

## リアルタイム分析基盤システムの提案

川島英之<sup>†1</sup>

本論文では我々が開発しているリアルタイム分析基盤システム SS\*<sup>†1</sup>について述べる。SS\*はリレーショナルデータモデルを基礎とし、回帰分析手段と暗号化データ処理機能を有する。

## A Proposal of a Real-Time Analysis Infrastructure System

HIDEYUKI KAWASHIMA<sup>†1</sup>

This paper describes a real-time data analysis infrastructure SS\*. SS\* is based on relational data model and it provides a regression function and encrypted data processing function.

## 1. はじめに

スマートグリッド、ネットワークにおける攻撃検知、通信ログ分析、粒子加速器、工場における異常検知、地球観測衛星、科学シミュレーション等、様々な分野でデータ処理が行われている。

データを分析する基盤技術としては、大まかに 2 種類のものがある。1 つはバッチデータ処理方式であり、もう 1 つはリアルタイムデータ処理方式である。バッチデータ処理方式の例には RDBMS や Hadoop が挙げられる。リアルタイムデータ処理方式には SPE や S4, STORM が挙げられる。本論文ではリアルタイムデータ処理方式の 1 方式であるストリームデータ処理方式、およびその方式を実現しているソフトウェア基盤である SPE (Stream Processing Engine) に着目し、その 1 実装である SS\*<sup>†1</sup>について述べる。なお、ここで記すところのリアルタイムデータ処理方式とは、デッドラインに基づく計算方式ではなく、出力を低遅延で発生させる方式である。

本論文で述べるところのストリームデータ処理方式とは、データがシステムに到着した後、システムが二次記憶に触れることなくメモリアクセスまででデータ処理を行い、その結果を出力する方式である。

我々はその中でリレーショナルデータモデルを基礎とする SPE を開発している。リレーショナルデータモデルを基礎とする利点は、選択・射影・結合・集約などの利便性の高い演算子が組込処理として実現されているからである。組込処理であるため、それらの高速化をシステム開発者が実現することができる。一方、データモデルを特に提供しないシステムもある。そのような場合には処理記述の自由度が存在するが、性能は大きく劣化する。

さて、リレーショナルデータモデルを基礎とする SPE に

は様々なものがある。古い学術的研究としては、Borealis [11], STREAM [8], TelegraphCQ[21], MavStream [20], SASE[12] などがある。一方、商用化されているシステムには uCSDP [22], System S [10], Sybase が挙げられる。

これらの全てのシステムにおいて不足している点は、分析手段である。リレーショナルデータモデルを基礎とするシステムはリレーショナル演算子を提供するが、それ以外の手法を提供しない。同様に XML 処理系では XML データを処理する演算は提供するが、分析技術は提供しない。また、KVS では PUT/GET などのデータ挿入・取得手段は提供するが、分析手段は提供しない。分析基盤システムには分析手段とデータ処理系と結びつける手段が必要である。

## 1.1 データ処理系と分析手段の統合方式

通常、DBMS は選択・射影・結合・集約などの基本的演算しか提供せず、分析システムは分析処理のみを提供する。これらをまとめてしまい、DBMS 内部で解析処理を行う方式は in-DBMS analytics と呼ばれる [6, 18, 19]。DBMS は特定の演算のみを提供する。例えば RDBMS の場合には選択・射影・結合・集約などの限られた種類の演算のみを提供する。そのため、機械学習・データマイニングなどの分析を行いたいユーザにとっては処理機能が不足する。そこで DBMS 内部に機械学習・データマイニング等に関する様々な処理機能を追加するというコンセプトが in-DBMS analytics である。この例には MauveDB [7], WaveScope [17], MADlib [6, 19], Bismarck[18] が挙げられる。

上とは逆に解析処理を DBMS 外部で行う方式を out-DBMS analytics 方式と記す。Out-DBMS analytics の例としては SciDB と Python or R クライアントとの連携、PostgreSQL と Mathematica の連携などが考えられる。あるいは Mahout [3], Mallet [4], Jubatus [5]などの分析システムとデータ処理基盤の連携も考えられる。

In-DBMS analytic 方式が out-DBMS analytics 方式に対し

<sup>†1</sup> 筑波大学システム情報系  
Faculty of Information, Systems and Engineering, University of  
Tsukuba.  
kawasima@cs.tsukuba.ac.jp

て優れる点は、性能とユーザインタフェースである。性能差はデータ移動に起因する。out-DBMS analytics 方式においては、分析を行うには DBMS 内部のデータを外部へ移動しなければならない。データ量が小さければ、この移動に関するコストは小さい。しかしながら巨大なデータを扱う場合には、その移動コストは莫大になる。なお、データ移動量を調整する研究 [14]も存在する。

次にユーザインタフェースに差が存在する。ここで我々の意図するユーザインタフェースとは、分析プログラムにおいて巨大データを論理的に記述可能か否か、である。R や Matlab では最大でもメモリに乗るだけのデータサイズでなければ配列を記述できない。それゆえプログラマはこの制約を満足させるために DBMS アクセスを考慮したプログラムを記述することを余儀なくされる。一方 in-DBMS analytics 方式ではメモリサイズを超える量のデータについても論理的には取扱いが可能になる。それゆえプログラマはメモリサイズを気にせずに分析ロジックを記述可能である。

ただし in-DBMS analytics 方式の実現には大きな設計・実装コストを要することを欠点として指摘されなければならない。

上記では DBMS における in-DBMS analytics について述べた。一方、in-SPE analytics は筆者らの知る限り存在しない。この方針で開発している SPE である SS\*について本論文では述べる。

本論文の構成は次の通りである。2 節で SS\*の概要を述べると共に、CPD と暗号化処理の実装手法を述べる。3 節で論文をまとめる。

## 2. SS\*

SS\*は C++により実装されており、論文執筆時、40 万タプル/秒程度の性能を有する。SQL ライクな問合せを記述するとそれを演算木へ変換し、tuple at a time 方式でタプルを処理する。

SS\*が現在提供する分析手段は回帰分析 (CPD [1]) のみである。CPD については 2.1 節で述べる。一方、分類として SVM と Bayesian network、クラスタリングとして k-means と LDA、類似シーケンス検索として Euclidean 距離と DTW 距離、推薦技術として協調フィルタリングを導入中である。

また、暗号化機能も有しており、選択・結合・集約に関して暗号化したデータ処理を行える。これについては 2.2 節で述べる。

### 2.1 異常検知手法の実装

#### 2.1.1 異常検知手法：Change Point Detection

SS\*には Change Point Detection (CPD) [1]と呼ばれる技法を実装してある。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  $i$  時点前のデータの影響が  $(1 - R) \times i$  倍に減少するようにする機能である。

SDAR (Sequentially Discounting AR model learning)アルゴリズムとは現在までの  $T$  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。SDAR は学習した AR モデルを用いて現在のデータを推定し、実際の現在のデータに対する推定値の外れ値らしさを計算する。

時系列データに対して第一段階の SDAR アルゴリズムを適用して外れ値を検出し、第二段階の SDAR アルゴリズムを適用して変化点を検出する。アルゴリズムの手続き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  $y_T$ の確率密度を学習;
- 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

#### 2.1.2 CPD の実装

我々は Eigen [13]と C++言語を用いて CPD を実装した。Eigen を利用した理由は、CPD の計算において逆行列、転置行列、行列積などを求める必要があったからである。Eigen の利用により、CPD に要したコード行数は 300 程度で済んだ。SS\*においては CPD を組込関数として実装し、リレーショナル演算として表現した。具体的には次のパラメータを演算子に管理させる実装を行った。パラメータを関数レベルで管理すると、複数の CPD がパラメータを共有してしまうことになり、本来異なるべきであるモデルが混合してしまって意味をなさなくなる。それゆえ演算子によるパラメータ管理が必要であった。

- ✓ 時刻
- ✓ 第一段階 SDAR から得られる確率密度
- ✓ 第二段階 SDAR から得られる確率密度
- ✓ 入力ベクトル  $x_1 \dots x_T$
- ✓ 期待値  $\mu$
- ✓ 新しい入力推定値
- ✓ 共分散  $C_0 \dots C_K$
- ✓ AR 係数  $A_1 \dots A_K$
- ✓ 共分散行列

これにより我々は CPD を集約関数の一種として扱うことを可能にし、Grouping 演算からも呼出すことを可能にした。即ち、下記のような問合せを実行可能にした。この問合せでは CPD はデフォルトパラメータで動作する (3 つのパラメータを設定可能：忘却率  $R$ , AR 次数  $K$ , 移動時間  $T$ )。そして dst\_port 毎に AR モデルを構築し、1 秒間のウィ

ンドウについて、各ポートに関する CPD スコアを計算して出力する。

```
SELECT dst_port, cpd()
FROM packet[1 sec]
GROUP BY dst_port;
```

CPD の利用に際してはパラメータが重要である。パラメータによっては異常値を発見できない場合がある。そこで異なるパラメータを有する複数の CPD を並列的に動作させることが異常検知には重要である。このとき、共有計算方式を用いて性能を高める方法を我々は考案している。この詳細については [25] を参照されたい。

## 2.2 暗号化ストリーム処理

RDBMS における暗号化データ処理方式として CryptDB [24] が知られている。一方、我々の手法は CryptDB とは決定的に異なる。CryptDB は Database proxy と呼ばれる単一のモジュールによって暗号化及び復号を実現している。しかし一般的に情報源とユーザとが広範囲に分散しているストリームデータ処理において、このようなスキームが適用できるケースは極めて稀であると言える。そのため、我々の手法では暗号化及び復号を行うためのそれぞれ独立したモジュールを用意する。

DET として用いる AES-CMC、及び OPE として用いる Boldyreva らにより提案された暗号化アルゴリズムは共通鍵暗号方式である。そのため我々の手法においては、事前に各 Encryption module 及び Decryption module の間で共通鍵を共有しておく必要がある。本稿では、各モジュールは安全な経路を用いて事前に共通鍵を共有しているものとする。

前述したとおり、Encryption module は入力されたタプルの各要素に対して 3 種類 (DET, OPE, HOM) の暗号値を生成する。そのため、暗号化されたタプルのデータサイズの合計が平文タプルのデータサイズに対して著しく増加することが考えられる。以上より、我々の手法では、以下に示す 2 つの問題点が生じる。

### ✓ 課題 1: タプルサイズの増加

我々の提案手法では 1 つの平文値に対して 3 つの暗号値が生成される。また、一般に暗号値は平文値よりも大きい。それゆえに、SPE に転送される暗号化タプルサイズは平文タプルに対して 3 倍以上の大きさになると考えられ、通信帯域を圧迫してしまう。

### ✓ 課題 2: SPE のメモリ消費量の増加

課題 1 と同様の理由により、SPE 上の処理木の各シノプシスに暗号化タプルが貯まることで、平文に比べて SPE のメモリ使用量が増加してしまう。

これらの問題に対処するために、我々はこれまでに効率

化手法について研究を行ってきた。この詳細は [23] を参考にさせて頂きたい。

## 3. まとめ

本論文ではリアルタイム分析基盤システムである SS\* について述べた。特に CPD の実装と暗号化処理方式についての説明を行った。今後の課題は SS\* の高機能化である。

**謝辞** 本研究成果の一部は科研費[#24500106]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) Jun'ichi Takeuchi and Kenji Yamanishi, "A nifying Framework for Detecting Outliers and Change Points from Time Series," IEEE transactions on Knowledge and Data Engineering, pp.482-492, 2006.
- 2) 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.
- 3) Mahout: <http://mahout.apache.org/>
- 4) Mallet: <http://mallet.cs.umass.edu/>
- 5) Jubatus: <http://jubat.us/>
- 6) MADlib: <http://madlib.net/>
- 7) Amol Deshpande and Samuel Madden. "MauveDB: supporting model-based user views in database systems", SIGMOD 2006.
- 8) Arvind Arasu, Shivnath Babu, and Jennifer Widom. "The CQL continuous query language: semantic foundations and query execution", The VLDB Journal 15, 2 (June 2006), 121-142.
- 9) StreamSpinner: [www.streamspinner.org/](http://www.streamspinner.org/)
- 10) Bugra Gedik, Henrique Andrade, Kun-Lung Wu, Philip S. Yu, and Myungcheol Doo. "SPADE: the system s declarative stream processing engine", SIGMOD 2008.
- 11) Yanif Ahmad, Bradley Berg, Uğur Cetintemel, Mark Humphrey, Jeong-Hyon Hwang, Anjali Jhingran, Anurag Maskey, Olga Papaemmanouil, Alexander Rasin, Nesime Tatbul, Wenjuan Xing, Ying Xing, and Stan Zdonik. "Distributed operation in the Borealis stream processing engine", SIGMOD 2005.
- 12) Eugene Wu, Yanlei Diao, and Shariq Rizvi. "High-performance complex event processing over streams", SIGMOD 2006.
- 13) Eigen: <http://eigen.tuxfamily.org>
- 14) Patrick Leyshock, "Agrios: A Hybrid Approach to Scalable Data Analysis Systems", XLDB 2012.
- 15) Yousuke Watanabe, Hiroyuki Kitagawa, "Query Result Caching for Multiple Event-driven Continuous Queries", Information Systems, Elsevier, Vol.35, No.1, pp.91-110, January 2010.
- 16) NEGI: <https://github.com/westlab/negi>
- 17) WaveScope: [http://www.cs.indiana.edu/~rrnewton/wavescope/WaveScope\\_+\\_WaveScipt/WaveScope\\_Homepage.html](http://www.cs.indiana.edu/~rrnewton/wavescope/WaveScope_+_WaveScipt/WaveScope_Homepage.html)
- 18) Aaron Feng, Arun Kumar, Benjamin Recht, and Christopher Ré, "Towards a Unified Architecture for In-Database Analytics", SIGMOD, 2012
- 19) Joseph M. Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleks Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, and Arun Kumar, "The MADlib Analytics Library or MAD Skills, the SQL", PVLDB 2012

- 20) Sharma Chakravarthy and Qingchun Jiang. 2009. Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing (1st ed.). Springer Publishing Company, Incorporated.
- 21) Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel R. Madden, Fred Reiss, and Mehul A. Shah. 2003. TelegraphCQ: continuous dataflow processing. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD '03). ACM, New York, NY, USA, 668-668.
- 22) クラウドサービスプラットフォーム Cosminexus : uCosminexus Stream Data Platform, <http://www.hitachi.co.jp/Prod/comp/soft1/cosminexus/sdp/>
- 23) Katsuhiro Tomiyama, Hideyuki Kawashima, and Hiroyuki Kitagawa, “A Security aware Stream Data Processing Scheme on the Cloud and its Efficient Execution Methods”, Proc. Third International Workshop on Cloud Data Management (CloudDB'12)
- 24) Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: protecting confidentiality with encrypted query processing. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP '11). ACM, New York, NY, USA, 85-100.
- 25) 川島英之, 大桶真宏, 北川博之, “ストリームデータ処理における異常検知手法の共有化に関する検討”, 情報処理学会第 156 回データベースシステム研究会.