完全準同型暗号を用いた秘匿データマイニング計算のデータベース更新時の分散処理による高速化

山本 百合[1]　小口 正人[1]

概要：データマイニング計算を外部に委託し，利用者が問い合わせを行う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となる．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は，計算量が大きいためにサーバ上の計算負荷が大きくなりやすい．本研究では，Apriori アルゴリズムに対してデータベースの更新に伴う再計算の最適化を目的とする改良を行った FUP アルゴリズムでの秘匿データマイニング計算委託システム実装と分散処理化を行なった．またサーバ上の演算に対してマスタ・ワーカ型の分散処理を実装することで，クラウドコンピューティングを想定した環境での高速化について検討する．

Distributed System for Secret Data Mining while Updating Large Itemsets using Fully Homomorphic Encryption

YURI YAMAMOTO[1]　MASATO OGUCHI[1]

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 [1][2] 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．データマイニングの活用例として有名なバスケット分析などの購買履歴を用いるデータマイニングでは，膨大なデータ量が蓄積されるため，統計処理に処理能力の高い計算機での計算が必要となる．そのため各企業が所有する膨大なデータをクラウドやデータセンタ等の大型の計算機とストレージを所有する機関に委託し，利用者が問い合わせを行うことで統計計算が可能な委託データマイニングシステムが今後広がっていくと考えられる．しかし個人情報や機密性の高い情報を扱う場合，適切なセキュリティ管理が求められる．プライバシ保護データマイニングに関する研究の多くは，入力あるいは出力データに対し，抽象化・ランダム化・ノイズ付加・匿名化することによって個人情報の漏洩を防ぐ手法が一般的だが，データの曖昧性を高めることからデータマイニング結果への影響が懸念される．機密性の高いデータの安全性と解析結果の正確性の双方を両立するために，暗号を利用した秘匿検索手法によるデータ活用が必要である [3].

先行研究 [4][5][6][7] では，暗号文同士の加法と乗法が成立する完全準同型暗号を Apriori アルゴリズムによる秘匿データマイニングに適用し，サーバ側が暗号化されたデータの復号を行うことなく，統計処理などの複雑な演算が可能な手法を目指し，アルゴリズムの高速化を進めている．特にデータを個々に暗号化するのではなく，複数のデータをひとつのベクトルとして扱う高速化手法を用いることで高速化に貢献している．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，Apriori アルゴリズムの発展として，データベースの更新に伴う再計算の最適化を目的とする改良を行った FUP アルゴリズム [8] を本研究の実装に適用し，データベース更

[1] お茶の水女子大学大学院人間文化創成科学研究科

新が発生するアソシエーション分析に対応し，システムの高速化を目指す．また本システムの高速化手法としてマスタ・ワーカ型分散処理を適用し，さらなる高速化に向けて検討を行いたい．

2. 完全準同型暗号

完全準同型暗号とは式 (1)，(2) のように暗号文同士の加算と乗算の演算が成立する性質を持ち，暗号化した状態で平文と同様の多項式演算が可能な暗号である．完全準同型暗号は公開鍵暗号方式の機能を持つが，秘密鍵を用いることなく暗号文同士の演算から平文同士の演算を暗号化した値を導くことが可能となる．そのため，秘匿データマイニングに完全準同型暗号を適用することで，秘密鍵を渡すことなくサーバがデータを用いた統計処理を行えると期待できる [9]．また完全準同型暗号の概念自体は，1970 年代後半に公開鍵暗号が考案された当初より提唱されていたが，2009 年に Gentry[10] が実現する手法を提案した．提案当時は計算量の大きさから実用性が乏しかったが，その後も様々な研究によって高速化や改良が進められ，簡単な計算であれば十分な性能レベルを示している [11]．しかし暗号の解読困難性を保つためのノイズの付加と解読時のノイズの除去のための工夫の必要性などから，依然として複雑な計算や大きなデータに対する演算では暗号文サイズが大きくなる傾向にあり，計算量が大きくなる難点を持つ．

完全準同型暗号

$$Encrypt(m) \oplus Encrypt(n) = Encrypt(m+n) \quad (1)$$

$$Encrypt(m) \otimes Encrypt(n) = Encrypt(m \times n) \quad (2)$$

3. Apriori アルゴリズム

Apriori アルゴリズムはデータベースのアイテム間における相関ルール抽出の効率的な手法として，1993 年に Agrawal ら [12][13] に提案され，アソシエーション分析などの頻出データマイニングで用いられる．購買履歴を対象とする場合，トランザクションごとに商品の種類である各アイテムを購入したか否かをバイナリ表現で記録したデータベースから，各アイテムの部分集合であるアイテムセットの頻度計算を行う．頻度は全トランザクション数に対して対象のアイテムセットの全アイテムを購入したトランザクションの数の割合であり，これをサポート値と表現する．Apriori アルゴリズムは，図 1 のように探索するアイテムセットの長さを徐々に増加させながら，幅優先的に繰り返しサポート値計算と枝刈りを行うことで頻出なアイテムセットを導出する．

ここでアイテムセットとその部分集合のサポート値の関係を立式する．アイテムを I_k で表現し，アイテムセット $X+Y$ がアイテムセット X に $Y = I_1 I_2 ... I_k$ を加えたものと仮定する．また X を含むアイテムセットは x 個存在し，そのうち c 組はアイテムセット $X+Y$ を計算する前に生成された他のアイテムセットの数とする．またアイテム I_j の相対頻度を $f(I_j)$，トランザクション数を $DB.size$ と表すと，アイテムセット $X+Y$ のサポート値 $\bar{s}$ は式 (3) と表現できる．

$$\bar{s} = f(I_1) \times f(I_2) \times ... \times (x-c) \,/\, DB.size \quad (3)$$

$(x-c)/DB.size$ は X の実サポート値であるため，統計的独立の仮定から，$X+Y$ のサポート値 $\bar{s}$ は X のサポート値と Y のアイテムの個々の相対頻度の積で表現することが可能である．このことから X のサポート値が小さければ，$X+Y$ のサポート値が小さくなる．これにより頻出ではないアイテムセットを含むアイテムセットは頻出ではないことが言える．したがって Apriori ではサポート値がミニマムサポートを超えないアイテムセットの探索の延長を行わないことによって効率化を行う．

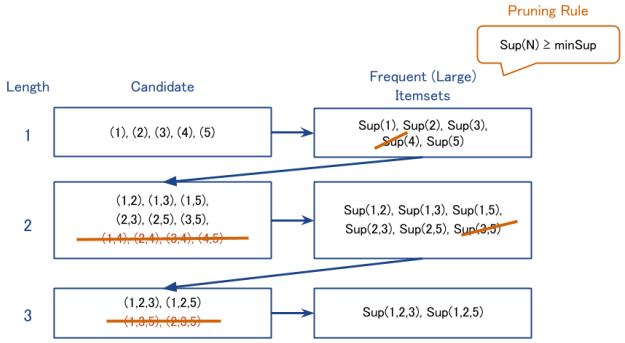

図 1　Apriori アルゴリズム

4. FUP アルゴリズム

FUP（Fast UPdate）アルゴリズムは 1996 年に Cheung らによって考案された手法であり，データベースのトランザクション追加後の Apriori 計算時に効率良くデータベース走査をするアルゴリズムである [8][14]．Apriori は静的なデータを扱うことを前提にサポート値算出を行っていたが，データベースにトランザクションが追加される場合，そのトランザクションの偏り次第では各段階で頻出と判定されるアイテムセットが変化することとなるため，サポート値の再計算が求められる．FUP アルゴリズムでは，データベース更新前の頻出アイテムセットの判定とサポート値の閾値を再利用することによって再計算コストの削減を試みる．更新前，更新分のデータベースをそれぞれ DB，db，そのトランザクション数を $DB.size$，$db.size$ と表現し，長さ i の候補アイテムセットの集合を C_i，頻出アイテムセット（ラージアイテムセット）の集合を L_i と表現する．アイテムセット X のサポート値は $Support(X)$ と表現する．なおサポート値という用語に関して，様々な文献で用いられ方が異なるが，本稿では出現数のサポートカウントとは区別

し，全トランザクション数に対する出現数の割合と定義する．またそのサポート値の閾値を $minS$ と表現する．FUP アルゴリズムの手順は以下の通りである．

(1) $i = 1$ のとき
  (a) 更新後のデータベース $DB + db$ に対して，更新前の頻出アイテムセット $X \subseteq L_1$ に関して，式 (4) を満たす X の集合を L'_1 とする．
  (b) 更新前の候補アイテムセット C_1 から更新前の頻出アイテムセット L_1 を取り除き，更新分データベース db に対して，$X \subseteq C_1$ を満たす X のうち，式 (5) を満たすものを C'_1 とする．
  (c) 更新後のデータベース $DB+db$ に対して，$X \subseteq C'_1$ のうち，式 (4) を満たす X を L'_1 に加え，L'_1 を長さ 1 の更新後の頻出アイテムセットとする．

(2) $i > 1$ のとき
  (a) 更新前の長さ i の頻出アイテムセット L_i から $X \subseteq L_{i-1} - L'_{i-1}$ を満たすアイテムセット X を除く
  (b) 更新後のデータベース $DB + db$ に対して，アイテムセット $X \subseteq L_i$ に関して式 (4) を満たす X の集合を L'_i とする．
  (c) 更新分データベース db に対して，L'_{i-1} から Apriori のアルゴリズムに従って導出した長さ i のアイテムセットから更新前頻出セット L_i を除いた集合を C_i とし，$X \subseteq C_i$ に関して，式 (5) を満たすものを C'_i とする．
  (d) 更新後のデータベース $DB+db$ に対して，$X \subseteq C'_i$ のうち，式 (4) を満たす X を L'_i に加え，L'_i を長さ i の更新後の頻出アイテムセットとする．

$$Support(X)_{DB \cup db} \geq minS \times (DB.size + db.size)/DB.size \quad (4)$$

$$Support(X)_{db} \geq minS \times db.size/DB.size \quad (5)$$

5. 先行研究

本章では，先行研究 [4][5] が実装した完全準同型暗号による Apriori アルゴリズムによるデータマイニングシステムの概観を述べる．

5.1 安全な委託データマイニング

Liu ら（2015）は，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(Privacy Preserving Protocol for Counting Cadidates) を提案した [15]．Liu らが提案した安全委託システムは，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．データの保持者であるクライアントはバイナリデータ 1 つずつ暗号化を行い，暗号化されたデータを受け取ったサーバがデータの中身を知ることなく完全準同型暗号を用いた Apriori 計算を行い，暗号化された状態のサポート値を算出するため，安全にデータマイニング計算を委託することが可能である．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，比較を行う． その結果，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．

5.2 多項式 CRT パッキングと暗号文キャッシング

高橋ら（2015）は従来の P3CC 手法に対して，アイテムの種類ごとに複数のトランザクションをベクトルとして暗号化し，準同型計算を行う，暗号文パッキングによる計算量の削減を行った [4]．データをベクトルとしてパッキングするにあたり，P3CC で使用されていた整数ベースの完全準同型暗号ではなく，複数の多項式に対して中国剰余定理（CRT: Chinese Remainder Theorem）を適用することで 1 つの多項式として表現する完全準同型暗号を用いている．

また今林ら（2016）は，各段階で算出した暗号化されたサポート値をキャッシングすることによって，Apriori 特有の繰り返しを生じる演算に対し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 [5][7][6].

これらの手法により，P3CC 手法の計算時間の大幅な改善が行われたが，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，実用化に向けて更に計算量の削減に邁進する必要があると考えられる．特に暗号文同士の演算の多いサーバ側における計算時間に改善の余地がある．

6. 提案手法

6.1 秘匿 Apriori 計算の分散処理手法

6.1.1 概要

本研究では，以下の完全準同型暗号を用いたアソシエーション分析のマスタ・ワーカ型の分散システムを提案する．提案システムの概要を図 2 に示す．

本システムの具体的な動作を以下に示す．

(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する．

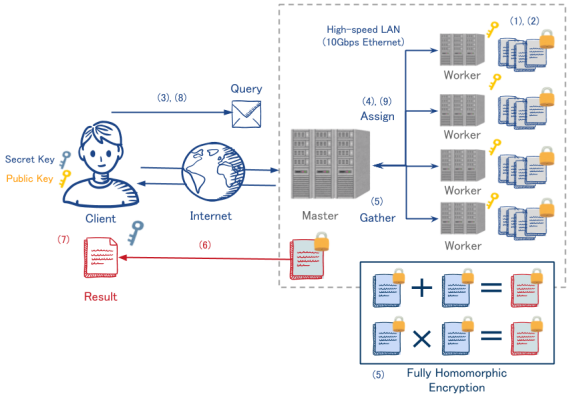

図 2　提案手法概観

(6)　マスタはクライアントへ収集した結果を送信する．
(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の比較を行う．
(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伝える．
(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側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．

6.1.2　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) データベースを分割する手法，(2) Σ計算などの独立性の高い計算を分割する手法，(3) 独立性の高い手順を別タスクとして分割する手法，などが考えられる．5.2 章で記述した通り，サーバサイドで行われる暗号文同士の加算と乗算の演算が最も計算量が大きいことが課題である．サーバサイドで完全準同型暗号の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．

今回の Apriori 手法では，一律のデータに対して算出されたサポート値と閾値の比較を行い，アイテム数を繰り上げるためにサーバとクライアント間でコミュニケーションが発生するため，全ての完全準同型暗号演算による負荷を分散化できるデータベースの分割を行い，特にアイテムセットごとでの分割を適用した．

6.2　秘匿 FUP 計算の分散処理手法

6.2.1　概要

6.1.1 章で実装したシステムを踏襲し，以下の手順を行うシステムを提案する．なおアルゴリズム中で使用する計算式は 4 章で提示したものである．

(1)　クライアントが暗号化した更新分データベースをマスタに転送．
(2)　クライアントは更新前データベースに対する頻出アイテムセットの情報から，4 章 (1) の計算依頼をマスタに送信．
(3)　更新分データベースに関して，4 章 (1) の (b) のサポート値計算をワーカに割り振る．
(4)　マスタは結果を収集し，クライアントに送信．
(5)　クライアントは復号し，式 (5) の比較を行い，頻出と判定したアイテムセットの更新後データベースでの再計算依頼をマスタに送信．
(6)　マスタは手順 (5) で新規に頻出と判定されたアイテムセットと手順 (2) で計算依頼されたアイテムセットに関して，4 章 (1) の (a) と (c) の計算をワーカに割り振る．
(7)　マスタは結果を収集し，クライアントに送信．
(8)　クライアントはデータを復号し，式 (4) の比較を行い，データベース更新後の頻出アイテムセットを得る．
(9)　(2)〜(8) の計算を 4 章 (2) の場合に関しても同様に行い，最終的に閾値を超えるアイテムセットが無くなるか，頻出アイテムセットが最長になるまで繰り返す．

6.2.2　分割方法

6.1.2 章と同様にアイテムセットごとでの分割を行った．今回の本研究の実装では手順 (3) のタスクを全てのワーカで分担し，その結果を含めて更新後データベースに対する計算をまとめて手順 (6) で行っている．更新後データベースに関する計算をまとめて行うことで，不必要なクライアントとのコミュニケーションを抑制できることから本手法は妥当だと考えられるが，FUP アルゴリズムの場合，手順 (3) の 4 章 (1) の (b) の計算と手順 (6) の一部の計算である 4 章 (1) の (a) の計算が独立であるため，ワーカごとで異なる計算を行うタスクを割り振ることも考えられる．今後の課題として，ワーカごとで異なる計算を行う手法での実装を行い，本研究の分散方法とワーカのタスクの割り振りを変更した分散方法の比較などを行うことで，更なる高速化に向けて検討したい．

6.3　実装方法

本研究では提案手法のシステム 2 つを C++で実装した．完全準同型暗号計算は，GitHub 上で公開されている準同型暗号計算ライブラリである HElib[16][17] で実装されている．また分散化における各マシンの制御のための規格であ

る，Message Passing Interface(MPI)[18] を利用するライブラリの Open MPI[19] を適用し，さらに C++拡張ライブラリの Boost MPI[20] によって制御を行うプログラムを実装した．

7. 実験

7.1 実験環境

実装したプログラムを複数のマシンにおいて実行した．各マシンの環境は，Intel® Xeon ® Processor E5-2643 v3 3.4 GHz，6 コア，12 スレッド，メモリ容量 512 GB，ストレージは RAID 0 の SSD が 480 GB，HDD が 2 TB 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 台をワーカとして最大 2 スロット稼働させた．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．先行研究ならびに P3CC 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．

7.2 秘匿 **Apriori** 計算の分散処理システムの実行時間評価

アイテム数 50，トランザクション数 19,800 のデータセットに対し，秘匿 Apriori 計算の分散処理実験を行った．ワーカ数ごとの実行時間のグラフを図 3 に示す．

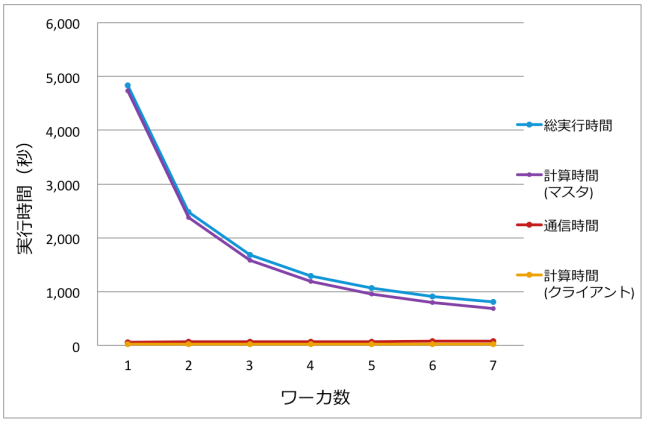

図 3　秘匿 Apriori 計算のワーカ数ごとの実行時間（秒）

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．特にマスタ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．

7.3 秘匿 **FUP** 計算の分散処理システムの実行時間評価

アイテム数 50，更新前トランザクション数 19,470，更新後トランザクション数 19,800 のデータセットに対し，秘匿 FUP 計算によるデータベース更新時再計算の分散処理実験を行った．ワーカ数ごとの実行時間のグラフを図 4 に，秘匿 Apriori 再計算による計算時間との比較を図 5 に示す．

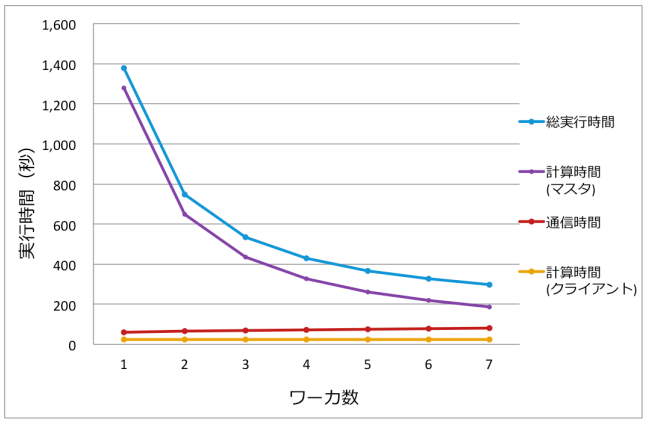

図 4　秘匿 FUP 計算システムのワーカ数ごとの実行時間（秒）

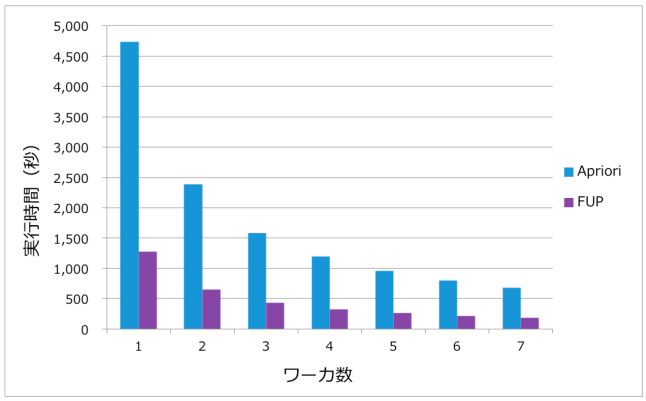

図 5　秘匿 Apriori 計算と秘匿 FUP 計算のワーカ数ごとの計算時間（秒）比較

図 4 より，秘匿 Apriori 計算と同様にワーカ数が増加するにつれて総実行時間は減少し，マスタ側の計算時間の分散化効果が顕著であることが示された．また通信時間とクライアント上の計算時間は，ワーカ数の増加に対してほとんど変化しない．

図 5 より，FUP アルゴリズムによる再計算は Apriori アルゴリズムによって計算を最初からやり直す再計算よりも，計算時間が約 3〜4 倍高速化される結果となった．FUP アルゴリズムはデータベース更新後の再計算において，更新前データベースの結果を再利用することによって，候補アイテムセットに関する演算を減らし，計算時間を短縮するアルゴリズムである．Cheung らの論文 [8] では，FUP アルゴリズムは Apriori アルゴリズムよりも約 3〜7 倍高速化が期待できると主張している．完全準同型暗号を用いた秘匿 Apriori 計算システムでは，候補アイテムセットごとでの暗号化したデータ同士の計算時間が長いことが課題であった．そのため計算対象のアイテムセットに対応する計算を削減する FUP アルゴリズムを適用することによって，計算時間が大幅に削減されたと考えられる．

7.4 秘匿 **FUP** 計算の分散処理システムの高速化率評価

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 (6) に基づいた Amdahl の法則による並列度と高速化率の関係 [21] と共に図 6 に示す．

$$\text{高速化率} \leq \frac{1}{(1 - \text{並列実行時間の割合}) + \frac{\text{並列実行時間の割合}}{\text{ワーカ数}}} \quad (6)$$

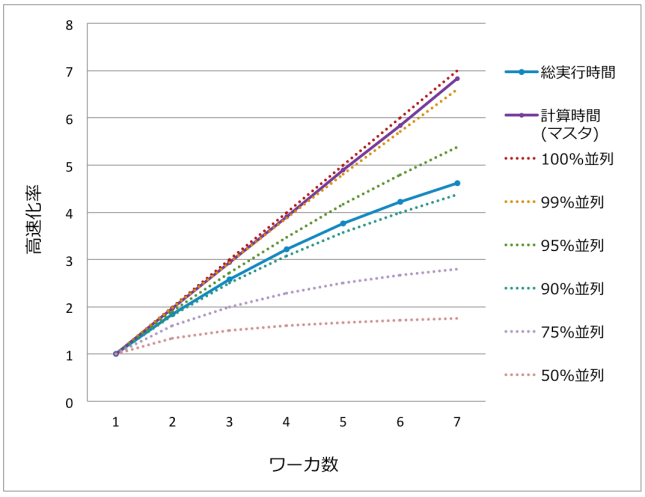

図 **6** ワーカ数ごとの FUP 計算の高速化率と Amdahl の法則による並列度に対する高速化率

マスタ側の完全準同型暗号の暗号文同士の計算時間はほぼ 99%～100%並列時に近い高速化率で分割されていることが示された．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%並列時の高速化率に近い．したがって本システムは式 (6) より，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．

8. 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計算の高速化を目指し，データベース更新時の Apriori アルゴリズムの高速化を行う FUP アルゴリズムを用いた秘匿データマイニングシステムを実装した．またマスタ・ワーカ型分散処理を適用し，システムの高速化を行った．分散処理方法にアイテムセットごとの分割を適用した．

その結果，データベース更新時において FUP アルゴリズムによる再計算は Apriori アルゴリズムによる再計算と比較して，約 3～4 倍の計算時間の短縮が可能になった．また分散処理化によって，マスタ側の計算時間が分散台数に応じて減少した．分散処理効果に関しては，Amdahl の法則に基づく並列度と高速化率の関係から，秘匿 FUP 計算の分散処理システムはワーカ数を最大限に増やす場合，最大で約 10 倍程度の高速化率が期待できると考えられる．

今後は本手法のクラウドコンピューティングを想定した環境での実験を行い，秘匿 FUP 計算システムの更新前データベースと更新部分のデータベースに対するそれぞれの演算を別ワーカで役割分担を行う分散処理システムを提案し，データベース更新時の完全準同型暗号を用いたアソシエーション分析のさらなる効率化を目指したい．

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] Qiankun Zhao and Sourav S. Bhowmick. Association rule mining a survey. *Technical Report, CAIS, Nanyang Technological University, Singapore*, No. 2003116, p. 1, 2003.

[2] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering*, Vol. 8, No. 6, pp. 866–883, 1996.

[3] 佐藤宏樹, 馬屋原昂, 石巻優, 今林広樹, 山名早人. 完全準同型暗号のデータマイニングへの利用に関する研究動向. 第 15 回情報科学技術フォーラム F-002, 2016.

[4] 高橋卓巳, 石巻優, 山名早人. SV パッキングによる完全準同型暗号を用いた安全な委託 apriori 高速化. 第 15 回情報科学技術フォーラム F-002, 2016.

[5] 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.

[6] 今林広樹, 石巻優, 馬屋原昂, 佐藤宏樹, 山名早人. 完全準同型暗号による安全頻出パターンマイニング計算量効率化. 情報処理学会論文誌データベース (TOD), Vol. 10, No. 1, pp. 1–12, 2017.

[7] H. Imabayashi, Y. Ishimaki, A. Umayabara, and H. Yamana. Fast and space-efficient secure frequent pattern mining by FHE. In *2016 IEEE International Conference on Big Data (Big Data)*, pp. 3983–3985, Dec 2016.

[8] D. W. Cheung, Jiawei Han, V. T. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: an incremental updating technique. In *Proceedings of the Twelfth International Conference on Data Engineering*, pp. 106–114, Feb 1996.

[9] 安田雅哉. 完全準同形暗号の応用 (小特集 完全準同形暗号の研究動向). 電子情報通信学会誌, Vol. 99, No. 12, pp. 1167–1175, 2016.

[10] Craig Gentry, et al. Fully homomorphic encryption using ideal lattices. In *STOC*, Vol. 9, pp. 169–178, 2009.

[11] Tibouchi Mehdi. 整数上完全準同型暗号の研究 (特集クラウドビジネスを支えるセキュリティ基盤技術). NTT 技術ジャーナル, Vol. 26, No. 3, pp. 71–75, 2014.

[12] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, Vol. 22, pp. 207–216. ACM, 1993.

[13] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215, pp. 487–499, 1994.

[14] David Wai-Lok Cheung, Vincent TY Ng, and Benjamin W Tam. Maintenance of discovered knowledge: A case in multi-level association rules. In *KDD*, Vol. 96, pp. 307–310, 1996.

[15] 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.

[16] Shoup V. and Halevi S. HElib. `http://shaih.github.io/HElib/index.html`. Accessed: 2017-1.

[17] Shai Halevi and Victor Shoup. Algorithms in HElib. Cryptology ePrint Archive, Report 2014/106, 2014. `http://eprint.iacr.org/2014/106`.

[18] Peter S Pacheco. *Parallel programming with MPI*. Morgan Kaufmann, 1997.

[19] Open MPI. `https://www.open-mpi.org/`. Accessed: 2017-1.

[20] Boost. `http://www.boost.org/`. Accessed: 2017-1.

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