1P-3

ストリーム処理システムにおける時系列異常検出手法の共有計算技法

大桶真宏† 川島英之‡ 北川博之‡
筑波大学情報科学類† 筑波大学システム情報系情報工学域‡

1. はじめに

トラフィック分析によるマルウェア検知の手法には様々なものが現存する．新種のマルウェアが毎年現れる点，ならびにマルウェアを防ぎたいという要求がなくならない点を鑑みれば，検知手法はこれからも増加すると考えられる．

多数のマルウェア検知手法を効率的に運用することは容易ではない．複数のマルウェア検知手法を運用する際，単純な方式として，各手法を別のプログラムとして実装し，別々にコンパイルして，それらを別のプロセスとして動作させる方式が考えられる．この方式は単純であるために実現が容易だが，次の 2 つの問題が存在する．

第一の問題は,運用・開発に関する問題である．プログラムの整理・起動が煩雑であることに加えて，処理結果を受信するインターフェースが統一化されているとは考えられないため，処理結果を利用したプログラムの作成は複雑・困難となる． 第二の問題は，性能に関する問題である．システムの入力であるパケットストリームは別々の形式で処理されるため，それぞれのプロセスに同じパケットデータをフィードせざるを得ない．

これらの問題を解決するために，我々はストリーム処理システム(Stream Processing System, SPS)をベースにしたマルウェア検知基盤システムを開発してきた．提案システムはパケットをリレーショナルストリームとしてモデル化し，SPS で処理することで上記の性能に関する問題を解決する．次に提案システムは分析手法を SQL ライクな問合せ言語で記述可能にする．これによりユーザは SQL を発行すれば結果をタプルストリームとして受信可能になり，運用の問題は解決する．

マルウェアによるアクセスパターンをトラフィックにおける異常アクセスと見做すことがある．この異常アクセスを検知する手法として，時系列回帰分析がある．例えば NICTER[1]では Change Point Detection(CPD)[2]と呼ばれる，忘却の概念を導入した回帰分析手法を用いて異常アクセスを検知し，異常アクセスをマルウェアによるアクセスと見做す研究が行われている．しかし CPD はパラメータ設定を誤ると侵入検知が不能になる可能性がある．これを防ぐために，異なるパラメータを有する複数の CPD を並列に動作させることが求められる．

このような並列動作を高速化するために，本論文では SPS における CPD の共有計算技法を提案する．CPD の内部処理である SDAR を分析し，共有可能な部分についての考察を行う．

本稿では 2 節では提案システムについて述べ，3 節では CPD における共有計算技法とその技法を使用するシミュレーション実験について述べ，最後に 4 節では本論文をまとめる．

A Shared Computation Technique of Time Series Anomaly Detection on Stream Processing System
Masahiro Oke†, Hideyuki Kawashima‡, Hiroyuki Kitagawa‡
†College of Information Science, University of Tsukuba.
‡Faculty of Information, Systems and Engineering, University of Tsukuba.

2. 提案システム

2.1. ストリーム処理システム

SPS とは，問合せを永続的に保持し，ストリームデータがシステムに到着する度に問合せを評価するシステムである．SPS では全ての問合せ処理は主記憶上で行うため，一度データベースに保存し必要に応じて問合せ要求を与えた処理結果を得る，従来のデータベース管理システムに比べて高速な処理が可能になる．

我々の開発している SPS の SS*は StreamSpinner[3]の後継として開発されている．各問合せは SQL ライクな問合せ言語により記述される．各問合せは SS*に登録され，ユーザの指示により起動/停止を行うことができる．起動した問合せはデータが到着する度に評価され，その結果をタプルストリームとして出力する．

SS*は外部ライブラリを呼び出す機構として組み込み関数がある．組込関数を C++で開発し，ライブラリとして実装することで問合せから呼び出し可能となる．また SS*では各問合せはスレッドで実現されるが，run/stop コマンドにより個別に起動/停止ができる．各問合せは SS*上で確認できる．SS*では入力データをタプルに変換した後，各問合せにタプルの複製が入力される．この複製作業はプロセス内部で実行される．

SS*を用いることで開発・運用・性能に関する問題を解決することができる．SS*では検知手法を組込関数として実装でき，さらに出力結果はタプルストリームで統一できるため，開発柔軟性を有する．また各問合せがどのような検知手法を使用しているか確認でき，起動/停止ができるため，管理が容易になる．また検知手法を含む各問合せに対しての入力パケットの複製作業はプロセス内部で実行されるため，性能に関する問題を解決する．

2.2. Change Point Detection (CPD)

CPD とは外れ値と変化点を検出する技法である．
CPD は AR モデルに逐次学習と忘却機能を導入した SDAR アルゴリズムを用いる．逐次学習とはデータを 1 つ読み込む度に AR モデルを学習する．忘却機能とは，忘却率 R を用いて，i 時点前のデータの影響が(1-R)×i 倍に減少するようにする機能である．

SDAR(Sequentially Discounting AR model learning)アルゴリズムとはT時間のデータからARモデルを学習するアルゴリズムである.

時系列データに対して第一段階のSDARを適用して外れ値を検出し，第二段階のSDARを適用して変化点を検出する．アルゴリズムの流れは下記のようになる.

1:x_tを入力
2:SDARでx_tの確率密度を学習
3:外れ値スコアを計算
4:スコアの移動平均y_tを計算
5:SDARでy_t確率密度を計算
6:変化点スコアを計算
7:2へ戻る

アルゴリズム1:Change Point Detection

3. 提案手法

3.1. CPDの共有計算技法

CPDには3つのパラメータ(忘却率R, AR次数K,移動時間 T)をユーザが指定する必要がある．パラメータが異なれば検知率が異なるため，適切なパラメータを与える必要があるが，適切なパラメータを自動的に見つける方法は我々が知る限り存在しない.

これらのパラメータのうち，移動時間 T は移動平均スコアと変化点スコアの計算に用いられる．忘却率 R およびAR次数KはSDARで用いられる．またSDARの最初の計算である平均μの計算では忘却率Rのみを用いる.

ある入力に対してパラメータの異なる複数の CPD を適用する場合，以下の場合において計算結果の共有が可能になる.

1. 移動時間 T が異なるが，忘却率 R と AR 次数 K が等しい複数の CPD を実行する場合

この場合では第一段階における SDAR の結果を共有することができる．N 個の CPD を適用する場合，共有しない場合 2N 回の SDAR の計算が必要になるが，共有した場合は最小で(N+1)回で済む.

2. 移動時間 T と AR 次数 K が異なるが，忘却率 R が等しい場合

この場合では AR 次数が異なるため，第一段階 SDAR の結果を共有することはできないが，第一段階 SDAR の最初の計算である平均μの計算結果を共有することができる.

SS*においての共有化方式の実装を行う際には，共有部分を 1 つの演算子とし，非共有部分は異なる演算子とすればよい.

3.2. シミュレーション実験

上記の 2 つの手法についてシミュレーション実験を行った．実験機は CPU:Intel(R) Xeon(R) CPU E5640 @ 2.60GHz(4 cores) Dual CPU, RAM: 64GB, の機能を有する.

各実験においては 100 件の CPD を実行させ，10000 件のデータを読み込み，全ての処理を終えるまでの時間を測定した．3 種類のパラメータセットにおいて実験を行った結果(実験 1, 2, 3)を表 1 に示す．実験 1 は最も計算量の少ないパラメータ，実験 2 は参考文献[2]で使用されたパラメータ，実験 3 は最も計算量の多いパラメータ設定となっている．なお，表 1 において，x_R, x_K, x_Tは第一段階 SDAR のパラメータ，y_R, y_K, y_Tは第二段階 SDAR のパラメータを表す.

実験結果は，手法 1 により最大 187%の性能向上が得られたことを示している．第一段階 SDAR において AR 次数が大きい場合には計算量が増大するため，特に顕著な性能向上が観察されたと考えられる.

一方，手法 2 による顕著な性能向上は観察されなかった.

	実験 1	実験 2	実験 3
忘却率 x_R	0.02	0.02	0.02
AR 次数 x_K	1	2	9
移動時間 x_T	5	5	10
忘却率 x_R	0.02	0.02	0.02
AR 次数 x_K	1	3	9
移動時間 x_T	5	5	10
処理時間(共有無)	2003ms	2943ms	10102ms
処理時間(手法 1)	1164ms	1820ms	5387ms
処理時間(手法 2)	1943ms	2886ms	10068ms
性能向上率(手法 1)	172%	162%	187%
性能向上率(手法 2)	103%	102%	100%

表 1:シミュレーション実験結果

4. まとめ

本論文ではストリーム処理システムをベースにしたマルウェア検知基盤システムについて述べた後，Change Point Detection(CPD)を SS*に実装する方法を述べた．そして複数の CPD を効率的に実行するために CPD の一部を共有計算する手法を提案した．簡易実験の結果，共有計算により最大で 187%程度の性能向上が得られたことを観察した.

今後の課題はさらなる性能向上と，SS*への実装である.

謝辞

本研究成果の一部は科研費(#24500106, #24240015)および独立行政法人　情報通信研究機構(NICT)の委託研究「新世代ネットワークを支えるネットワーク仮想化基盤技術の研究開発」により得られたものである.

参考文献

[1] Daisuke Inoue, Katsunari Yoshioka, Masashi Eto, Masaya Yamagata,Eisuke Nishino, Jun'ichi Takeuchi, Kazuya Ohkouchi, and Koji Nakao, "An Incident Analysis System NICTER and Its Analysis Engines Based on Data Mining Techniques", ICONIP 2008, Part I, LNCS 5506, pp. 579-586, 2009.

[2] J. Takeuchi and K. Yamanishi, "A Unifying Framework for Detecting Outliers and Change Points from Time Series," IEEE transactions on Knowledge and Data Engineering, pp.482-492, 2006.

[3] StreamSpinner:http://www.streamspinner.org/