4X-06

SSDのセキュアな活用に向けた暗号化アプリケーション実行時のボトルネック調査

廣江　彩乃†　圓戸　辰郎††　小口　正人†

† お茶の水女子大学
†† キオクシア株式会社

1 はじめに

近年ゲノムデータなどの秘匿情報を活用する取り組みが増えている．これらのデータの処理を外部のサーバに委託する場合，セキュリティの観点から，完全準同型暗号を用いるなどして暗号化したまま処理することができることが望ましい．しかしこの場合，暗号化処理の計算量が多くなるため，実用上に向かない計算時間がかかってしまう．暗号化の際の計算量に加えて，ゲノムデータなどの大きなデータを扱う処理でメインメモリが不足してストレージへのアクセスが発生することも，実行時間が長くなる要因である．ストレージへのアクセス速度はメインメモリへのアクセスに比べて格段に遅いためである．また，完全準同型暗号方式を用いると暗号化データが元の数万倍のデータ量になるため，大概メインメモリが不足するが，メモリが高価であることを考慮すると，実行時間の課題に加えてコストの面にも課題がある．

そこで，実行効率とコストの課題を解決するため，近年高速化に向けて研究開発が進む，高性能で比較的安価な SSD の有効利用を検討する．本研究では，完全準同型暗号を用いたゲノムの秘匿検索アプリケーション [2] を用いて，暗号化アプリケーションの実行時負荷を計測・検証する．

2 先行研究

石巻らの (2016) 先行研究によるゲノム秘匿検索アプリケーション [2] を，本研究のメモリ性能評価に用いる．図 1 にアプリケーションの流れを示す．これ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ある．また，ゲノムデータは A,C,G,T の四文字から成る配列であるため，文字列検索と見なせる．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検索開始地点を受け取り，秘匿検索を行ってマッチしたか否かの結果を返す．クライアントから送られる暗号化された文字列をクエリ，文字列検索開始地点をポジションと呼ぶ．

このアプリケーションで用いるゲノム配列のデータベースは，検索の高速化のため，離散データ構造である Positional-Burrows Wheeler Transform (PBWT) [1] 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，計算量を大幅に削減することが出来る．また，クライアントがサーバに，ダミーを含めた複数の検索開始ポジションを伝えることで，実際に利用するポジションを秘匿することが出来る等，秘匿性向上のための工夫も為されている．

Bottleneck's study of a cryptographic application handling data on SSDs
† Ayano Hiroe　‡ Tatsuro Endo　† Masato Oguchi
† Ochanomizu University　‡ KIOXIA Ltd.

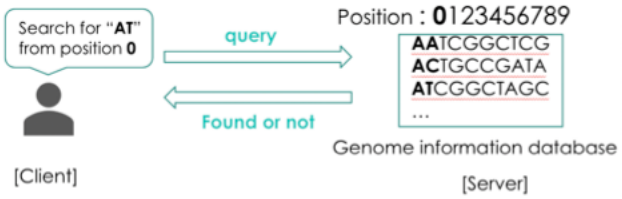

図 1　アプリケーションの流れ

3 実　験

3.1 実験概要

本研究では上で述べたゲノム秘匿検索アプリケーションを用いて，コンピュータリソースへの負荷やメモリの使用量を計測する．そして，実行環境の差による分析を行う．

プログラム実行に必要なメインメモリの量が容量を超える際には，ストレージに領域を確保する必要がある．この場合において，メインメモリに比べると圧倒的に遅いストレージへのアクセスが，大きなデータを扱う暗号化アプリケーションの実行時間にどの程度影響するか調べたい．ストレージの性能を見る上で，まず swap 処理に着目する．実験の都合上小さいテストデータを用いる本研究では，Docker コンテナを用いて，使用可能メモリが不足する状況を作る．そして，コンテナが使用可能なメインメモリを制限して，メインメモリの外の swap 領域へのアクセスを発生させる．さらに，その swap 先メモリに高性能 SSD を指定して，アクセス速度の差を検証していく．

表 1　サーバ

CPU	Intel®Xeon®Processor 6 Cores × 2 Sockets
DRAM	DDR4 512GB 2133MT/s
HDD	HGST SATA 2TB

表 2　Docker コンテナ情報

	コンテナ 1	コンテナ 2
memory	約 500GB	1GB
swap memory	なし	9GB

用いたサーバのスペックは表 1 に示した．このサーバ上に割り当てメモリが異なる二種類の Docker コンテナを構築した．各 Docker コンテナの使用可能メモリを表 2 に記した．使用可能なメモリの量を制限していないものがコンテナ 1，swap メモリに 9G，非 swap メモリに 1G を割り当てたのがコンテナ 2 である．

3.2 高性能 SSD の比較

swap デバイスに高性能な SSD を用いた場合の性能評価を行っていく．比較対象とする高性能 SSD は 2 種類である．そ

れらの性能をまとめたのが表 3 である．

表 3　比較対象デバイス

	KIOXIA EXCERIA PLUS SSD	Samsung 980 PRO
capacity	1TB	500GB
sequential read	3,400MB/s	6,900MB/s
sequential write	3,200MB/s	5,000MB/s
ramdom read (IOPS)	680,000 (4KiB,QD32)	800,000 (4KB,QD32)
ramdom write (IOPS)	620,000 (4KiB,QD32)	1,000,000 (4KB,QD32)

表 3 の 2 種類の SSD を swap 先として指定して実行する他に，実行に十分なメモリを Docker コンテナに割り当てることで swap 処理を発生させない条件と，メモリが不足する状況を模倣して HDD に swap 領域を作成させるという条件を加えて，以下の 4 種類のケースで計測していく．使用するコンテナ 1,2 へのメモリ割当は表 2 に示す通りである．

（1） DRAM 上で処理が完結する場合 (コンテナ 1 使用)
（2） swap 処理により，HDD へのアクセスが発生する場合 (コンテナ 2 使用)
（3） swap 処理により，Kioxia SSD へのアクセスが発生する場合 (コンテナ 2 使用)
（4） swap 処理により，Samsung SSD へのアクセスが発生する場合 (コンテナ 2 使用)

3.3　アプリケーション実行条件

実験に使用するゲノムデータは 1 サンプルあたり 10,000 文字のデータを 1,000 サンプル用意した．これは，ゲノムデータ [3] をプログラム [4] を用いて PBWT の形に変換したものである．パラメータであるゲノムデータ長であるクエリ長とゲノムデータ開始位置を表すポジション数は，一般的な塩基配列の長さや実行時間・計算機にかかる負荷を考慮して，それぞれ 4,3 とした．また，実行に際してはノイズのリセット機能である bootstrap 処理を用いている．なお，演算内容は毎回同じで，キャッシュのクリアも行った．

3.4　実験結果及び分析

ゲノム秘匿検索アプリケーションを DRAM 上のみで実行を終わらせた場合と，使用可能メモリを制限して swap の状況を作った場合で負荷を取得し，比較した．3.2 章における，4 つの条件で実験を行った．条件 (1)-(4) のメモリの使用量と，条件 (2)-(4) で発生した swap の状況をそれぞれ図 2-図 5，図 6 に示す．メモリ使用量として，server と client 各々のプロセスの top コマンドの RES 項目を抽出し，swap 発生状況として，vmstat コマンドの swap in/out の値を抽出した．コマンドは 10 秒おきに実行した．

実験結果から分かったことが 3 点ある．まず，用いる SSD のみを変えた条件 (3),(4) について，ここで示す図や実行時間を比べても，今回用いた 2 種類の SSD の間には実行効率の差はほどんどないと分かる．次に，条件 (2) と条件 (3),(4) を比べると，swap 領域に HDD を用いる場合と高性能 SSD を用いる場合を比較することができるが，SSD を用いる方が圧倒的に実行時間が短く，効率が良いことが分かった．よって，HDD を用いると大きな遅延となるため，メインメモリ不足時に HDD を用いることは現実的ではないと言える．また，条件 (1) と条件 (3),(4) の実行時間を比べると，実行時にメインメモリのみを用いた場合と，swap 領域として SSD を使用する場合の効率を比較することができる．メインメモリと SSD へのアクセス速度には大きな差があるが，今回の実行時間を比べると，ハードウェアほどの性能差がないと分かる．このことから，実行時間全体のボトルネックはデータの Read/Write ではなく，演算処理が大きな影響を与えていると言える．

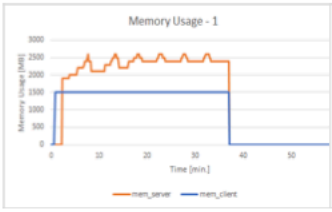

図 2　条件 (1) メモリ使用量

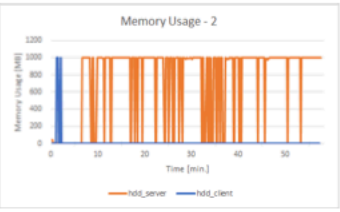

図 3　条件 (2) メモリ使用量

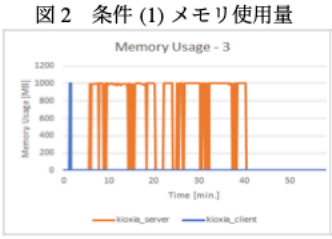

図 4　条件 (3) メモリ使用量

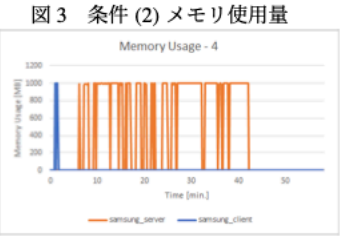

図 5　条件 (4) メモリ使用量

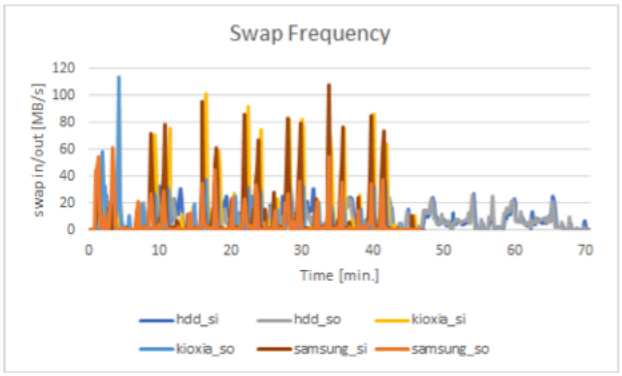

図 6　swap 発生状況

4　まとめと今後の課題

今回行った実験結果より，高性能 SSD を用いることで，swap 処理を高速化できることがわかった．今後，性能差がある SSD を用いるとパフォーマンスに違いがあるか，どのような性能を持つデバイスが暗号化アプリケーションに適しているかを調べていく．また，プロセスごとに実行結果を見ていくことで，アプリケーションのどのカーネル処理がボトルネックになっているのかを検証し，暗号化アプリケーション実用化に向けて研究を進めていく．

謝　　辞

本研究の一部は，キオクシア株式会社の支援を受けて実施したものである．

文　　献

[1] R. Durbin, ”Efficient haplotype matching and storage using the Positional Burrows-Wheeler Transform (PBWT), ” Bioin- formatics, vol. 30, no. 9, pp. 1266-1272, 2014
[2] Y. Ishimaki et al., “ Privacy-preserving string search for genome sequences with FHE bootstrapping optimization. ” 2016 IEEE International Conference on Big Data (Big Data). IEEE. 2016, pp. 3989–3991.
[3] International Genome Sample Resource, https://www.internationalgenome.org/data/
[4] Github, https://github.com/iskana/PBWT-sec/blob/master/sample_dat/gensmpl/cvSNPSeq2pbwt.cpp