6U-01

完全準同型暗号を用いた秘匿データマイニングの分散処理による高速化の実装と検証

山本 百合 †　　小口 正人 †
† お茶の水女子大学

1. はじめに

データマイニングは膨大なデータから有益な情報や特徴的なパターンを発見するために考案された手法であり，企業の商業的活用や分析などの様々な場面での利活用が期待されている．ビックデータを用いるデータマイニングでは計算資源が必要であることから，各企業が保持するデータをクラウドやデータセンタ等の外部機関に委託し，利用者が問い合わせを行うことで結果の取得が可能な委託データマイニングシステムが提案されている．しかしプライバシ保護の観点から，データの外部委託の際には暗号化によるデータの秘匿が必要である．そのためデータを暗号化した状態で乗算と加算の操作が可能な完全準同型暗号を利用することで，安全な委託計算システムの構築を目指す研究が近年盛んである．

先行研究 [1] では，完全準同型暗号を Apriori アルゴリズムによる秘匿データマイニングに適用し，アルゴリズムの高速化を進めている．しかしながら，完全準同型暗号演算などの演算に関する処理は，計算量が大きいためにサーバ側での計算負荷が大きくなりやすい．本研究では，先行研究が用いている手法のサーバ側での演算に対して，Apriori アルゴリズムにおけるアイテムセットごとでデータベースを分割する分散処理を適用し，秘匿データマイニングシステムの高速化を行った．

2. 先行研究

今林ら（2016）は，Liu ら（2015）が提案した完全準同型暗号を用いた安全委託頻出パターンマイニング手法 P3CC[2] に対して，暗号文パッキングによる計算量の削減と，各段階で算出したサポート値をキャッシングすることによって，結果を再利用する手法を提案し，計算量の削減を行った [1]．安全委託頻出パターンマイニングシステムは，トランザクションごとに購入したか否かを 0 または 1 のバイナリ表現されたバスケットデータを対象とし，Apriori 計算を行うサーバ・クライアント型のシステムとして設計されている．ただし完全準同型暗号は加算と乗算の機能を有するが，比較をすることは極めて困難な暗号である．そのため Apriori で必要とされるミニマムサポートとの比較は，クライアント側で結果を復号した上で比較を行うので，サーバ・クライアント間のコミュニケーションは最大でアイテム数分の回数生じる．先行研究により実行時間は大幅に削減されたが，完全準同型暗号は暗号文同士の演算において計算量が大きいことから，クラウド環境への実用化に向けて暗号文同士の演算の多いサーバ側における計算時間の改善を試みる．

Implementation of Distributed System for Secure Data Mining with Fully Homomorphic Encryption
† Yuri YAMAMOTO, † Masato OGUCHI
Ochanomizu University (†)

3. 提案手法

3.1 概要

本研究では，図 1 の完全準同型暗号を用いた秘匿データマイニングのマスタ・ワーカ型の分散システムを提案する．

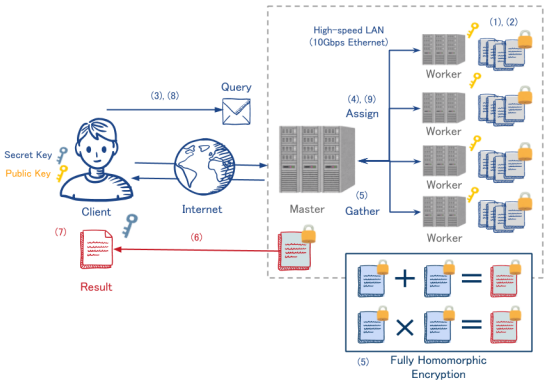

図 1: 提案手法概観

(1) クライアントはデータを秘密鍵で暗号化し，マスタに公開鍵と共に委託．
(2) マスタは受け取ったデータと公開鍵を各ワーカに転送．
(3) クライアントはマスタに長さ 1 のアイテムセットに対するサポート値の計算依頼を行う．
(4) マスタは各ワーカにタスクを割り振る．
(5) 各ワーカは完全準同型暗号を用いたサポート値計算を行い，結果をマスタへ転送．
(6) マスタはクライアントへ収集した結果を送信．
(7) クライアントはデータを復号し，各アイテムセットのサポート値と閾値を比較．
(8) クライアントは閾値を超えたアイテムセットをマスタに送信．
(9) マスタは計算対象を閾値を超えたアイテムセットに各アイテムを 1 つ追加したアイテムセットに設定．
(10) (4)〜(9) を閾値を超えるアイテムセットが無くなるか，アイテムセットが最長になるまで繰り返す．

将来的に本システムがクラウドコンピューティング環境に実装されることを想定し，分散処理化によってクライアントに変化が生じないようにマスタ・ワーカ型分散処理を行った．

3.2 分割方法

アプリケーションのタスクの分割方法として，(1) データベースの分割，(2) 独立性の高い計算の分割，(3) 独立性の高い手順を別タスクとして分割などが考えられる．またサーバ側で暗号文同士の演算が行われている箇所は，(a) 相関性を調べたい複数のアイテムごとのパッキングベクトル同士の掛け算，(b) パッキングベクトル内の頻出度の合計値算出，(c) パッキングベクトルごとの合計値をさらに

アイテムセットごとで足し合わせた合計値算出の3ヶ所である．ただし暗号文パッキングはアイテムごとに複数のトランザクションに対して行われている．今回はサーバ側で行われる暗号文同士の演算が最も計算量が大きいことから，全ての完全準同型暗号計算をタスクとして分割する，データベースの分割をアイテムセットごとで行う方法を採用した．

4. 実験

4.1 実験環境

実験環境は，Intel®Xeon ®Processor E5-2643 v3 3.4GHz，6コア，12スレッド，メモリ容量512GB，ストレージはRAID0のSSDが480GB，HDDが2TBであり，同スペックのマシンを4台使用する．1台をマスタの機能を持ったマシンとし，同時にワーカとして1スロット分の演算も行う．また他3台をワーカとして各々最大2スロット稼働させ，最大で7スロット分のワーカを稼働させてワーカ数ごとの実行時間を比較する実験を行った．先行研究と同様に，実験に使用する頻出パターンマイニングのデータセットはIBM Almaden Quest research groupが開発したジェネレータで生成した．今回はアイテム数50，トランザクション数330で生成したデータを使用する．C++で実装し，完全準同型暗号計算にはHElib[3]を，分散化における各マシンの制御のためにOpen MPIを用いた．

4.2 実験結果

ワーカ数ごとの実行時間のグラフを図2に示す．

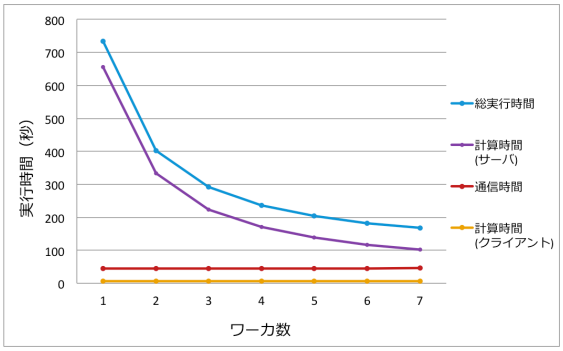

図 2: ワーカ数ごとの実行時間 (秒)

ワーカ数が増加するにつれて総実行時間を短縮することができた．特にサーバ側の計算時間で分散化効果が顕著である．通信時間とクライアント上の計算に関しては，ワーカ数の増加に対してほとんど変化しないことが示された．

また分散処理化の評価として高速化率を逐次実行時間(秒)/並列実行時間(秒)で算出し，式(1)に基づいたAmdahlの法則による並列度と高速化率の関係[4]と共に図3に示す．

$$\text{高速化率} \leq \frac{1}{(1 - \text{並列実行時間の割合}) + \frac{\text{並列実行時間の割合}}{\text{ワーカ数}}} \quad (1)$$

サーバ側の完全準同型暗号の暗号文同士の計算時間は99%並列時に近い高速化率で分割されていることが示され

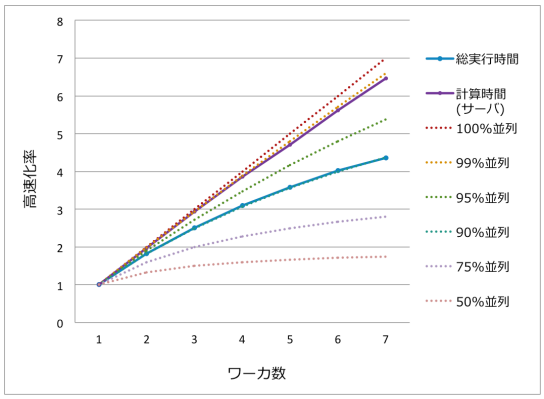

図 3: ワーカ数ごとの高速化率とAmdahlの法則による並列度に対する高速化率

た．ただし，通信時間，クライアント側での暗号化や復号に伴う計算時間，ファイル入出力等に必要な時間も含めた総実行時間では，90%並列時の高速化率となる．したがって，本システムはワーカ数を最大限に増やす場合，最大で約10倍の高速化率が期待できる．

5. まとめと今後の課題

完全準同型暗号を用いた秘匿データマイニングシステムに，マスタ・ワーカ型の分散処理を適用することによって高速化する手法を提案した．今後は秘匿データマイニングシステムにおいて，データベースの更新時の高速化手法を検討する．特にデータベースの更新時に伴う再計算の最適化を目的とする改良をAprioriアルゴリズムに対して行ったFUPアルゴリズム[5]への秘匿計算適用を対象とし，クラウド環境への適用を考える．

6. 謝辞

本研究を進めるにあたり，大変有益なアドバイスを頂いた早稲田大学山名研究室並びに工学院大学山口研究室の皆様に感謝いたします．

本研究は一部，JST CREST JPMJCR1503の支援を受けたものである．

参考文献

[1] Hiroki Imabayashi, Yu Ishimaki, Akira Umayabara, Hiroki Sato, and Hayato Yamana. Secure frequent pattern mining by fully homomorphic encryption with ciphertext packing. In *International Workshop on Data Privacy Management*, pp. 181–195. Springer, 2016.

[2] Junqiang Liu, Jiuyong Li, Shijian Xu, and Benjamin CM Fung. Secure outsourced frequent pattern mining by fully homomorphic encryption. In *International Conference on Big Data Analytics and Knowledge Discovery*, pp. 70–81. Springer, 2015.

[3] Shoup V. and Halevi S. HElib. `http://shaih.github.io/HElib/index.html`. 2017年1月閲覧.

[4] Clay Breshears. *The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications*. O'Reilly Media, Inc., 2009.

[5] David W Cheung, Jiawei Han, Vincent T Ng, and CY Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pp. 106–114. IEEE, 1996.