DEIM2021 G25-5

# 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性能評価

廣江　彩乃† 　圓戸　辰郎†† 　小口　正人†

† お茶の水女子大学理学部情報科学科　〒 112-8618 東京都文京区大塚 2-1-1
†† キオクシア株式会社　〒 108-0023 東京都港区芝浦 3-1-21 田町ステーションタワー S
E-mail: †{ayano-h,oguchi}@ogl.is.ocha.ac.jp, ††tatsuro1.endo@kioxia.com

**あらまし**　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，計算量が多くなることから，実用上に向かない計算時間がかかってしまう．そこで，このように処理が多い暗号化アプリケーションの実行をハードの面から高速化することを考える．ゲノムデータなどの大きなデータを扱う処理では，DRAM が多く必要になるため，不足する可能性が十分にある．そこで，近年高速化に向けて研究開発が進む，高機能 SSD の有効利用を模索したい．そのために、まずは従来のメモリとストレージの環境において，暗号化アプリケーション実行時のリソースの使用状況を調べ，リソースアクセス時のボトルネックなどを評価する．

**キーワード**　暗号化, SSD, 性能評価

## 1　はじめに

近年，遺伝子情報などの秘匿データの解析技術が急速に進歩し，特に医療分野での活用が大変注目されている．2003 年には，人間のゲノム情報を全て解析するプロジェクトであるヒトゲノム解析計画 [1] が完了した．それ以降日本国内でも，2015 年に厚生労働省によるゲノム情報を用いた医療等の実用化推進タスクフォース [2] が編成されるなど，実用化に力を入れている．具体的な実用例としては，大量のゲノムデータをクラウドに保存して，そのデータと患者のゲノムパターンのマッチ判定を行うことで，病気の性質を持つ遺伝子塩基配列の有無を判定したり，薬の副作用を引き起こす特定の塩基配列の有無を判定したりすることが挙げられる．これらの可能性から，医療分野でのゲノム情報の活用による，医療診断や新薬開発の発展が大いに期待されている. [3]

ゲノム検索では大量のゲノムデータを扱う必要がある．しかし，各医療機関や研究機関内部の設備で大容量のデータ保存領域を確保したり，膨大な計算処理を行う計算機資源を確保することは困難である．そこで，クラウドに処理したいデータを預け，利用者の問い合わせを受けてクラウドで演算を行うというゲノムデータ委託システムが今後普及していくと考えられる．しかしクラウドはインターネットに接続され，不特定多数からアクセスを受ける可能性があり，情報漏洩のリスクがある．遺伝子情報は究極の個人情報であり，ゲノムデータベースの情報が漏洩することは絶対に防ぐ必要がある．その点で，サーバ側であるデータベース保持者は，例えそのシステムのクライアントであっても，データベースの中身の情報を不必要に渡すことは避けたい．また，ゲノムデータ委託システムを使ってゲノム検索を行うクライアント側としても，例えば研究開発に用いる場合，未発表の自身の研究内容をサーバ側に知られたくはない．よって，相互のプライバシを保護するための暗号化処理が必要である．

クライアント・サーバ方式を構築して，従来の共通鍵暗号方式を利用する場合、演算を行うサーバ側でデータを復号する必要があるため，この手法は適さない．よって，暗号化したまま演算や通信を行うことが望まれる．暗号化したまま演算を行うことができる暗号化方式としては，暗号文同士での加法演算が成立する加法準同型暗号があるが，複雑な演算が困難である．そこで先行研究では，完全準同型暗号を利用した秘匿検索手法が提案されている．完全準同型暗号 (以下 FHE : Fully Homomorphic Encryption) は，暗号化されたデータ同士の積を含む演算が可能な暗号方式であるため，加法準同型暗号よりも複雑な演算が可能である．

しかしこの手法には，コンピュータリソースへの負荷が大きすぎるという問題がある．離散データ構造 Positional-Burrows Wheeler Transform (PBWT) の利用や計算手順の最適化など，秘匿検索アルゴリズムの高速化が勧められているものの，依然としてサーバ側の秘匿計算の量が多く，メインメモリの使用量が多くなるため．そこで本研究では，近年開発が進む高速な SSD を活用することを考える．異なる性質を持つメモリを秘匿検索に用いて，その際のコンピュータリソースへの負荷や実行時間といった実行状況の比較・評価を行うことで，効率よく秘匿検索手法を助けるメモリの性質を検証する．

## 2　完全準同型暗号

### 2.1　特　　徴

式 (1) が示すように暗号文同士での加算が成立する性質を加法準同型性，また式 (2) のように暗号文同士での乗算が成立する性質を乗法準同型性という．

加法準同型性・乗法準同型性

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

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

完全準同型暗号 FHE はこの両方の性質を持ち合わせた暗号化手法である．FHE を用いることで，平文上で行うのと同様に暗号文同士での加法演算・乗法演算を行うことが出来る．完全準同型暗号は公開鍵暗号方式の機能を持つため，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，ゲノム秘匿検索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同士の処理を行えると期待できる [4].

### 2.2 暗号手法の課題

完全準同型暗号の概念自体は，公開鍵暗号が考案された当初の 1978 年に Rivest ら [5] によって提唱され，2009 年に Gentry [6] が実現手法を提案した．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 [7].

各暗号文には，暗号の解読困難性を高めるため，ランダムなノイズが付加されている．完全準同型暗号処理の課題として，計算量が大きいことに加えて，暗号文に含まれるノイズが演算の度に増加し，閾値を越えると復号することが不可能になるということが挙げられる．特に乗算を行った際のノイズの増加が著しいため，暗号文に対する乗算操作の演算回数を限定した制限付き準同型暗号， Somewhat Homomorphic Encryption (SHE，SwHE) が考案され，処理速度の改善に成功している．しかし制限付き準同型暗号では，演算回数が制限されるため，実装できる機能が限られて汎用性に欠ける. [8]

また，bootstrap と呼ばれるノイズをリセットする手法の導入を行うことで，演算回数が限られてしまうことは解決される．bootstrap 処理とは，暗号文のノイズを初期値に近い量に減少させ，暗号化した状態での計算を理論上何度でも可能にする手法である．しかし，実際にはこの処理も計算量が大きく，依然として計算量の大きさの問題は残る. [9]

### 2.3 完全準同型暗号ライブラリ

暗号スキームの研究が進むに連れて，多くのオープンソースの暗号ライブラリが幅広い用途に向けてオンラインで公開されるようになった．例えば HElib [10] は，初期に公開されたよく知られているライブラリの一つである．IBM の研究者らによって C++で実装され，ノイズをリセットする bootstrap をサポートしている．本研究でも HElib を用いて暗号化アプリケーションを実行する．

他にも，PALISADE [11] や SEAL [12] などがよく知られており，この二つもどちらも C++で実装されている．これらは HElib と違って bootstrap 処理が無かったり，HElib は外部のライブラリに依存しているのに対して，PALISADE や SEAL は外部ライブラリに依存していないという点が異なる点である．

## 3 ストレージデバイス

### 3.1 ストレージデバイスの現状

近年開発が進む記憶装置の一つとしてストレージクラスメモリや高性能 SSD がある．これらメモリの階層は図 1 が示す通りである [13].

高性能なメモリの研究開発が進む背景が以下だ.

5G の伸長などにあわせて，生成されるデータが爆発的に増えていき，生成されたデータは機械学習や AI といった様々なアプリケーションで CPU が処理するためにデータベース化されストレージに保存されている。現在のコンピューティングシステムでは，データを処理するためにデータが保存されているストレージ空間から CPU がデータを処理するためのメモリ空間へのデータの転送やデータの入れ替えが発生する．しかし，メモリ空間のデバイスである DRAM とストレージ空間のデバイスである フラッシュメモリを用いた SSD の間に大きな性能ギャップ

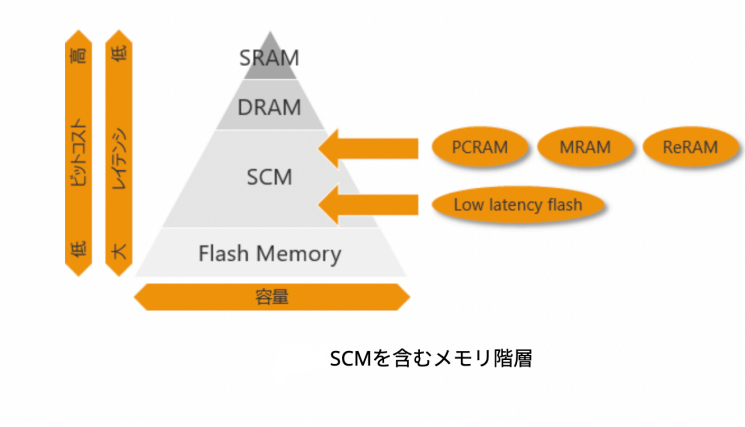


図 1　メモリ階層

があり，データの処理効率が良いとは言えない．そのため、データの入れ替えを少なくするために，主記憶容量の拡張と，データの転送を高速にするためのストレージ容量の低レイテンシ化が求められている．その DRAM とフラッシュメモリの間の性能・容量ギャップを埋めるために、SCM(ストレージクラスメモリ) と呼ばれる階層が考案され、様々なデバイス候補を各社が開発している．

### 3.2 ストレージクラスメモリ/低遅延 SSD

ストレージクラスメモリとは，メインメモリとストレージとの間の性能・コストの差を埋めるメモリの総称である．その特徴は，DRAM などのメモリよりも大容量で，SSD などのストレージよりも読み書き速度が速い不揮発性のメモリである，という点である．メインメモリのように処理が高速で、ビット単価が DRAM よりも安い不揮発性のメモリを作ろう，ということで考案された．メインメモリとストレージの長所を盛り込んだ形だ．最初にこの言葉が使われたのは，IBM が発表した論文 [14] である．その後，東芝メモリを前身にもつキオクシアや Intel がストレージクラスメモリの開発を進めている．ストレージクラスメモリは階層の名前であり，近年開発が進む低遅延 SSD や PCRAM(相変化メモリ) や MRAM(磁気抵抗変化メモリ) などの次世代不揮発性メモリを含む様々な記憶装置がこれにあたる．

低遅延 SSD は，低遅延フラッシュメモリとも呼ばれ，メモリに比べるとアクセスに時間がかかるフラッシュメモリを高速化したものである．低遅延 SSD は，ストレージクラスメモリに比べるとかなり製品化が進んでいる．キオクシアや Intel，Samsung などの各大手メーカが低遅延フラッシュとして，多層の 3D NAND フラッシュメモリ・チップを発表している．3D NAND チップとは，従来の 2DNAND チップがすでに限界まで高密度になっていることから開発された．3D NAND チップは，メモリセルを並べたレイヤーを積み上げることによって，構成されている [15]．本研究では，まずは高性能 SSD を用いて評価実験を行う．今回実験で用いる SSD についてはそれぞれの特徴を，5 章の実験環境にまとめる．

## 4 先 行 研 究

先行研究 [16] によるゲノム秘匿検索アプリケーションを，本研究のメモリ性能評価に用いるため，本章ではその概要を述べる. [17]

### 4.1 問題設定

ゲノム秘匿検索を適用するモデルについて述べる．サーバにデータベースを設置し，ゲノム配列データをサンプルごとに並べる．このゲノムデータはゲノム配列全体を用いるのではなく，個体差が現れやすい特定の位置の塩基を取り出した SNP（一塩基多型）を並べた SNP 配列を用いている．ゲノムデータは A, G, C, T の 4 種の塩基配列から構成されるため，ゲノム秘匿検索は 4 種に限定された文字列検索とみなす．サンプルそれぞれの長いゲノム配列データを行ごとに並べて二次元配列状のデータベースとすることで，各サンプルについての文字列検索を行うことが出来る．ゲノム秘匿検索システムはサーバ・クライアント形式で実装される．サーバはクライアントから一致判定を行いたいクエリ文字列を FHE により暗号化したものとゲノム配列上の検索開始点（以下 ポジション）を受け取ると，データベース上のデータと FHE 演算によるマッチング判定を行い，その結果を返す．この検索を，サーバとクライアントの双方のデータが秘匿された状態で，可能な限り高速に行うことが先行研究の目的である．

### 4.2 手法概要

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，その演算の最適化を行うことで，計算量の削減を行った [16]．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，先行研究 [16] ではゲノムデータの秘匿検索を高速に行うために，ゲノム配列のデータベースを離散データ構造である Positional-Burrows Wheeler Transform (PBWT) [18] 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．

### 4.3 ゲノム秘匿検索アプリケーション

具体的なアプリケーションの流れを以下に示す．ゲノム秘匿検索アプリケーションの目的は，ゲノムデータベースに対して問い合わせを行い，クエリとデータベース間でのマッチの有無について検索結果を得ることである [19]．クエリを生成するのをクライアント側，ゲノムデータベースを保持し，それとクエリとのマッチ判定を行って結果を送信するのをサーバ側として，クライアント・サーバ型のシステムとする．図 2 にアプリケーションの大まかな流れを示す．

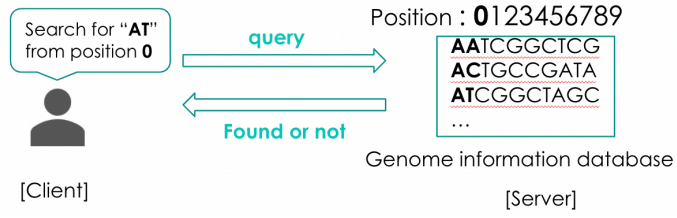


図 2　アプリケーションの流れ

細かいアプリケーションの処理の流れは以下である．

（1） クライアントは検索したい文字列とその文字列の検索開始位置の情報をクエリとして生成する．
そのクエリ全文を暗号化し，公開鍵などと共にサーバに送信する．
（2） サーバは FHE を用いた秘匿検索演算を行い，保持するデータベースとの一致判定を行う．
bootstrap 処理によるノイズのリセットを適宜行う．
（3） サーバはマッチしたか否かの結果をクライアントへ送信する．
（4） クライアントは自身の秘密鍵によって復号を行い，結果を得る．

データベース内でゲノム配列データをサンプルごとに並べることによって，各サンプルを特定のポジションから検索することが可能となる．この仕組みは，PBWT の手法を用いることで実現される．また，クライアントはダミーを含む複数のポジションを指定することで，実際に用いるポジションを秘匿することが出来る．このように，ゲノム検索を高速かつできるだけ秘匿に行う工夫が施されたアプリケーションである．実装は C++で行われている．また，完全準同型暗号計算には GitHub 上で公開されている HElib [10] を，分散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は，Open MPI [20] が用いられている．

## 5 実　　験

### 5.1 実験概要

使用するストレージデバイスによるコンピュータリソースへの負荷を調べるため，上で述べたゲノム秘匿検索アプリケーションを用いて，実行時間やメモリの使用量，swap 発生状況を計測した．

本研究で検証したいのは，メインメモリとストレージへのアクセス速度の差を埋めるために開発された高性能なメモリを用いると，暗号化アプリケーションなどの計算量の多い処理を，どの程度効率よく実行可能かということである．実際にゲノム秘匿検索を行う際には，扱うデータ量と演算量の大きさから，メインメモリ上では処理が完結せず，ストレージへのアクセスが発生すると考えられる．この際のストレージへのアクセス速度が，メインメモリに比べると圧倒的に遅くなるため，大きなデータを扱う暗号化アプリケーションの実時間内の実行が難しいということである．ストレージの性能を見る上で，swap 処理に着目した．実験の都合上，メインメモリの容量より小さいテストデータを用いる本研究では，故意に swap 処理を引き起こして，メモリの外の swap 領域へのアクセスを発生させる．そして，その swap 先領域となる記憶デバイスとして HDD や開発が進む各種高性能 SSD を用いて，それぞれの実行結果を検証していく．

### 5.2 実験環境

本研究では，サーバ上に Docker コンテナを構築し，そのコンテナ上でゲノム秘匿検索アプリケーションを実行する．サーバのスペックは，表 1 に示す通りである．本実験で Docker コンテナを用いる理由は，サーバなど実行環境に依存しないという特長に加えて，クラウドを利用する想定のもと，メインメモリ容量が不足する状況を再現して現実性を持たせるためである．

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．1 つは使用可能なメインメモリの量を制限していないコン

表 1　サーバのスペック

| | |
|---|---|
| CPU | Intel®Xeon®Processor E5-2643 v3 (3.4GHz) 6 Cores × 2 Sockets |
| DRAM | DDR4, 2133MT/s, 512GB |
| HDD | HGST, SATA, 2TB |

テナ，もう 1 つは swap メモリに 10GB,non-swap メモリに 1GB を割り当てたコンテナである [21]．それぞれをコンテナ 1，コンテナ 2 と呼ぶことにする．コンテナ 1 のメインメモリは 503.6GiB であり，サーバに備わるメインメモリのほとんどを使用できる．一方でコンテナ 2 は，使用可能なメインメモリの量に制限がかかっており，使えるメインメモリは 1GB である．

### 5.3　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 3 種類である．それらの性能をまとめたのが表 2 である．今回のアプリケーション実行に影響を及ぼすのが，SSD に用いられるメモリの種類である．3D XPoint を用いた SSD は，従来の NAND SSD に比べて速いと言われている．[22]

表 2　比較対象 SSD

| | KIOXIA EXCERIA PLUS SSD [23] | Samsung 980 PRO [24] | Intel OPTANE SSD 800P [25] |
|---|---|---|---|
| capacity | 1TB | 500GB | 118GB |
| memory type | NAND Flash | NAND Flash | 3D XPoint |

これら 3 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，表 3 に示す 5 種類のケースで計測していく．

表 3　実 験 条 件

| 条件 | 使用コンテナ | 使用する swap 領域 |
|---|---|---|
| 条件 (1) | コンテナ 1 | swap 処理なし |
| 条件 (2) | コンテナ 2 | HDD |
| 条件 (3) | コンテナ 2 | Kioxia EXCERIA PLUS SSD |
| 条件 (4) | コンテナ 2 | Samsung 980 PRO SSD |
| 条件 (5) | コンテナ 2 | Intel Optane SSD |

### 5.4　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．また，検索クエリは長さ 4 文字のものを用いた．さらに検索時の秘匿性を高めるため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 3 箇所を始点とした文字列検索を行った．また，ノイズのリセット機能である bootstrap 処理を用いている．なお，アプリケーションの挙動は毎度同じである．

### 5.5　実験結果と考察

5.3 節に述べた 5 つの条件の下で，実行時間・メモリ使用量・swap 処理発生状況を計測した．それぞれについて，実験結果と考察をまとめていく．

#### 5.5.1　実 行 時 間

まず，ゲノムデータ秘匿検索アプリケーションの実行終了までにかかった時間が，図 3 の通りである．データはそれぞれ 3 回の平均値を取ったものであり，実行のたびにキャッシュのクリアを行った．また，swap 領域を用いる条件 (3)-(5) では，実行終了後に swap 領域の無効化を行って，前の実行による影響が後に及ばないようにしている．

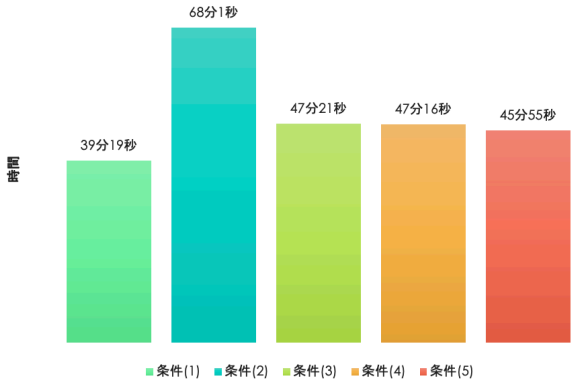


図 3　実 行 時 間

以上の結果から，実行時間に着目するとメインメモリ上で処理が完了する条件 (1) が最短であり，HDD を swap 領域として用いる条件 (2) が最長であることがわかる．条件 (2) と条件 (3)-(5) を比べると，swap 領域として HDD を使うか，高性能 SSD を用いるかによって，実行時間にかなりの差が出ている．このことから，高性能 SSD を用いることで，本暗号化アプリケーション swap 時の実行時間をかなり短縮できたと言える．

また，3 つの高性能 SSD に着目する．KIOXIA と SAMSUNG の SSD に比べて，Intel の SSD のみが実行時間が短くなっている．前者 2 つの SSD には，NAND Flash メモリが使用されている一方で，後者の Intel SSD には 3D XPoint メモリが搭載されている・3D XPoint のレイテンシが NAND Flash メモリよりも短い [26] ため，それが本実験の結果に反映されたと考えられる．

#### 5.5.2　メモリ使用量

次に，アプリケーションのメインメモリ使用量を計測した．この目的は，アプリケーション実行中に使用されるメモリの量の推移を見るためである．top コマンドを 10 秒毎に実行して，実際に使用した物理メモリの値を示す RES 項目を抽出した．今回実行しているアプリケーションはサーバ・クライアント構成であるため，サーバ側とクライアント側それぞれのメモリ使用量を調べた．図 5.5.2〜5.5.2 にメモリ使用量のグラフを示す．縦軸がメモリ使用量，横軸が時間を示しており，ピンク色の線がサーバ側で緑色の線がクライアント側の処理で使用されたメモリの量を表す．

この結果から，本アプリケーション実行時に，プロセスがどのようにメインメモリを使用するかが分かる．まず，今回 swap 処理を発生するために条件 (2)〜(5) で使用可能メモリを 1GB に制限してあるが，この結果から，条件 (1) の結果グラフより，本来であれば 1GB を超えるメインメモリが必要であると分かるので，実験の正当性が確認できる．

また，サーバとクライアントの各プロセスでのメモリの使い方も観測できた．このアプリケーションでは，まずクライアント側がクエリを生成して，そのクエリを受け取ったサーバ側がそれを暗号処理するため，このメモリ使用量のグラフの形は容易に理解できる．

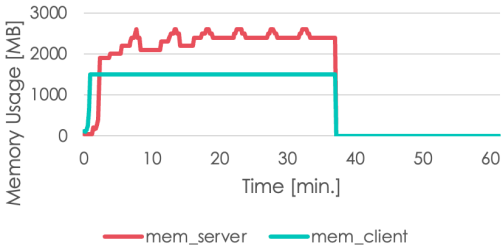


図 4　条件 (1) メモリ使用量

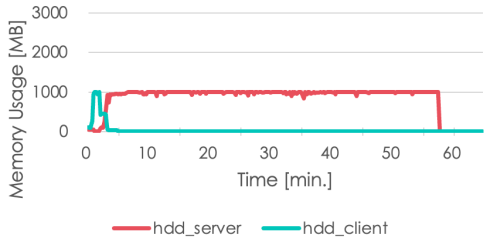


図 5　条件 (2) メモリ使用量

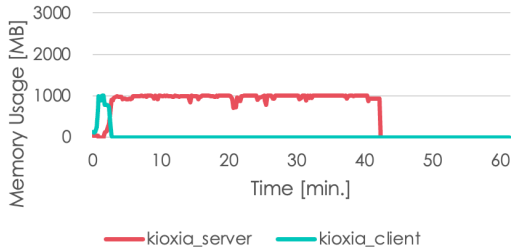


図 6　条件 (3) メモリ使用量

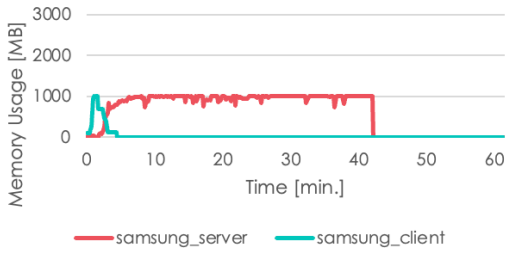


図 7　条件 (4) メモリ使用量

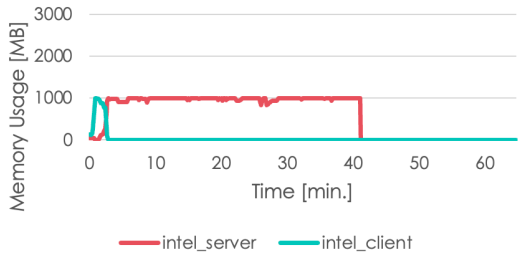


図 8　条件 (5) メモリ使用量

### 5.5.3　swap 発生状況

最後に，ストレージの性能評価として，swap の発生状況を計測した．この計測は，swap 処理が発生する条件 (2)〜(5) に対して行った．vmstat コマンド 10 秒毎に実行して，単位時間あたりの swap 領域の使用量として si(swap in)，so(swap out) の項目を抽出した．その結果をグラフにしたものが図 5.5.3〜5.5.3 である．縦軸が単位時間あたりの swap in/swap out のメモリの量を表し，横軸が時間を表している．

図 5.5.3 を見ると，高性能 SSD を用いた条件 (3)〜(5) は連続的に高い値を示しているのに対して，HDD を用いた条件 (2) は比較的低い値で推移していることが分かる．このことから，swap 量が少なくてリクエスト数が少なく，今回の実験で用いたアプリケーションの実行効率にはレイテンシが大きく影響するということが分かる．

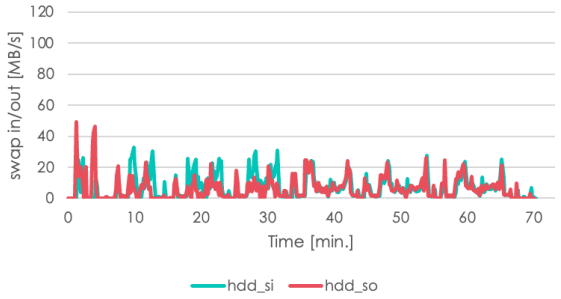


図 9　条件 (2)swap 発生状況

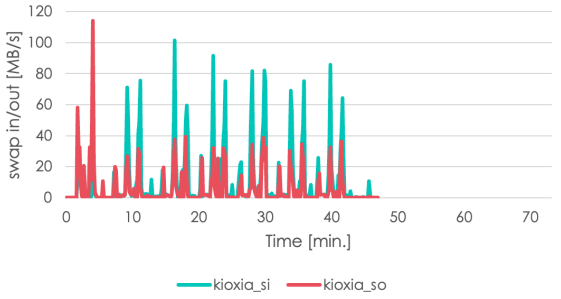


図 10　条件 (3)swap 発生状況

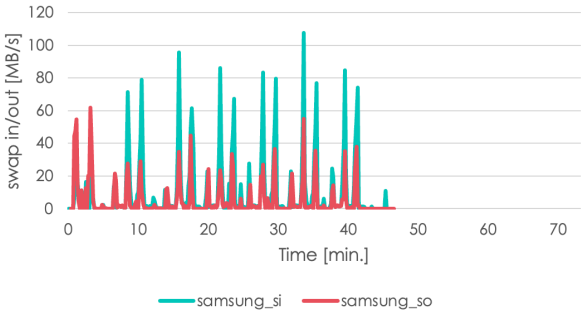


図 11　条件 (4)swap 発生状況

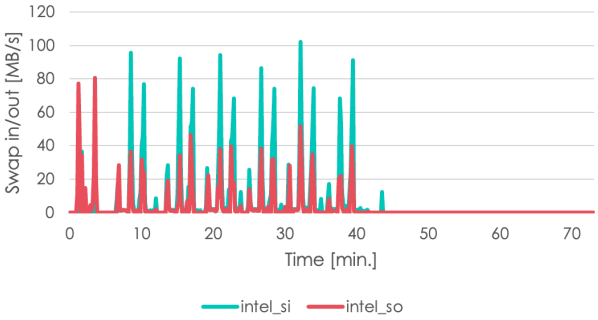


図 12　条件 (5)swap 発生状況

## 6 まとめと今後の課題

今回の実験により，レイテンシが小さい高性能 SSD を用いて，データの読み書き処理が多い暗号化アプリケーションで swap が発生した時の高速化ができた．そのため，高性能 SSD の本アプリケーションへの有効性を示すことができた．高性能 SSD は現在活発に研究開発が進められている．その一方で，これらの技術の実用性を具体的に示していくことが重要である．そのため，今後の研究において，開発が進む SSD の費用対効果や具体的な活用方法を検証していく．

## 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す．

### 文　　献

[1] ヒトゲノム解析センター，ヒトゲノム解析計画とは：http://hgc.jp/japanese/humangenome-j.html
[2] 厚生労働省，ゲノム情報を用いた医療等の実用化推進タスクフォース：http://www.mhlw.go.jp/stf/shingi/ other-kousei.html?tid=311652
[3] 中外製薬，ゲノム解析の将来とは？：https://www.chugai-pharm.co.jp/ptn/bio/genome/genomep11.html
[4] 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
[5] R. L. Rivest et al.,“On data banks and privacy homomorphisms,”Foundations of secure computation, vol. 4, no. 11, 1978, pp. 169 – 180
[6] Craig Gentry, et al.，Fully homomorphic encryption using ideal lattices. In STOC, Vol. 9, pp. 169–178, 2009
[7] 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
[8] nri，クラウドの普及を支える新たな暗号技術への期待 デジタルイノベーション 2, itf_201710_6.pdf
[9] 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
[10] Shoup V. and Halevi S., http://shaih.github.io/HElib/ index.html
[11] PALISADE, https://palisade-crypto.org/software-library/
[12] Microsoft SEAL, https://github.com/Microsoft/SEAL
[13] キオクシア株式会社，技術トピックス- DRAM 代替を目指した XL-FLASH™ によるデータベース性能比較実証 https://about.kioxia.com/ja-jp/rd/cutting-edge-researches/technology-topics/topics-20.html
[14] IBM Journal of Research and Development - ”Storage-class memory: The next storage system technology”, Volume: 52 , Issue: 4.5 , July 2008
[15] Intel，3D NAND が必要な理由 https://www.intel.co.jp/content/www/jp/ja/technology-provider/products-and-solutions/storage/why-3d-nand.html
[16] 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.
[17] 山田　優輝,『完全準同型暗号を用いた暗号化データベースにおける秘匿検索の分散処理による高速化』 http://www.is.ocha.ac.jp/ oguchi_lab/Publications/paper2017/ deim2018_yuki.pdf
[18] R. Durbin, ”Efficient haplotype matching and storage using the Positional Burrows-Wheeler Transform (PBWT),”Bioin- formatics, vol. 30, no. 9, pp. 1266-1272, 2014
[19] K. Shimizu, K. Nuida, and G. Ratsch,“Efficient privacy-preserving string search and an application in genomics.”Bioinformatics 32.11 (2016), pp. 1652– 1661.
[20] Open MPI, https://www.open-mpi.org/
[21] Docker document, https://docs.docker.com/config/containers/ resource_constraints/
[22] Micron, https://investors.micron.com/static-files/8ce1925c-f488-47c1-be33-15ba6049b747
[23] KIOXIA, https://personal.kioxia.com/ja-jp/ssd/exceria-plus-nvme-ssd.html
[24] SAMSUNG, https://www.samsung.com/semiconductor/minisite/jp/ssd/consumer/980pro/
[25] Intel， https://www.intel.co.jp/content/www/jp/ja/products/memory-storage/solid-state-drives/consumer-ssds/optane-ssd-8-series/optane-ssd-800p-series/800p-118gb-m-2-80mm-3d-xpoint.html
[26] Intel, https://www.intel.co.jp/content/www/jp/ja/architecture-and-technology optane-technology/balancing-bandwidth-and-latency-article-brief.html