DBSJ Journal, Vol.9, No.2
November 2010

リアルタイムバースト検出手法の提案

A Proposal for a Real-Time Burst Detection Method

蝦名 亮平♥　中村 健二♦
小柳 滋♠

Ryohei EBINA　Kenji NAKAMURA
Shigeru OYANAGI

複数のウィンドウサイズに渡ったバーストを正確かつリアルタイムに検出することは，データストリームの解析において有用である．多くのバースト検出手法が提案されているが，本研究では不要なデータ更新を行わず，バーストをリアルタイムに検出する手法を提案する．これは，イベント発生時に解析を行い，直前の状態よりも到着頻度が急激に高くなっている期間を発見することで実現する．また，イベントが集中発生しても一定期間内のデータを圧縮して保持・解析することにより，計算量を抑える．実験により，提案手法の解析結果の有効性と，大量のデータをリアルタイムに処理可能であることを示す．

Real-Time burst detection over multiple window size is useful for analyzing data streams. A number of burst detection methods have been proposed, however they are not effective for real-time detection. This article proposes a new burst detection method that reduces computation by avoiding redundant data updates. It analyses events on its occurrence, and detects the period where arrival frequency rises rapidly to the previous period. In addition, it reduces computation by suppressing data within a certain period even in the case of emergent increase of events. The effectiveness of the proposed method is evaluated by experiments with practical data.

1. はじめに

ネットワーク技術の発展により大規模なデータストリームが登場し，これらの異常な状態を機械的に検出する方法が必要となった．データストリームとは高速に次々と流れてくる大量のデータを指し，例としてオンラインニュース，ブログ，掲示板，通信記録，センサーデータなどが挙げられる．このデータストリームの異常な状態や急激な変化を素早く知る方法として，イベントの異常な集中発生をリアルタイムに検出する手法が注目されている．例えば，オンラインニュースの急激な変化をリアルタイムに検出することができれば，注目されている事柄などを膨大な情報の中から素早く把握することが可能である．また，通信記録をリアルタイムに監視することによって，DoS攻撃やアクセス状況の変化などを検出可能となり，それらのアクセスに対し早急な対応が可能となる．

データストリームにおけるイベントの異常な集中発生はバーストと呼ばれ，バーストを検出する様々な研究[1]，[2]，[3]，[4]が行われている．また，バースト検出手法を応用した研究[5]，[6]，[7] も行われており，現実世界の実データにおいてバースト検出手法の重要性が高まっている．既存のバースト検出手法では，ドキュメントストリームのバーストを解析する手法[1]や，一定期間毎のイベント発生数を観測したデータからバーストを解析する手法[2]，[3]，[4]がある．前者の手法は，あるイベントの発生に対して即座にバーストを解析するには不向きである．後者の手法は，複数のウィンドウサイズに渡ったバーストをリアルタイムに検出することは可能であるが，後述のとおり条件によってデータの不要な更新が多くなり計算量が膨大になる場合がある．

本論文ではデータの不要な更新を防ぎ，データストリームのバーストをリアルタイムに検出する手法を提案する．これは既存手法[2]，[3]，[4]の一定期間毎にイベント発生数を監視する方法と異なり，イベントが発生する毎にバーストを解析することによってリアルタイムに検出を行う手法である．本手法ではイベントの集中発生時に，保持するデータを圧縮することで計算量を抑え，リアルタイム性の高いバースト解析を実現している．

本論文の構成は次のとおりである．2章では関連研究について述べる．3章ではリアルタイムにバーストを検出する手法を提案する．4章では評価実験を行い，提案手法の解析結果の有効性と，大量のデータをリアルタイムに処理可能であるかを検証する．最後に5章で今後の課題を述べる．

2. 関連研究

Kleinbergの手法[1]は，あるトピックに関するドキュメントストリームにおいて，頻繁にドキュメントが到着する部分がバーストしている度合いが高いと考えて，バーストを解析する．Kleinbergの手法の利点は，各トピックにおけるバーストの発生期間と，バースト度合いを特定することができる点である．しかし，あるイベントの発生に対して即座に解析することを前提としていないため，リアルタイムなバースト検出には不向きである．

Zhangらの手法[2]やZhuらの手法[3]，[4]はバーストに関する閾値を定め，一定期間毎に複数のウィンドウサイズのイベント発生数を監視する方法である．これらの手法は監視する期間の間隔を短くすることでリアルタイムに近いバースト検出が可能である．しかし，そのためには一定期間毎のイベントの発生数を保持する必要があり，長期間イベントが発生していなくても一定期間ごとに保持しているデータを更新する必要があり，無駄な計算が行われる．このため，ドキュメントストリームに含まれる全ての名詞など，出現頻度にばらつきがある膨大な種類のイベントをリアルタイムに監視する場合，監視対象とするイベントの数に比例して計算時間が増加し，不向きである．次章ではこれらの問題に対応したバースト検出手法を提案する．

3. 提案手法

♥ 学生会員　立命館大学大学院理工学研究科博士前期課程
cs001066@ed.ritsumei.ac.jp
♦ 正会員　立命館大学情報理工学部
k-nakamu@fc.ritsumei.ac.jp
♠ 正会員　立命館大学情報理工学部
oyanagi@cs.ritsumei.ac.jp

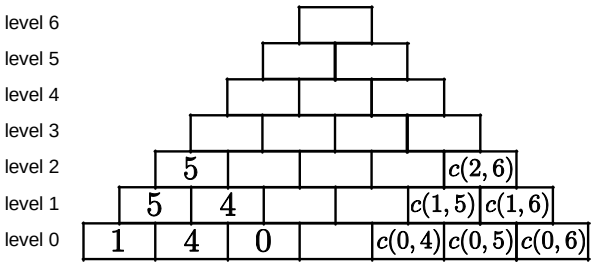


図 1　Aggregation Pyramid の例

Fig.1　Example of Aggregation Pyramid

本章では，ウィンドウサイズを固定せず，リアルタイムにバーストを検出する手法を提案する．

3.1 概要

本手法の目的は，データストリーム中のバーストの検出をリアルタイムに行うことである．本手法では，既存手法[2]，[3]，[4]のように一定期間毎にバーストを解析するのではなく，イベント発生毎にバーストを解析する．このことにより，イベントが発生していない期間の無駄な計算を削減でき，リアルタイムなバースト検出が可能となる．また，イベント発生時に毎回バーストを解析すると，イベントが集中発生したときに膨大な計算回数となりリアルタイム処理の弊害となる．そのため，本手法では一定期間内に複数のイベントが発生した場合，それらを一つのデータに圧縮してデータを蓄積する．さらに，イベントの発生頻度が低い場合も直前までの状態より急激に高くなっていれば，イベントの異常な集中発生と判断することができるが，既存手法では見逃されることが多い．このような状態を発見するために，期間が重複していない直前の状態よりも，到着間隔が急激に小さくなっている期間をバーストと定義して解析する．

3.2 データ構造

本手法のデータ構造は，Zhang らの手法[2]で提案された aggregation pyramid を参考としている．本節では最初に aggregation pyramid について述べ，次に提案手法のデータ構造について述べる．

3.2.1 Aggregation Pyramid

Aggregation pyramid とは，図 1 のようなデータ表現である．ここでは，最新のデータがセルに格納されて各レベルの右側に追加され，追加された分だけ各レベルの左側のセルが破棄される．N個のレベルから成り，時間tに終了するレベルhのセルを$c(h,t)$と定義すると，以下のルールにより構成される．

- レベル0はN個のセルを持つ．これらは実際のデータに対応する．
- レベルhは$N-h$個のセルを持つ．$c(h,t)$が持つ値は$c(0,t-h)$から$c(0,t)$が持つ値を利用して計算される．
  - レベル1は$N-1$個のセルを持つ．1番目のセルはレベル0の1番目と2番目のセルが持つデータを利用して計算する．レベル1の2番目のセルはレベル0の2番目と3番目のセルが持つデータを利用して計算する．3番目以降も同様である．
  - 最も高いレベル，つまりレベル$N-1$は1つのセルを持つ．これはレベル0の全てのセルが持つ値を利用して計算される．

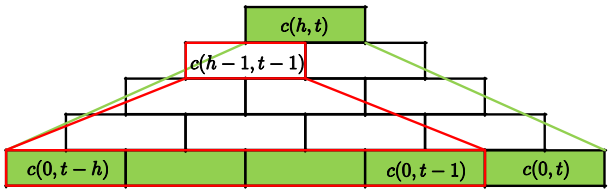


図 2　Aggregation Pyramid の特徴

Fig.2　Feature of Aggregation Pyramid

$c(h,t)$が持つ値の計算に必要なセルは$c(0,t-h)$から$c(0,t)$までのセルであるが，図 2 で示すとおり，aggregation pyramid では，$c(0,t-h)$から$c(0,t-1)$までのセルの値は$c(h-1,t-1)$に集約されるため，$c(h,t)$が持つ値は$c(h-1,t-1)$と$c(0,t)$を利用して計算が可能である．あるイベントの発生数について解析する場合，レベル0の各セルは観測された値（一定期間内のイベント発生数，期間の開始時間，期間の終了時間，期間の長さ）を持つ．$c(h,t)$が持つ値は$c(0,t-h)$から$c(0,t)$が持つイベント発生数の合計，開始時間の最小値，終了時間の最大値，期間の長さの合計となり，それをもとに各レベルのセルが生成される．イベント発生数が閾値を超えていればバーストと判定する．

Zhang らの手法[2]ではこのデータ表現の中で表されているセルをすべて生成・保持するのではなく，計算量が最小になるように効率的に解析するためのデータ構造を提案している．また，ドキュメントストリーム中の複数の単語について解析する場合，監視する単語と同数のデータ構造が必要となり，処理時間は監視単語数に依存して増加する．

3.2.2 提案手法のデータ構造

本手法は aggregation pyramid の性質を応用した新たな手法である．Zhang らの手法では，各セルはイベント発生数，開始時間，終了時間，期間の長さを持つのに対して，本手法では各セルは合計到着間隔$gaps$，到着時間$arrt$，間隔個数$gapn$の3つのデータを持つ．

一連の$n+1$個のイベントに対して，各イベントのn個の発生間隔を$x=(x_1,x_2,\ldots,x_n)$と表す．また，大量のイベントが集中発生するとセルの更新回数が膨大となるため，解析するウィンドウサイズの最小値W_{min}を定義する．W_{min}は，指定された値よりも短い期間内の情報を一つのセルに圧縮し，更新回数を一定以下に抑えるための値である．

新しくイベントが発生すると以下のルールに従ってデータ構造を構築する．

1. レベル0のセルの生成方法
   - $x_i \geq W_{min}$の場合，$c(0,t).gaps$にはx_i，$c(0,t).arrt$には$i+1$番目のイベントが発生した時間，$c(0,t).gapn$には 1 を格納する．iとtを1増やす．
   - $x_i < W_{min}$の場合，$c(0,t).arrt$には$c(0,t-1).arrt + W_{min}$を格納する．$c(0,t).gapn$には$c(0,t-1).arrt$から$c(0,t).arrt$直前までの期間に発生したイベントの発生回数を格納する．このとき，$c(0,t).arrt$にイベントが発生していなければ$c(0,t).gapn$を1減らす．$c(0,t).gaps$にはW_{min}を格納する．$i=i+c(0,t).gapn$，tを1増やす．i更新後のx_iは$c(0,t).arrt$から次のイベント発生までの

経過時間に書き換えるが，$c(0,t).arrt$で複数のイベントが発生していれば$x_i = 0$とする．

2. レベル1以上のセルの生成方法．
   - $c(h,t).gaps = c(h-1,t-1).gaps + c(0,t).gaps$
   - $c(h,t).arrt = c(0,t).arrt$
   - $c(h,t).gapn = c(h-1,t-1).gapn + c(0,t).gapn$

こうすることによって，データの更新がイベント発生時のみとなり，イベントが発生していないときの無駄な計算を削減できる．また，期間W_{min}内に複数のイベントが発生した場合はそれらの情報が1つのセルに圧縮されるため，膨大な量のイベントが集中発生したときにデータの更新回数を一定に制限できる．

本手法のデータ構造の構築例を以下に示す．$W_{min} = 1$と設定し，一つのセルに格納されるデータを$c(h,t) = (gaps, arrt, gapn)$と表現する．イベント発生時間列$\{0,1,6,6,6,6,9,9,10\}$を得たとき，発生間隔列は$\{1,5,0,0,0,0,3,0,1\}$となる．セルには到着間隔を基準にデータが格納されるが，図 3 のように補正されてから圧縮されてセルに格納される．レベル0のセルは$c(0,0) = (1,1,1)$，$c(0,1) = (5,6,1)$，$c(0,2) = (1,7,4)$，$c(0,3) = (2,9,1)$，$c(0,4) = (1,10,2)$の順に生成される．期間W_{min}の間にイベントが複数回発生しているため，時間6から時間7までの期間に4つの同じ長さの到着間隔が存在すると仮定して，その期間に対応する情報は圧縮されて一つのセルに格納される．これは，$W_{min} = 1$よりも短い期間内での大量のイベントの発生や，到着頻度の変化によって生成されるセルの数が増加し過ぎることを防ぐ．また，実際に時間7にイベントが発生していなくても，$arrt$は7に補正され，次のイベント発生時間9までの到着間隔が実際には3であるが，2として計算される．しかし，解析対象とするウィンドウサイズの最小値W_{min}を設定しているため，誤差がW_{min}以下であれば問題ないと考える．このように，到着間隔が長ければイベントが発生する度にセルが生成され，到着間隔が短くなると許容誤差の範囲で情報が補正され，圧縮される．

図 4 において，実線で書かれたセルは，塗りつぶされたセルが新しく生成されるときに保持していなければならないセルである．$c(h,t)$の生成後に$c(h-1,t-1)$を破棄する．対応するレベル0のセルが重複していない最新の2つのセルである$c(h,t)$と$c(N-1,t-1-h)$を後に示す方法で比較してバーストを判定する．

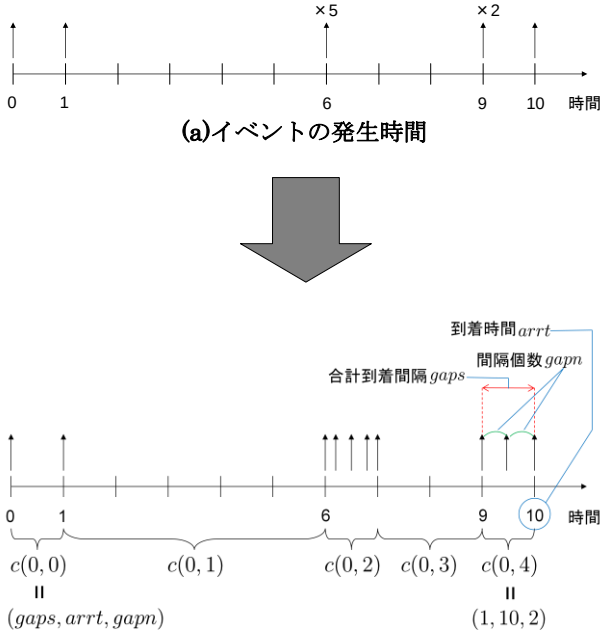
3.3 バースト判定方法

本手法では，到着間隔が重複していない直前の状態よりも急激に小さくなっている期間をバーストが発生している期間と定義する．各セルを同じ条件で比較するために1つのセル内の到着間隔の1つあたりの平均値を求める平均到着間隔関数を式(1)のように定義し，$avg(c(h,t))$と$avg(c(N-1,t-1-h))$を比較する．

$$avg(c(h,t)) = \frac{c(h,t).gaps}{c(h,t).gapn} \qquad (1)$$

バーストを判定するパラメータ$\beta(0 < \beta < 1)$ を用いて，条件(2)を満たすとき，バーストが発生していると定義する．

$$avg(c(h,t)) \le \beta \times avg(c(N-1,t-1-h)) \qquad (2)$$



(a)イベントの発生時間

(b)補正後のイベントの発生時間

図 3　レベル 0 のセルの生成方法の例($W_{min} = 1$)
Fig.3 Example of generating cells at level 0 ($W_{min} = 1$)

また，バーストと判定できるイベント発生間隔の最低個数A_{min}を設定し，$c(h,t).gapn < A_{min}$となるときは$c(h,t)$のバースト判定を行わない．既存手法では，観測するデータに合わせてパラメータの大きな変更が必要な場合が多いが，本手法はこのように直前状態と比較して相対的にバーストを判定することによって，異なるデータにおいても同じ条件で解析可能である．

データ構造内に十分なデータが蓄積され$c(N-1,t-1-h)$が存在する場合のバーストを検出するためのアルゴリズムを Algorithm1 に示す．変数tはレベル0のセルが何回目に生成されたかを表すため，レベル0の一つのセルに対応するデータを得る度に Algorithm1 の 2 行目から実行されることになる．バーストを監視し始めた初期段階において，保持しているデータが少なく，新たに生成した$c(h,t)$に対して$c(N-1,t-1-h)$が存在しない場合は，期間が重複していないセルの中で最新かつ最もレベルが高いセルと比較してバーストを判定する．今後使用することがないセルは順次破棄していく．

4. 実験

本実験では，提案手法が新たなリアルタイムバースト検出手法として有用であるかを検証するため，3 つの実験を行う．1 つ目の実験では，提案手法で抽出したバースト期間が正確であるかを検証するため，Kleinberg の手法[1]との比較実験を行う．2 つ目の実験では，Zhang らの手法[2]の条件によってはデータの不要な更新が多くなる点を改善できているかを検証するため，2 つの手法をバースト監視に利用した場合の CPU 時間を計測し，処理時間の観点から比較する．3 つ目の実験では，提案手法でリアルタイム検出を行う場合，

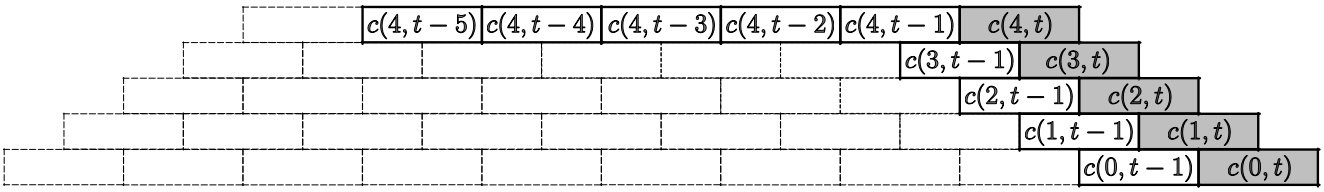


図 4　データの更新時に使用するセル
Fig.4　Utilization cells for update process

Algorithm1　提案手法のバースト検出アルゴリズム
Burst detection algorithm of our method

```
for t do
  h = 0
  while h < N do
    c(h,t)を生成
    c(h−1,t−1)を破棄
    if c(h,t).gapn >= Amin かつ
    avg(c(h,t)) ≤ β × avg(c(N−1,t−1−h)) then
      c(h,t) がバーストしていると判定
    end if
    if h = N−1 then
      c(N−1,t−1−h)を破棄
    end if
    c(h,t) をデータ構造に追加
    h++
  end while
end for
```

どの程度の規模のイベント数まで対応可能であるかを検証するため，単位時間当たりでの監視イベント数の上限を計測する．本実験では，監視イベントを単語の出現とする．

4.1 実験環境

実験環境を表 1 に示す．すべての実験において提案手法のパラメータは予備実験の結果に基づき，経験的に$N = 50$, $\beta = 0.4$, $W_{min} = 1$, $A_{min} = 15$を採用した．本実験では，提案手法が現実世界での単語の出現傾向に基づいた監視において有効であるかを検証するため，新聞記事データを利用する．新聞記事データは「CD-毎日新聞データ集 2006 年版」より，2006 年発行分の記事の本文キーワードと日付情報を用いる．

4.2 バースト検出の精度の比較実験

本実験では，バースト検出の精度を検証するために，単語「サッカー」のバースト期間を提案手法と Kleinberg の手法にて判定した結果を比較する．図 5 は単語「サッカー」を含む新聞記事の記事数の変化と，提案手法によりバーストが検出された期間を表す．横軸は 2006 年 1 月 1 日からの経過日数，縦軸は特定単語が含まれた記事数を表す．図 6 は同様のデータを Kleinberg の手法によって解析した結果である．

表 1　実験環境
Table 1　Experiment environment

OS	Ubuntu10.04
開発言語	C++
CPU	Intel Core2 Quad Q9550 @2.83GHz
メモリ	2GB

図 5，図 6 の結果より，バースト期間の開始時期は似ているが，バースト期間の終了時期が異なることがわかる．この違いは，Kleinberg の手法では記事全体の出現傾向に基づきバースト期間を判定しているのに対して，提案手法は直前の出現傾向に基づきバースト期間を判定しているためである．提案手法がバーストと判定していない経過日数 200 日前後の記事数を確認すると，単語「サッカー」の発生数は右肩下がりとなっており，実際にはバーストしていないと判断できる．この結果から提案手法はバースト期間を正しく判定しており，バースト検出手法として有効であることがわかる．

4.3 リアルタイム監視の処理負荷の比較実験

本実験では, Zhang らの手法と提案手法を用いて実験データのバーストを監視した場合の処理時間を CPU 時間にて比較する．ここで，監視する間隔は1日単位とする．また，監視する単語は，あらかじめ計算した DF（Document Frequency）値を用いて値が高い順に任意の数だけ選択する．

図 7 は実験データを Zhang らの手法と提案手法で解析するに当たって，監視単語数を変化させた場合の監視単語数と CPU 時間の関係である．横軸は監視する単語数，縦軸は365日監視した場合の CPU 時間を表す. CPU 時間は365日分を1日単位で監視した際に利用した CPU 時間を積算した値である．図 8 は実験データを Zhang らの手法と提案手法で監視単語数を100,000として監視した場合の経過日数と累積 CPU 時間の関係である. 監視する間隔は1日単位と0.5日単位である．新聞記事データには時間情報がないため，0.5日単位のデータはこの実験用に単語の発生数を1日あたりの発生数から均等に割り振って使用する．

図 7 の結果を確認すると，提案手法では監視単語数が10,000語を越えた段階で CPU 時間の傾きがゆるやかとなり，20,000語以降はほぼ横ばいとなっていることがわかる．それに対して，Zhang らの手法では，監視単語数が増加するにつれて CPU 時間が同様の傾きで増加していることがわかる．これらの結果から，提案手法は常時頻出する単語ではなく，季節や時期に応じて出現傾向の異なる単語や，突発的なイベントに対応して出現する単語のバーストを監視する際に非常に有効であることがわかる．この特徴は実際のバースト監視においても有効であり，例えば地震や台風，ゲリラ豪雨な

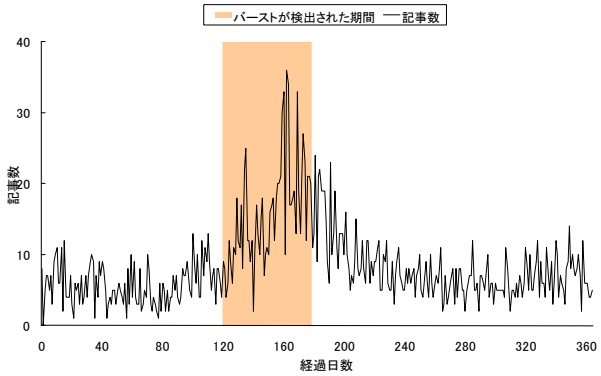


図 5　提案手法による新聞記事データの解析結果
Fig.5　Result of burst detection by our method

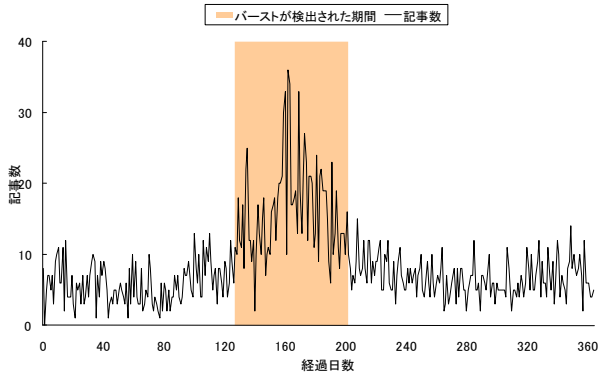


図 6　Kleinberg の手法による新聞記事データの解析結果
Fig. 6　Result of burst detection by Kleinberg's method

どの自然災害や大規模な事故に関する単語など，その事柄が発生した際に話題が急増する単語を監視する場合に適していると考えられる．

図 8 の結果を確認すると，Zhang らの手法と比較して提案手法は累積 CPU 時間が短いことがわかる．また，監視間隔と累積時間の関係を分析すると，Zhang らの手法では監視間隔を1日から0.5日と1/2にすることで，累積 CPU 時間は反比例して約 2 倍になっていることがわかる．これは Zhang らの手法が監視間隔に依存してバースト検出処理を実行するためである．それに対して，提案手法は監視間隔を1/2にした場合に，累積 CPU 時間は約1.5倍しか増加していないことがわかる．これは，提案手法が発生したイベントに対してバースト検出処理を実行するためである．この結果から，提案手法は監視間隔を短くした場合に有効であることがわかる．この特徴は，インターネット上の時々刻々と変化する情報（オンラインニュースやブログ，掲示板の投稿など）を対象としたリアルタイムなバースト検出に適していると考えられる．

4.4　解析時間の検証

本実験では，複数のデータ構造を1回ずつ更新した場合の CPU 時間を計測し，リアルタイム性を検証する．保持している全てのデータ構造を1回ずつ更新するために，全データ構造が監視するイベントで同時にバーストが発生するようなデータを人工的に作成し，それを解析する．図 9 は同時に

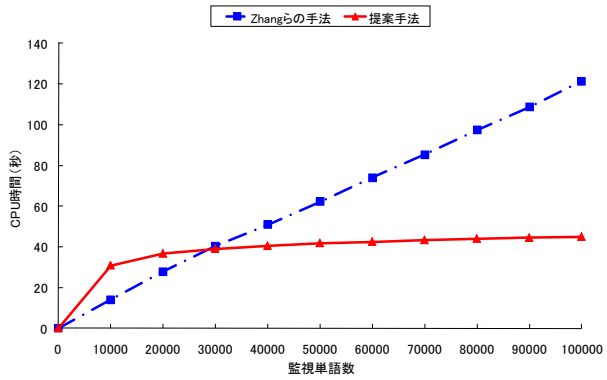


図 7　新聞記事の監視における監視単語数と CPU 時間の関係
Fig.7　Performance on number of monitoring words versus CPU time

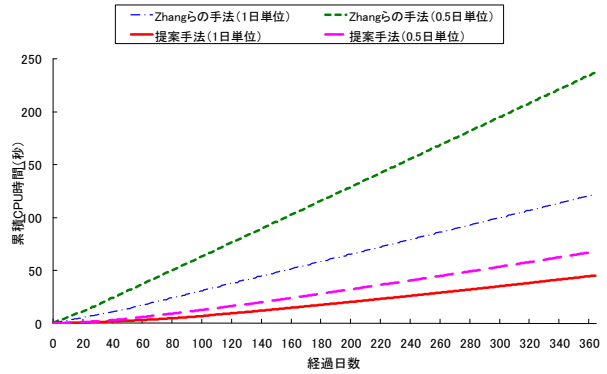


図 8　新聞記事の監視における経過日数と累積 CPU 時間の関係
Fig.8　Performance on elapsed days versus accumulated CPU time

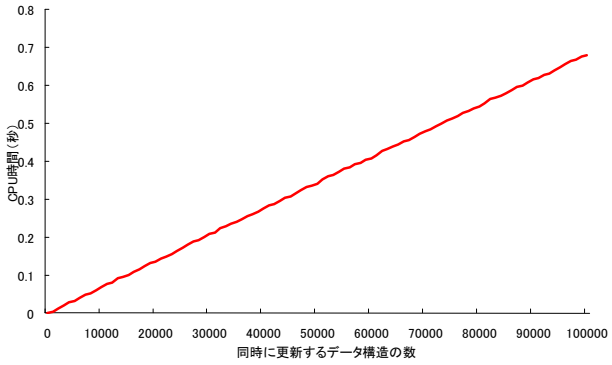


図 9　データ構造の数と更新に必要な CPU 時間の関係
Fig.9　Performance on number of data structures versus CPU time

更新するデータ構造の数と更新に必要な CPU 時間の関係である．横軸は同時に更新するデータ構造の数，縦軸は更新に必要な CPU 時間を表す．CPU 時間は任意の数のデータ構造をすべて1回ずつ更新した際に利用した CPU 時間の積算した値である．

実験結果より，100,000個のデータ構造をそれぞれ1回ずつ更新するために必要なCPU時間の合計は0.7秒以下であることがわかる．このように，膨大な種類のイベントが同時に発生した場合においても，短時間で解析可能であるため，実際のリアルタイム監視に利用可能な汎用性を確保していることがわかる．

5. おわりに

本論文では，直前の状態よりも到着頻度が急激に高くなっている期間を発見することにより，バーストを検出する手法を提案した．これは，一定期間毎ではなくイベント発生時にバーストの解析を行うことにより，イベントが発生していない期間の無駄なデータの更新を防ぎ，リアルタイムな解析を可能とする．さらに，イベントが集中発生している期間のデータを圧縮して保持するため，膨大な量のイベントが集中発生したときにデータの更新回数を一定に制限することが可能である．

実験により，実際のドキュメントストリームのような出題傾向に偏りがあるデータをリアルタイムに監視する場合，提案手法が有効であることを確認できた．また，大量のデータをリアルタイムに処理可能であることが確認できた．今後の課題として，より多くのデータストリームのバーストをリアルタイムに監視するために，使用メモリ量の削減等が挙げられる．

[文献]

[1] Kleinberg, J.: Bursty and Hierarchical Structure in Streams, Data Mining and Knowledge Discovery, Vol.7, No.4, pp.373-397 (2003).

[2] Zhang, X. and Shasha, D.: Better Burst Detection, ICDE'06: Proceedings of the 22nd International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society, pp.146-149 (2006).

[3] Zhu, Y. and Shasha, D.: Efficient Elastic Burst Detection in Data Streams, KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM, pp.336-345 (2003).

[4] Shasha, D. and Zhu, Y.: High Performance Discovery In Time Series. Techniques And Case Studies (Monographs in Computer Science), SpringerVerlag (2004).

[5] Kumar, R., Novak, J., Raghavan, P. and Tomkins, A.: On the Bursty Evolution of Blogspace, WWW '03: Proceedings of the 12th International Conference on World Wide Web, New York, NY, USA, ACM, pp.568-576 (2003).

[6] He, Q., Chang, K. and Lim, E.-P.: Using Burstiness to Improve Clustering of Topics in News Streams, ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, Washington, DC, USA, IEEE Computer Society, pp.493-498 (2007).

[7] Platakis, M., Kotsakos, D. and Gunopulos, D.: Discovering Hot Topics in the Blogosphere, Proceedings of the 2nd Panhellenic Scientific Student Conference on Informatics, Related Technologies and Applications, Samos, Greece, pp.122-132 (2008).

蝦名亮平 Ryohei EBINA

立命館大学理工学研究科博士前期課程在学中．2010 年立命館大学情報理工学部情報システム学科卒業．データマイニングの研究に従事．情報処理学会学生会員．日本データベース学会学生会員．

中村健二 Kenji NAKAMURA

立命館大学情報理工学部助手．2009 年関西大学大学院総合情報学研究科博士課程後期課程修了．博士（情報学）．知識情報処理，Web マイニング等の研究に従事．2002～2010 年株式会社関西総合情報研究所にて活動．情報処理学会，土木学会，日本データベース学会各会員．

小柳滋 Shigeru OYANAGI

立命館大学情報理工学部教授．1977 年京都大学大学院博士課程修了．博士（工学）．データマイニング，コンピュータアーキテクチャの研究に従事．IEEE CS，ACM，情報処理学会，電子情報通信学会，日本データベース学会各会員．