考察

5.4.1 利用環境に関する条件

提案手法はログの利用が前提であるため，提案手法を実行できる環境の条件として，ログを直接管理し，参照可能であることがあげられる．よって，コアネットワークではなくアクセスネットワークの，また上流よりは下流の環境での使用が想定される．対応可能なイベントの種類に関しては，イベントの内容がログに記録されることが条件となる．そのため，多様なイベントに対応するためには，ログに記録する項目を適切に設定しておくことが必要となる．

5.4.2 イベントの特徴に関する条件

第 1 に，ログへの影響の表れ方については，提案手法では，バイト数・パケット数の異常と比べ，アクセス数の異常の方がより正確に不要レコードを除外することができた．理由として，バイト数・パケット数の異常時には，レコード数ではなくレコードの内容に特徴が現れるものの，重要クラスタの決定方法がレコード数に基づいており，レコード内容の特徴をベクトル化の際の類似性以外に評価できなかったことがあげられる．そのため，重要クラスタの決定方法についてさらなる検討が必要である．

第 2 に，提案手法が対処可能な時間スケールの条件としては，提案手法は τ_1，τ_2 の時間に依存するパラメータを固定しており，その時間スケールで観測できるレコード数の特徴に依存して，重要クラスタを決定している．そのため，対象とした時間スケールに比べ，長い時間で特徴的なレコードを少しずつ出力するようなイベントや，短い時間でレコード数にピークが見られるイベントには対応できない．よって，扱うイベントの時間スケールを把握しておくことが必要となる．

第 3 に，複数の異常検出が短時間に重なる場合については，4.1 節で述べた異常発生情報のうち，(3) トラフィックデータの種類が似通っていたり，同じであったりする場合には，似通った特徴ベクトルを生成し，クラスタリングを行うため，結果として要約も似通ったものになり，両者を区別することが困難となる．したがって，そのような場合に提案手法を有効に用いるための条件として，異常発生情報のうちトラフィックデータの種類が大きく異なることがあげられる．

5.4.3 侵入検知システムとの比較

侵入検知システム（IDS）の分野では，検出された多数の false positive のレコードを除去し，必要な情報を抽出する作業が行われているが，提案手法はログの種類を限定せず，またイベントの識別の支援が目的であるため，それらとは用途や適用範囲が異なる．提案手法の IDS ログへの適用については，今後検討する．

6. おわりに

本論文では，リアルタイムでのイベントの「識別」の実現へ向けて，対応時間の短縮および作業負担の軽減を可能とする「ログ要約手法」を提案した．また，実験により，提案手法を用いてイベントの識別に有効なログ要約処理をリアルタイムに行うことが可能であることを示し，管理者の作業負担の軽減に大きく貢献できることを示した．

現在の提案手法では，クラスタリングの際の特徴ベクトルの選択，および重要クラスタの決定方法において，さらなる改善の余地がある．そこで今後の課題として，特徴ベクトルの選択指針に関する検討，および重要クラスタの決定方法に関して検討を行う予定である．

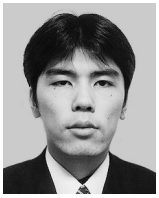
謝辞 本研究の一部は，科学研究費補助金（16300011）の援助を受けて実施した．

参考文献

1) 福士賢二，竹内春夫，倉内 博，森井昌克，辻井重男：不正アクセス被害解析支援システムの試作，情報処理学会研究報告 CSEC-18，pp.21–26 (2002).
2) 今野 将，吉村智志，羽鳥秀明，岩谷幸雄，阿部 亨，木下哲男：能動化された状態情報に基づくネットワーク管理支援方式，情報処理学会論文誌，Vol.46, No.2, pp.493–505 (2005).
3) Maxion, R.A. and Feather, F.E.: A Case Study of Ethernet Anomalies in a Distributed Computing Environment, *IEEE Trans. Reliability*, Vol.39, No.4, pp.433–443 (1990).
4) Barford, P., Kline, J., Plonka, D. and Ron, A.: A Signal Analysis of Network Traffic Anomalies, *Internet Measurement Workshop*, France (2002).
5) Hajji, H. and Far, B.H.: Integrated Network Operation Baselining and Adaptive Detection of Faults and Performance Problems, *IPSJ Journal*, Vol.44, No.2, pp.386–396 (2003).
6) Yamanishi, K. and Takeuchi, J.: On-Line Unsupervised Outlier Detection Using Finite Mixture with Discounting Learning Algorithms, *Data Mining and Knowledge Discovery*, Vol.8, No.3, pp.275–300 (2004).
7) 武藏泰雄，松葉龍一，杉谷賢一：DNS トラフィックとメールサーバのログ解析，情報処理学会研究報告 DPS-111/CSEC-20, pp.185–190 (2003).
8) Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals,

Soviet Physics - Doklady, Vol.10, No.8, pp.707–710 (1966).
9) Ali, R., Ghani, U. and Saeed, A.: Data Clustring and Its Applications. http://members.tripod.com/asim_saeed/paper.htm
10) The Apache HTTP Server Project. http://httpd.apache.org/
11) AWStats official web site. http://awstats.sourceforge.net/

(平成 18 年 7 月 4 日受付)
(平成 19 年 1 月 9 日採録)



長尾　真宏（学生会員）
1981 年生．2006 年東北大学大学院情報科学研究科博士前期課程修了．現在，同大学院同研究科博士後期課程在学中．ネットワーク管理手法に興味を持つ．

北形　　元（正会員）
1972 年生．2002 年東北大学大学院情報科学研究科博士後期課程修了．現在，東北大学電気通信研究所助手．博士（情報科学）．エージェント指向コンピューティング，エージェント型ネットワークミドルウェアに興味を持つ．電子情報通信学会会員．

菅沼　拓夫（正会員）
1966 年生．1997 年千葉工業大学大学院博士後期課程修了．1997 年東北大学電気通信研究所助手．2003 年同研究所助教授．やわらかいネットワーク，エージェント指向コンピューティング，ネットワークミドルウェア，共生コンピューティング等の研究開発に従事．The 8th JWCC Best Presentation Award，情報処理学会第 54 回全国大会大会奨励賞等受賞．博士（工学）．IEEE，電子情報通信学会各会員．

白鳥　則郎（フェロー）
1946 年生．1977 年東北大学大学院博士課程修了．1984 年同大学助教授（電気通信研究所）．1990 年同大学教授（工学部情報工学科）．1993 年同大学教授（電気通信研究所）．情報通信システム，人と IT 環境の共生の研究に従事．本会 25 周年記念論文賞．本会マルチメディア通信と分散処理研究会主査，本会理事，本会副会長，本会フェロー，IEEE フェロー，電子情報通信学会フェロー．