クラスタリングを用いたログ情報集約システム
Condensation of Log Data with Clustering

浦島 智*, 大津 三記男*, 安宅 彰隆*, 畑田稔*
Akira URASHIMA*, Mikio OTSU*, Shoryu ATAKA*, and Minoru HATADA*
富山県立大学*
Toyama Prefectural University

計算機システムの管理者とって，ログの監視は必要だが労力のかかる作業としてあまり実施されない傾向にある．本論文では，この労力を軽減するため，ログの情報をグループ化しランクを付けて提供することにより，管理者が見なければならないデータ量を削減するシステムを提案する．グループ化では単語を基準としたクラスタリングにより，先見的知識を与えることなく，類似のログをまとめることができる．試作した本システムを利用することにより，PC100台より得られた10日間のログデータ約29万行に対し，約240グループが生成され，その中で毎日約25グループ程度を見ることで，必要な監視が行えることを確かめた．

キーワード：ログ, クラスタリング, システム管理

Although it is recognized that monitoring and analysis of system log files are important job, administrators often leave them unwatched since it is time-consuming. To reduce the load of the administrators, we propose the log condensation system that put similar log messages together and ranks the generated groups. We employs the clustering method based on words, that can bundle similar log messages without prior knowledge. Our experimental system generates about 240 groups from about 290,000 lines of log data (collected from 100 PC's in 10 days), and enough monitoring can be performed by watching about top 25 groups everyday.

Keywords: log, clustering, system administration

1. はじめに

近年の情報化により，多数台で構成された計算機環境を安定して正常に運用することが，組織内においてますます重要になってきている．この安定した運用のために重要な作業として，ログの監視・調査がある．

ログとは，OSやサーバプログラム，アプリケーション等が出力する動作内容・結果などの情報を，時系列順に記録したものである．記録されたログを見ることによって，機器の動作状況を知り，障害や不正アクセス等を検出することができる．

管理者にとってログの調査・監視が有用であることは認識されている．例えば警察庁では，「不正アクセス行為の禁止等に関する法律」[1]第五条にある「防御措置」の主な項目の一つとしてログの取得と定期的な検査を挙げている[2]．

その一方で，ログの調査作業の実施度はそれほど高くない．比較的重要と思われるWWW・

*工学部電子情報工学科
〒939-0398 富山県射水郡小杉町黒河5180
Faculty of Engineering
〒939-0398 5180, Kurokawa, kosugi-machi, imizu-gun, Toyama, Japan
E-mail: a-urasim@pu-toyama.ac.jp
E-mail: r52005@st.pu-toyama.ac.jp
E-mail: ataka@pu-toyama.ac.jp
E-mail: hatada@pu-toyama.ac.jp

メールサーバ等のアクセスログに関しても，総務省の調査[3]によると，大企業でも64%程度，中小企業では34%程度しか監視や取得を行っていない．この原因として，ログから障害や不正アクセスの情報を得るためには，高度で専門的な知識が必要な一方で膨大な量を見なければならず，管理者の負担が大きいことが挙げられる．

そこで，本論文ではログ情報を監視する際の管理者負担を軽減するためのシステムを提案する．このシステムでは，類似のログをまとめるグループ化と，重要性に応じて順序をつけるランク付けという2種類の処理を行うことで，管理者が目で見なければならない情報量を削減する．

以下，2章ではログ監視に関する問題点を提起し，従来のログ監視・調査について述べる．3章ではログ情報集約システムを提案し，4章で試作システムに対する実験と評価を行う．最後に5章で本論文をまとめる．

2. ログの監視と調査

ログとは，OSやサーバプログラム，アプリケーション等が，作成者の意図や設定ファイルの指示などに従って動作内容・結果などを，エラーや警告，通知として順次出力したものである．出力されたログは時系列に添ってファイル等に記録され，その場に管理者がいない場合でも，後に記録されたログを見ることにより，障害や不正アクセスの試みなどを検知したり，異常発生時の原因推定に利用したりすることができる．

2.1 ログの種類

Linux等のUNIX系OSの多くには，syslogという汎用のログを処理する機能が備わっており，OSや多くのアプリケーションがこれを利用してログを出力する．syslogでは，記録される個別のログ情報を，レベルとファシリティという種別を利用して大まかに分別し，時刻・ホスト名と共にテキスト形式で記録する．また，syslogでは受け取った情報を，UDPパケットで他のホストに転送することができる．syslogで記録されたログの例を図1に示す．

```
Dec 10 12:12:12 s109 xinetd[763]: chargen disabled, removing
Dec 10 12:12:16 s109 lpd: lpd startup succeeded
Dec 10 12:21:48 s109 automount[707]: attempting to mount entry /nfs/home35
Dec 10 12:32:32 s245 sshd[1031]: session closed for user root.
```

図1　ログデータ例 (syslog)

WWW・メールサーバプログラムなどは，これらのOSに付随する汎用のログ機能とは別に，専用のログ記録を行うことが多い．例えば，WWWサーバであるapacheの標準の設定では，アクセスやエラーに関するログをそれぞれ独立した専用のファイルに記録する．

syslogのような，汎用のログ機能を利用する長所として，以下の点が挙げられる．

- OSやアプリケーション等の作成者が，簡単に実行の記録を残すことができる．
- システムの管理者が，ログを統一した方法で扱い，情報を集中して管理することができる．

現在の複雑なシステムは，多数の作成者によるソフトウェアで成り立っており，汎用のログ機能は必要な物である．その一方で，

- 汎用とするため，出力される形式を厳密に定めることができない．

という欠点がある．

2.2 ログ監視における問題点

先にも述べた通り，実際のログ監視作業の実施度はそれほど高くない．このログ監視が

行われにくい理由として，以下の3つの原因が考えられる．

1. 専門的で高度な知識が必要
   ログから意味のある情報を読み取るためには，そのログを出力したOSやアプリケーション自体の知識が必要な上，それらをどの様に設定したのか，あるいはシステム全体はどの様に構成されているのかなど，高度な知識が必要となる．

2. 膨大な量
   たとえば，富山県立大学の計算機センターでは，学生用クライアントPC 1台から，1日平均300行のログが出力される．計算機センター全体では約200台のクライアントPCがあり，これらだけでも1日平均60,000行にもなる．

3. 単調な作業
   前項で述べたように，ログ自体の量は膨大であるが，障害等のトラブルの発生は頻繁にあるものではない．また，クライアント200台が出すログは共通するものが多く，管理者は数100回も繰り返される異常の無いデータを，順次見ていかなければならない．

実際，システム全体に関する専門的で高度な知識をもつ人間は，管理者以外にそう多くはない．その管理者は，ユーザサポートやバージョンアップ等のメンテナンスも行わなければならないため，膨大な作業の必要となるログの監視は，優先順位の低い作業となってしまう．

2.3 これまでのログ監視・調査

ログ監視・調査に関しては，これまでも幾つかの研究があり，また実用として開発されたシステムなども多数ある．

例えば，swatch[4]，Logsurfer[5]などは，汎用のログを対象とし，事前に設定したパターンにマッチするログが出力された場合に，メールを送るなど一定のアクションを起こすというものである．また，Andrew[6]は，ログを解釈する状態機械の利用によって，ソフトウェアのテストを行う手法を提案している．これらの場合，多様なログに対応するためには，ログを出力するソフトウェアの知識を保持した上で，事前に多くの設定を行わなければならない．

また，WWWサーバ等が出力する専用のログを対象とするのならば，analog[7]，webalizer[8]などがある．これらは，WWWサーバの出力するログを解釈・統計処理し，月別や曜日別のアクセス量，何処からアクセスされたか，良くアクセスされるページなどを，レポートとして出力する．これらは，特定の分野に関しては，分かりやすくまとめられた情報を得ることができるが，汎用のログ情報には対応することができない．

高田らの見えログ[9]では，ログ情報を図として視覚化し，またログのテキストに対し色付けすることで，ログ情報の調査を行いやすくしている．また，各種のフィルタ機能を提供し，特定のログを抽出することを助けている．しかし，このシステムではログの自動的な分類は行われていない．

3. 提案方式

本論文では，管理者が見る必要のあるログのデータ量を削減するシステムを提案する．これにより，少ない作業量でログの監視ができるため，ログ調査の実施度を上げることができると期待している．

3.1 全体構成

このシステムでは，syslogのログを対象として，グループ化とランク付けの二つの処理を行う．図2に本システムの流れを示す．

まず，syslogの機能によりログを転送し，複数のクライアントやサーバからログ情報を収集する．収集したログは，ファイルに記録されると共に，本システムに入力される．

本システムに入力されたログ情報は，メッセージという単位で扱われる．メッセージとは，ログとして意味をなす最小単位で，syslogの記録においては1行に相当する．

メッセージは本システムに蓄積され，類似の内容をもつメッセージが一つのグループにまとめられる．このグループ化の処理により，後に管理者に提示する際，グループの代表となるメッセージのみを提示することができ，管理者の見るデータの量を削減することができる．このグループ化手法の詳細は後に述べる．

生成されたグループに対して，グループの重要度によってランク (順位) が付けられる．このランク付けにより，重要なグループが優先して提示され，管理者は重要性の低い情報を見ることなしに，障害対応などの必要な作業に取り掛かることができる．このランク付けの手法の詳細は後に述べる．

管理者は提示されたグループに対して，確認や詳細の検討，評価を行う．また，管理者からのフィードバックを受け付け，確認したグループを最下位にするなど，ランク付けに影響を与えることができる．

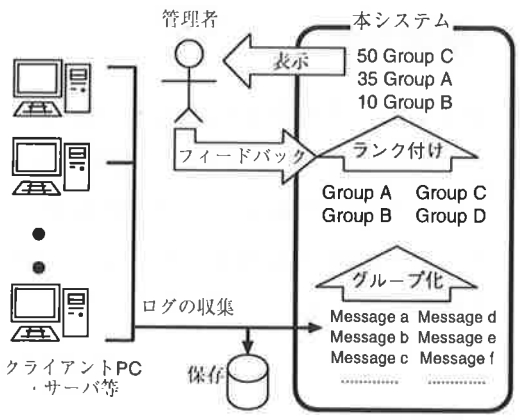

図 2　システムの流れ

3.2　グループ化手法

3.2.1　メッセージの単語への分解

まず，メッセージをRFC3164[10]の形式にしたがって，図3に示すように，TIMESTAMP, HOSTNAME, MSGの3つのフィールドに分割する．さらに，MSGフィールドを空白・記号を基準にして単語に分解する．

図 3　メッセージの分解

このMSGフィールドを分解して生成された単語の中には，そのメッセージが所属すべきグループの特徴を表さないと思われる単語が含まれている．例えば，図3の“1031”などである．そこで，グループ化における不要語として表1の単語を設定し，これらを除いたものを利用する．

表 1　グループ化における不要語

種別	例
数値	1031, 2484, ...
冠詞, be動詞, 接続詞, 前置詞	a, the, is, was, that, as, in, for...
日時を表す単語	fri, mon, oct, dec, ...
1文字のみの語	a, b, c, ...

3.2.2　クラスタリング

データ解析手法の中で，外的基準なしで分類を行う手法のことを，クラスター分析，もしくはクラスタリングと呼ぶ．

本システムでは，以下のような単純なクラスタリング手法を用いる．

1. 初期グループ生成
   新規に入力されたメッセージを，1 メッセージ 1 グループとして，新たに生成したグループに割り当てる．これらの新規グループは既存のグループと合わさり，一つのグループ群をなす．

2. 類似度の計算
   メッセージ x_i に含まれる単語の数を $|x_i|$，メッセージ x_j に含まれる単語の数を $|x_j|$，メッセージ x_i と x_j に共通して出現する単語の数を $|x_i \cap x_j|$ としたとき，メッセージ間の類似度 $s(x_i, x_j)$ を式 (1) のように定義し，全てのメッセージ間の類似度を計算する．

$$s(x_i, x_j) = \frac{|x_i \cap x_j|}{|x_i| + |x_j| - |x_i \cap x_j|} \quad (1)$$

この式 (1) による類似度の計算例を図 4 に示す．図において，ログメッセージ 1 とログメッセージ 2 は，接続先のサーバ名が異なるだけの同種のメッセージであるため，類似度が約 0.67 と高い．その一方，ログメッセージ 2 とログメッセージ 3 は，接続先サーバ名は同じであるが，その他の部分が全く異なるため，類似度は約 0.18 と低くなっている．

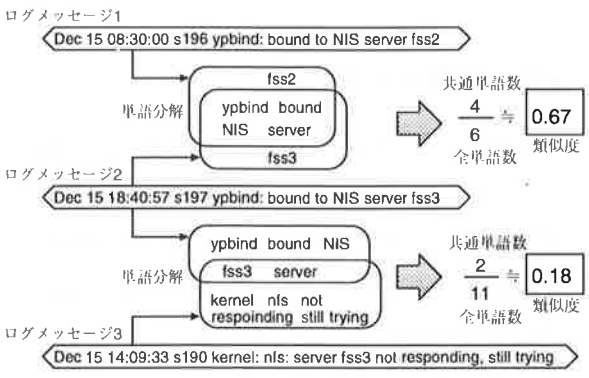

図 4　類似度の計算例

3. メッセージの移動
   メッセージ x_i を，グループ内の任意のメッセージ x に対し $s(x_i, x) > \alpha$ となるグループ G に移動する．ただし，α はグループ化パラメータであり，あらかじめ定めた値とする．もし，該当するグループが複数ある場合は，もっとも生成時刻の古いグループに移動する．該当するグループが存在しないときには，移動は行われない．

4. 空グループの削除
   メッセージの移動の結果，空となったグループを削除する．

3.3　ランク付け手法

グループのランク付けは，グループのスコアを計算し，スコアに添って整列することにより行う．グループのスコアは，以下の要素によるスコアを組み合わせることにより計算される．

1. 新規グループ
   新規グループには，管理者がこれまでに見たことのないメッセージが含まれている．そこで，新規グループには一定のスコア $score_n$ を与える．一度確認したグループは，このスコアを 0 にする．今回，新規グループには 40 点を与えることにした．

2. 重要単語の数
   “error”, “root” と言った単語を含むメッセージは，注目すべき情報を含むことが多い．そこで，あらかじめ設定した単語が何種類出現するかによって，それに比例したスコア $score_i$ を与える．今回設定した単語は，“error”, “warning”, “can’t”, “couldn’t”, “not”, “failed”, “failure”, “root” の 8 種類である．今回，スコア $score_i$ は (単語の種類数) × 10 点とした．

3. 管理者によるフィードバック
   グループが重要か否かには，管理者の価値判断に依存する部分がある．そこで，グループに対して評価をフィードバック

するため，管理者の操作により値を変更できるスコア $score_a$ をグループ毎に与える．今回は，重要と判断したグループに対しては一律に 35 点を与えた．

4. 未確認メッセージ
管理者が一度確認したグループについては，メッセージが新たに加わるまでは，再度確認する必要はない．そこで，未確認のメッセージを含むグループか否かを示すフラグ $flag_u$ を，グループに対して設定する．$flag_u$ は未確認なら 1，確認済なら 0 とする．

これらの要素によるスコアの計算式は以下の通りである．

$$score = (score_n + score_i + score_a) \times flag_u \quad (2)$$

フラグ $flag_u$ により，未確認のメッセージが存在しない場合は，最下位のランクに位置することになる．また $score_n$, $score_i$, $score_a$ の値は，複数の重要なグループに対して，事前の試行により大きな差がなくなるように決めたものである．

この他に，ログの出現頻度の変化を異常の指標として利用する方法も考えられるが，日周期，週周期変化に対して十分な補正を行うことが難しかったため，今回は利用しなかった．

4. 試作システムとその評価

4.1 試作システム

ログの処理に利用し，その有効性を評価するため，提案手法を実装したシステムを試作した．

図 5 に試作システムの画面例を示す．画面の 1 行には 1 グループが対応しており，各行は，アイコン，スコア，時刻，代表メッセージからなっている．グループに属するメッセージは，表示されている行をダブルクリックすることで表示される．

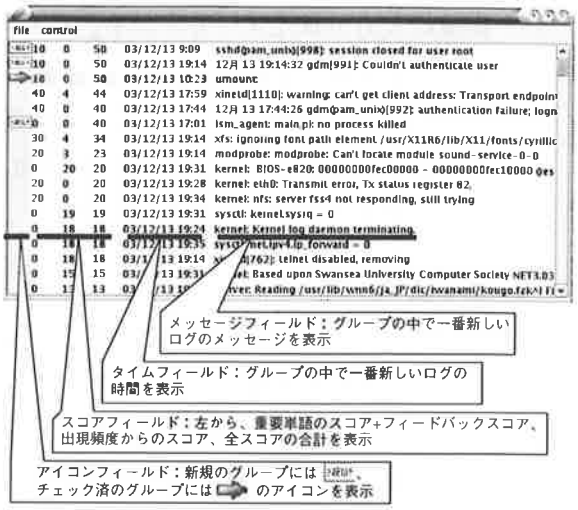

図 5　試作システムの画面

表 2　ログデータ量

月日	ログ行数
12 月 10 日 (火)	21,942 行
12 月 11 日 (水)	32,495 行
12 月 12 日 (木)	24,843 行
12 月 13 日 (金)	57,006 行
12 月 14 日 (土)	4,007 行
12 月 15 日 (日)	3,160 行
12 月 16 日 (月)	30,858 行
12 月 17 日 (火)	31,722 行
12 月 18 日 (水)	37,341 行
12 月 19 日 (木)	46,060 行
合計	289,434 行

4.2 実験データ

使用するログデータは，2002 年 12 月 10 日より 12 月 19 日までの 10 日間に，富山県立大学計算機センターのクライアント PC100 台より記録されたものである．クライアント PC の OS は，RedHat Linux7.2 である．表 2 に日単位でのログデータ量を示す．

もし，この 1 日平均 3 万行のログをそのまま監視するとなれば，単純に 1 秒に 1 行を見たととして，1 日 8 時間程度かかることになる．

表 3　グループ化の結果

α	総グループ数	重要かつ同一種類	重要かつ異なる種類が混在	非重要かつ同一種類	非重要かつ異なる種類が混在
0.0	1	0.0%	100.0%	0.0%	0.0%
0.1	76	7.9%	15.8%	48.7%	27.7%
0.2	132	6.8%	6.8%	71.2%	15.2%
0.3	196	9.2%	1.0%	83.7%	6.1%
0.4	223	10.3%	0.5%	88.3%	0.9%
0.5	236	9.8%	0.0%	89.4%	0.8%
0.6	277	9.0%	0.0%	90.3%	0.7%
0.7	312	8.0%	0.0%	92.0%	0.0%
0.8	343	7.3%	0.0%	92.7%	0.0%
0.9	371	7.8%	0.0%	92.2%	0.0%
1.0	381	8.1%	0.0%	91.9%	0.0%

4.3　グループ化パラメータの決定

まず，3.2.2 節におけるグループ化パラメータを決定するため，α の値を 0.0 から 1.0 まで，0.1 刻みで変化させて，実験データ全体をグループ化した．そして各グループを目視によって，重要であるか否か，同じ種類のメッセージのみで構成されているか否かの，計 4 種類に分類した．結果を表 3 に示す．

ここで重要であるとは，設定ミス，障害，不正アクセスの可能性をもつ，管理者が見るべきメッセージを含んでいると言うことである．また，同じ種類のメッセージとは，

- 同じプログラムから出力され，まとめて問題のないもの
- ブート時等の特定の同じタイミングで出力され，まとめて問題のないもの

の二つを指す．もし，重要なメッセージが他の種類のメッセージと混ざってグループ化されている場合，重要なメッセージを見逃す恐れがある．

表 3 中の，重要かつ異なる種類のメッセージが混在するグループの数と，総グループ数を，グループ化パラメータ α に対してプロットしたものを，図 6 に示す．この図を見ると，α が大きい方が異なる種類のメッセージが混ざり難くなることが分かる．その一方で，グループ数は α に対して単調に増加している．表 3 の同一種類かつ重要なグループの割合はあまり変化がないことから，α が大きくなると管理者の見るべきグループ数が増えることが分かる．

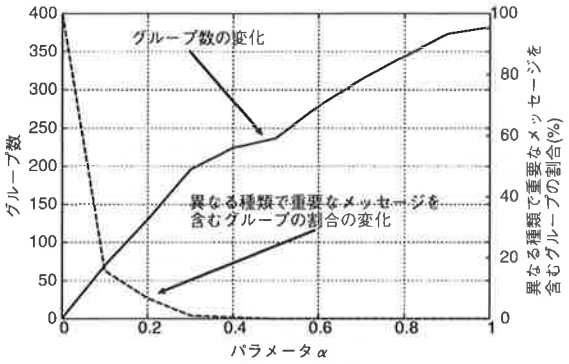

図 6　グループ化パラメータの影響

この結果から，重要かつ異なる種類のメッセージが混在するグループが 0 となる範囲で，総グループ数は少なくなる値として，グループ化パラメータ $\alpha = 0.5$ を採用することにする．

4.4　評価方法

通常の運用において，ログは継続して監視しなければならない．そこで，試作システムに実験データを 1 日分ずつ読み込ませ，1 日分の読み込みが終わった時点で表示される結果に対して，日単位での評価を行う．

表示された結果は管理者が確認し，新規に現れた重要なグループに対しては，フィードバックとして一定スコアを与える．この作業の結果は，翌日の分の結果に反映される．

表示されたグループが重要か否かは，各グループの詳細から目視によって判断した．この判断は全グループに対して行い，ここで重要と判断したグループを真の重要グループとする．評価は，重要グループ中の最下位の順位によって行う．これは，管理者がランク付けにしたがってログの調査を行う場合，最低限

この数のグループを確認する必要があると言うことである．

例えば，図7に示すように，実際には1, 2, 9番目の順位のグループが重要で，その他のグループが重要でなかった場合でも，管理者としては順位の高いものから順に調べていくことになる．そのため，この図の場合，全ての重要なグループを調べるためには，9個のグループを調べることになる．

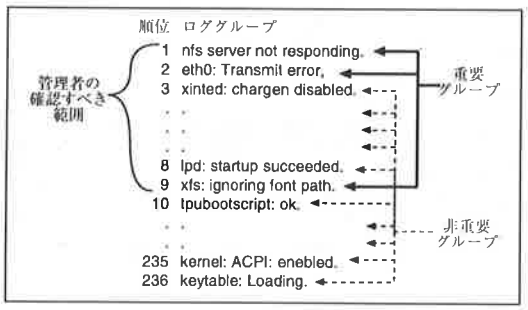

図7　評価例

4.5　実験結果

継続して使用する場合の評価とするため，1日目に関しては評価対象としていない．

2日目から10日目までの，総グループ数と確認する必要のあるグループ数，重要なグループ数を図8に示す．

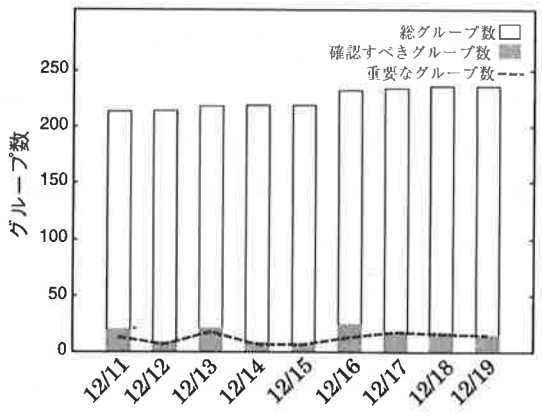

図8　管理者が確認べきグループ数

総グループ数は約210～240個で，それほど大きな変化はない．また，目視で判断した重要なグループは7-18個と日によって変動はあるが，割合で見ると全体の8%程度と少ないことが分かる．

これに対し，本システムで得られた確認すべきグループ数は最大で24個程度であり，ランク付けを行うことにより，重要なグループは上位約11%のグループを調査すれば確認できると言うことになる．

また，確認すべきグループ数と重要グループ数との間に，あまり大きな差がなく，このランク付け手法で上位にランクされたグループに，重要でないグループはそれほど多くないと言える．

評価実験の最終日における重要グループと非重要グループの例を，図9に示す．単一の原因によって生じる多数のログメッセージが，一つのグループとしてまとめられていることが分かる．

重要なグループ例1 (原因:特定マシンのネットワークインターフェースの異常)

```
Dec 13 13:46:11 s085 kernel: eth0: Transmit error, Tx status register 82.
Dec 13 13:46:13 s085 kernel: eth0: Transmit error, Tx status register 82.
<以下89回繰り返し(すべて同じPCより)>
```

重要なグループ例2 (原因: 一部マシンの設定ファイル記述ミス)

```
Dec 10 10:21:38 s216 xfs: ignoring font path element
 /usr/share/fonts/default/cmpsfont/pfb (unreadable)
Dec 10 10:21:51 s217 xfs: ignoring font path element
 /usr/X11R6/lib/X11/fonts/local (unreadable)
Dec 10 10:28:49 s215 xfs: ignoring font path element
 /usr/share/fonts/default/cmpsfont/pfb (unreadable)
<途中2212回 同様のメッセージ(複数のPCより)>
Dec 19 18:10:32 s090 xfs: ignoring font path element
 /usr/X11R6/lib/X11/fonts/cyrillic (unreadable)
```

非重要グループ例
(内部時計の誤差調整情報：誤差量によって2種類の調整方法が選択される)

```
Dec 10 10:21:38 s216 tpubootscripts: 10 Dec 10:21:38 ntpdate[957]:
 step time server 133.55.8.49 offset -1.263710 sec
Dec 10 10:21:53 s217 tpubootscripts: 10 Dec 10:21:53 ntpdate[971]:
 adjust time server 133.55.8.49 offset -0.294816 sec
<以下2種類のメッセージが合計 966回 (複数のPCより)>
```

(↵は紙面の都合上折り返した部分を示す)

図9　重要グループと非重要グループの例

5.　おわりに

本論文では，汎用のログに対し，グループ化とランク付けの二つの処理を行うことにより，管理者の見る必要のあるデータ量を削減するシステムを提案した．

さらに試作システムを作成し，実際に使用されているPCから得られた合計約29万行のログを用いて，評価を行った．その結果，グループ化により220～240グループ程度にまでまとめられ，さらにランク付けを行うことにより，それらの上位10%程度を調査することで，必要な監視が行えることが分かった．

このことから，本システムは管理者のログ監視の負荷を低減するものと言える．

今後の課題として，syslog以外の形式で出力されるログ情報の統合や，管理者に対するユーザインタフェースの検討と，実際の運用環境への適用などが挙げられる．

参考文献

(1) 総務省: “不正アクセス行為の禁止等に関する法律”, http://www.soumu.go.jp/joho_tsusin/top/access_law/law.html, (Aug. 1999)

(2) 警察庁: “不正アクセス行為の禁止等に関する法律の概要”, http://www.npa.go.jp/cyber/fusei_ac1/gaiyou.htm, (Jan. 2000)

(3) 総務省: “情報セキュリティ対策の状況調査”, http://www.soumu.go.jp/s-news/2002/pdf/020913_5_02.pdf, pp.46-49, (Sep. 2002).

(4) E. Todd Atkins: “swatch: the Simple WATCHdog”, http://swatch.sourceforge.net/

(5) Wolfgang Ley, Uwe Ellerman: “Logsurfer”, http://www.cert.dfn.de/eng/logsurf/

(6) J. H. Andrews: “Testing using log file analysis: tools, methods, and issues”, Proc. 13 th IEEE International Conference on Automated Software Engineering, pp. 157-166. (Oct. 1998).

(7) “Analog”, http://www.analog.cx/

(8) “The Webalizer”, http://www.mrunix.net/webalizer/

(9) 高田 哲司, 小池 英樹: “見えログ: 情報視覚化とテキストマイニングを用いたログ情報ブラウザ”, 情報処理学会論文誌, Vol.41, No.12, pp.3265-3275, (Aug. 2000).

(10) C. Lonvick: “The BSD syslog Protocol”, RFC3164, (Aug. 2001).

(2004 年 10 月 18 日原稿受付)
(2005 年 3 月 1 日採録決定)

著者略歴

浦島 智 1993 年京都大学工学部電気工学科卒業. 1999 年同大学大学院工学研究科電気工学博士後期課程修了. 1999 年富山県立大学工学部電子情報工学科助手 (計算機センター兼務). 博士 (工学).

大津 三記男 2001 年富山県立大学工学部電子情報工学科卒業. 2003 年富山県立大学大学院工学研究科電子情報工学専攻博士前期課程修了.

安宅 彰隆 1979 年東京大学理学部化学科卒業. 1984 年同大学大学院理学系研究科化学専攻博士課程修了. 同年富山県立技術短期大学講師. 1990 年富山県立大学工学部電子情報工学科助教授, 現在に至る. 理学博士.

畑田 稔 1972 年京都大学大学院工学研究科電気工学博士課程単位取得退学. 1972 年 (株) 日立製作所入社. 1998 年富山県立大学工学部教授. 2000 年富山県立大学計算機センター所長, 現在に至る. 工学博士.