完全準同型暗号を用いたゲノム秘匿検索の高速化手法

山田優輝　(指導教員：小口正人)

1　はじめに

近年ヒトゲノムの解析と応用が進み，様々な分野でゲノムデータ利用の実用化が注目されるようになった．ゲノムデータの活用には大型の計算機とストレージが必要になるため，利用者の問い合わせを受けてクラウド上で演算を行う委託システムが今後普及していくと考えられる．しかしクラウドは安全性が低く，またゲノム情報は変更できない重要な個人情報であるため，プライバシ保護のための暗号化処理が必須となる．

特にバイオインフォマティクスの研究において頻繁に行われる検索演算についてこのシステムを実現するために，先行研究 [1]-[2] では暗号化されたデータ同士での演算が可能な完全準同型暗号（以下 FHE: Fully Homomorphic Encryption）を用いる秘匿検索手法を提案しその高速化に取り組んでいるが，依然としてサーバ側で行われる秘匿検索演算の計算量が課題となっている．そこで本研究では，先行研究のサーバ側で行われる FHE 演算にデータベースの分割による分散処理を適用し，実用に向けた高速化を目指す．

2　先行研究

FHE は暗号文同士の加算と乗算を両方可能とするが，演算のたびに暗号文に含まれるランダムなノイズが増加し，閾値を越えると復号できなくなるという問題点も持っている．これに対しては，演算回数の限定または bootstrapping と呼ばれるノイズ削減手法の導入，という二つの解決手法が考えられるが，石巻ら（2016）は従来の FHE を用いたゲノム秘匿検索手法に bootstrapping を導入し演算の最適化を行うことで計算量の削減を行った [1]．石巻らのシステムはサーバとクライアントが 1:1 で問い合わせを行うもので，サーバはクライアントから検索したい文字列を暗号化処理したものとその文字列を検索したい配列上の位置（以下 ポジション）を受け取り，秘匿検索を行ってマッチしたか否かの結果を返す．また，山本ら（2017）は従来のゲノム秘匿検索手法に分散処理を導入することによる高速化を試みている [2]．山本らのシステムでは問い合わせごとの演算回数が限られるため，サーバは一文字のクエリを受け検索を行う．クライアントはサーバから受け取った演算結果同士を比較して最終的な結果を得る．このサーバ側での演算に対してデータベースの分割によるマスタ・ワーカ型の分散処理を適用することで，高速化が行われている．

また，これらの先行研究 [1]-[2] ではゲノム配列データベースを離散データ構造 Positional-Burrows Wheeler Transform (PBWT)[3] の形に変換している．これはゲノムデータに対して列ごとのソートを行ったもので，クエリとデータベースに含まれるサンプルとのマッチング判定の計算量を大幅に削減することを可能としている．また，クライアントがダミーを含む複数のポジションを指定することで，実際に利用するポジションを秘匿する等の秘匿性向上のための工夫も為されている．

3　提案手法

3.1　提案システム概要

本研究では先行研究に基づき，bootstrapping を導入した FHE によるゲノム秘匿検索のマスタ・ワーカ型の分散システムを以下の通り提案する．図 1 に提案システムの概要を示す．

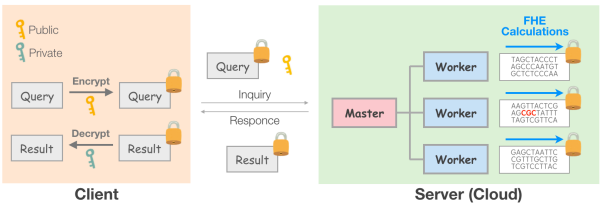

図 1: 提案手法概観

(1) クライアントはクエリ全文を暗号化し，公開鍵などと共にマスタへ送信する．

(2) マスタは受け取ったデータを各ワーカに転送する．

(3) 各ワーカは FHE を用いた秘匿検索演算を行う．また，bootstrapping によるノイズのリセットを適宜行う．

(4) 秘匿検索演算終了後，各ワーカは結果をマスタへ転送する．

(5) マスタは結果を収集し，クライアントへ結果を送信する．

(6) クライアントは復号を行い，問い合わせの結果を得る．

以上のプロトコルを C++ で実装した．

また，完全準同型暗号の演算にはオープンソースのライブラリである HElib[4] を，分散処理適用時における各マシンの制御のための MPI (Message Passing Interface) を利用するライブラリとして Open MPI[5] を用いた．

3.2　分散方法

検索アプリケーションの分散化手法としては，データベースの分割による分散処理，独立性の高い計算部位に適用する分散処理，独立性の高い手順に適用する分散処理，などが挙げられる．山本らによる先行研究 [2] では各ワーカマシンに分割したデータベースを設置することで，同時により多くのクエリとデータベース間のマッチング有無を調べることが可能となっている．また石巻らによる先行研究 [1] のプロトコルでは直前の演算結果を用いた演算を繰り返し行うため，計算部位や処理手順へ適用する分散処理では分散による効果を期待することが出来ない．したがって本研究の提案システムにおいても，データベースの分割による分散処理を適用した．

4 実験

4.1 実験環境

同スペックのマシンを 4 台用いて実験を行った．各マシンの環境は，Intel®Xeon®Processor E5-2643 v3 3.4GHz，6 コア 12 スレッド，メモリ容量 512GB，ストレージとして RAID0 の SSD が 480GB，HDD が 2TB である．各マシンで最大 2 スロットずつのワーカを稼働させ，そのうち 1 スロットにマスタの機能を持たせた．

実験に使用するゲノムデータとして 1 サンプルあたり 10,000 文字のデータを 200 サンプル用意し，検索クエリは長さ 5 文字のものを用いた．さらに 2 節で述べたようにダミーの検索開始ポジションを加え，1 クエリにつきデータベースの最大 50 箇所を始点とした文字列検索を行った．

4.2 実験結果

クエリとデータベースとの間でマッチングの有無を判定するプログラムを用いた実験を 3 回ずつ行った際の，ワーカ数ごとのマスタにおける平均実行時間とポジション数におけるワーカ数ごとのマスタの平均秘匿演算実行時間とのグラフをそれぞれ図 2，図 3 に示す．

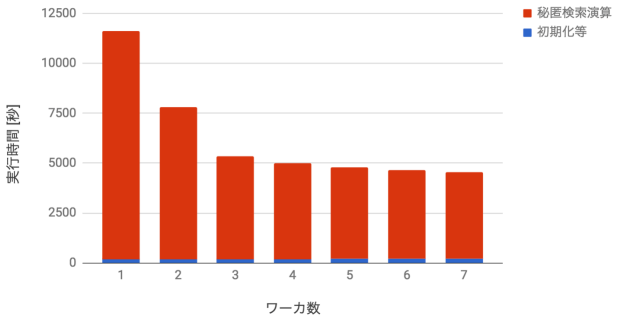

図 2: ワーカ数ごとのマスタにおける平均実行時間 (秒)

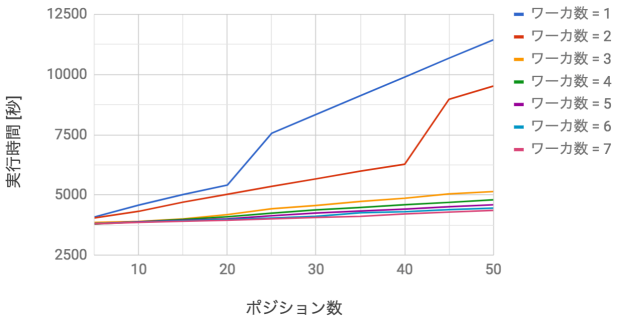

図 3: ポジション数におけるワーカ数ごとのマスタの平均秘匿演算実行時間 (秒)

図 2 よりマスタでの実行時間において，ワーカ数の増加とともに計算時間が減少することが確認された．クライアントとの通信やワーカ間でのファイル共有等を含む初期化に要する時間はワーカ数に応じて増加するが，秘匿検索演算の実行時間と比較するとオーバーヘッドは十分に小さい．また，ワーカ数が増えるに連れて，分散処理の導入による計算時間の減少効果は徐々に横ばいになっている．これは計算手順の中に，計算量がデータベースの大きさに依存しない処理が含まれているためである．

また図 3 より，ポジション数が増加すると，分散による効果が大きくなることが分かる．2 節で述べたように先行研究は PBWT 構造のデータベースを用いており，これによりクエリの検索はデータベースを全て検索するのではなく，クエリの長さの分のみデータベースと演算することで最長マッチ数を算出する事が出来る．このためポジション数が少ない実験では前処理等の並列分散化できない演算の計算の影響が並列分散化可能な演算よりも大きくなり，分散効率が抑えられてしまうが，ポジション数を増やすとデータベース上の検索する箇所が多くなるため，データベースによる分散化の効果が現れやすくなったと考えられる．

5 まとめと今後の課題

実験により，bootstrapping を導入したゲノム秘匿検索のサーバ側での処理にマスタ・ワーカ型の分散処理を適用すると，分散台数に応じて計算時間が減少することが示された．

今後は実装の改善に取り組むとともに，頻繁に行われる操作に特化した高速化処理なども検討していきたい．

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及び工学院大学山口研究室の皆様に感謝いたします．

特に早稲田大学山名研究室所属の石巻さん及びお茶の水女子大学小口研究室所属の山本さんからは，システムのプログラムと多くの助言を賜りました．深く感謝いたします．

また本研究は，JST CREST JPMJCR1505 の支援を受けております．

参考文献

[1] Y. Ishimaki, H. Imabayashi, K. Shimizu and H. Yamana : ”Privacy-preserving string search for genome sequences with FHE bootstrapping optimization,” 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, 2016, pp. 3989-3991.

[2] Y. Yamamoto and M. Oguchi : ”A Decentralized System of Genome Secret Search Implemented with Fully Homomorphic Encryption,” 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, 2017, pp. 1-6.

[3] R. Durbin : ”Efficient haplotype matching and storage using the Positional Burrows-Wheeler Transform (PBWT),”Bioin- formatics, vol. 30, no. 9, pp. 1266-1272, 2014.

[4] HElib : http://shaih.github.io/HElib/, 2017 年 12 月閲覧

[5] Open MPI : https://www.open-mpi.org/, 2017 年 12 月閲覧