完全準同型暗号を用いたFP-growthによる頻出パターンマイニングの分散環境への実装

種村 真由子　(指導教員：小口 正人)

1　はじめに

クラウドサービスが発展し，大量のデータをクラウド上で処理する機会が増えている．その中で，医療データ等の大規模な機密データをクラウド等に外部委託して処理する場合には，送信する情報の漏えい対策が重要となる．完全準同型暗号（FHE）は，暗号文同士の加算と乗算が成立する暗号であり [1]，これを利用することで委託先サーバに平文データを渡さずに処理を行うことが可能になる．FHE の応用例として，Apriori アルゴリズムによるトランザクションデータの頻出パターンマイニングを行った Liu らの P3CC や，その高速化を行った先行研究の [2][3][4] などがある．本研究では，FHE を使用し，同様にパターンマイニングを行う他アルゴリズムの FP-growth による頻出パターンマイニングの実装に取り組む．

2　完全準同型暗号

完全準同型暗号は，公開鍵暗号方式の機能を持ち，暗号文同士の加算，乗算が成立する暗号である．2009 年に Gentry が実現手法を提案した [1]．各暗号文は，暗号の解読不可能性を高めるため，ノイズパラメータという値が付加されている．この値は，暗号文同士で計算を行うたびに増加し，特に乗算を行う際に大きく増加する．ノイズパラメータの値は，閾値に対して十分に小さい必要があり，閾値を超えると復号が不可能となる．また，暗号の構造上比較演算を行うことは困難である．復号せずとも計算を行えるという特長により，活用が期待されているが，一般に処理の計算量が大きいことから，現在実用化に向けての研究が盛んに行われている．

3　頻出パターンマイニング

3.1　頻出パターンマイニングと Apriori

頻出パターンマイニングは，データマイニングの一種であり，大量のトランザクションデータの中から，頻出であるアイテムの組み合わせや条件を取り出す手法である．応用例として，購買データから頻繁に購入される商品の組み合わせを抽出するマーケットバスケット分析が挙げられる．代表的なアルゴリズムでは Apriori がある．Apriori は，アイテム数 1 から小さい順に頻出アイテムの組み合わせを挙げていく方式である．

3.2　FP-growth

FP-growth は，本研究で用いる頻出パターンマイニングのアルゴリズムである．先述の Apriori に対して，深さ優先探索のような形で頻出パターンの抽出を行う．FP-tree という頻出アイテムが格納されたデータ構造を再帰的に走査することで，頻出パターンを求める．FP-growth アルゴリズムの概要は以下である [5]．

・FP-tree の構築

FP-tree は，トライ木に類似したデータ構造である．トランザクションに含まれるアイテムで，出現頻度が閾値を超えたものを格納する．アイテムは頻出なものからルートに近い位置に格納され，各ノードはアイテムの種類と出現頻度を保持する．

・FP-tree の走査

作成した FP-tree において，最も出現頻度の低いアイテム（A とする）を含む枝から順に走査する．A に至るまでのアイテムと頻度を格納したリストから条件付き FP-tree を作成し，走査する．tree に分岐が存在する場合には，さらにその中で最も出現頻度の低いアイテムに対し FP-tree を構築し，その tree を再帰的に走査する．分岐が存在しない場合，現在の tree に含まれるアイテムの組と条件付けに使用したアイテムを頻出パターンとして保存する．これを tree 全体の走査が完了するまで繰り返す．

4　提案システム

本研究では，以下のようなシステムを提案する．

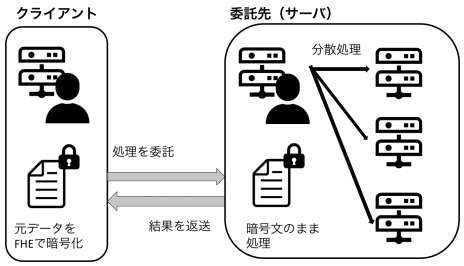

図 1: システムの概要

図 1 のシステムにおいて，クライアント側で暗号化したトランザクションデータをサーバに送信し，FP-growth を使用した頻出パターンマイニングにおける，アイテムのサポート値計算部分を委託する．サーバ側はマスタ・ワーカ型の分散・並列処理を行う．実装は C++で行った．分散・並列処理のライブラリは MPI を使用する．

処理の大まかな流れは以下である．

1. クライアントでのデータの準備を行う．元データを FHE で暗号化し，アイテムの候補とともにサーバに送信する．
2. サーバでの委託処理を行う．クライアントから暗号化されたデータを受信し，このときマスタ･ワーカ型の分散処理によりアイテムのサポート値を計算する．その後クライアントに結果を返送する．
3. クライアントで FP-growth を行う．サーバから受信したファイルからアイテム数を絞ったトランザクションを再生成し，FP-tree を構築する．構築した FP-tree を走査し，結果を出力する．

5 実験

5.1 実験概要

同一ネットワーク内の 2 台のマシンを使用し，各マシン上でクライアントプログラムとサーバプログラムを実行した．実行時間とマシンのリソースを，クライアント側プログラムにおいて測定した．リソースに関しては，CPU 使用率，メモリ使用量，ネットワークで送信・受信したデータ量を測定した．

実験で用いたマシンの性能は，OS は CentOS6.9，CPU は Intel Xeon E5-2643 v3，コア数 6，スレッド数 12，メモリが 512GB である．

実験のためのデータセットの生成は，IBM Quest Synthetic Data Generator で行った．データ作成時のパラメタは，アイテム数 10, 20, 30 の場合に対して，それぞれトランザクション数を 3300，6600，9900 に設定した．アイテム数とトランザクション数のみを変化させ，平均アイテム長と最大パターンの大きさは 5 に固定した．最小サポート値は 0.1 としている．また，この実験において，サーバ側のワーカ数はすべて 1 としている．

5.2 実験結果

実行時間の測定を 5 回行った平均値を図 2 に示す．図中ではサポート値計算，FP-tree 構築，FP-tree の走査，その他の処理の実行時間それぞれ測定したものを示している「その他」の時間には，データの暗号化と復号の処理時間，サーバとの通信時間が含まれる．

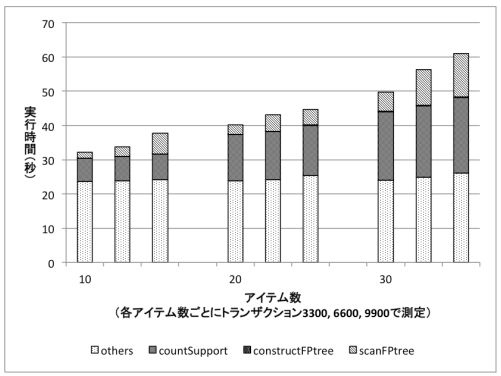

図 2: クライアントの実行時間測定結果

FP-tree の構築時間は，全体の処理時間に対して非常に短いことがわかる．サポート値計算時間はアイテム数に比例している．また，FP-tree を走査する時間はアイテム数，トランザクション数の両方の影響を受けて指数関数的に増加することが分かる．その他の処理はアイテム数やトランザクション数ではほぼ変化しない．

また，図 3 には，アイテム数 30，トランザクション数 9900 で実行した際のクライアントプログラムを実行したときのマシンの CPU 使用率とメモリ使用量を示した．

時刻 17 秒ごろ CPU 使用率のピークの直後にメモリ使用量が大きく上昇し，時刻 53 秒ごろからメモリ使用量と CPU 使用率が再び上昇している．また，時刻ごとのデータ通信量と比較したところ，初めの CPU 使用率のピークはサーバへの送信量のピーク時とほぼ一

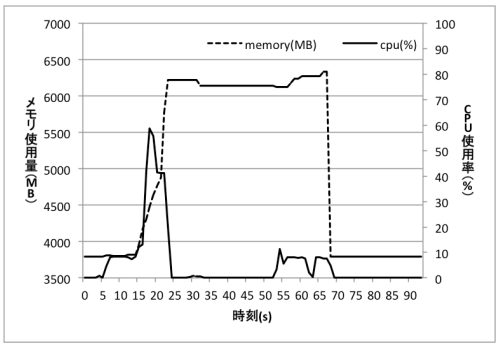

図 3: クライアントの CPU 使用率とメモリ使用量

致していることがわかった．また，後半で CPU 使用率が約 10%上昇している部分は，サーバからのデータの受信と，FP-growth の処理により起こっていると考えられる．その際，同時にメモリ使用率が階段状に上昇しているが，これは FP-growth で再帰的に tree を構築することによりメモリを消費しているものと考えられる．

6 まとめと今後の課題

完全準同型暗号を使用した FP-growth アルゴリズムによる頻出パターンマイニングのシステムを実装し，クライアントのマシンで実行時間と使用リソースの測定を行った．その結果，FP-growth の処理時間は，特にアイテム数の増加の影響で増加しやすいことがわかった．今後は，さらに大きなサイズの入力データに耐えうる処理方法の検討が必要である．また，FP-growth における tree 走査部分の処理を部分的にサーバへ委託することで，クライアント側で行う処理の削減に取り組んでいく．今回の実験は同一ネットワーク内のみで行ったが，外部のネットワークからサーバに接続する実験の実施を考えている．

謝辞

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] C. Gentry: A Fully Homomorphic Encryption Scheme. Doctoral dissertation. pp.2-8. September 2009.

[2] 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. DEIM Forum 2016 F8-6. 2016.

[3] 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. 2017.

[4] 山本百合, 小口正人: 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. DICOMO2018. 2018.

[5] J. Han, J. Pei, and Y. Yin, ”Mining Frequent Patterns without Candidate Generation,” 2000