# 完全準同型暗号を用いた秘匿データマイニングにおける低レイテンシ SSD の検討

廣江 彩乃[1] 圓戸 辰郎[2] 小口 正人[1]

**概要**：本研究では，高性能 SSD を活用することで完全準同型暗号アプリケーション実行時における演算時間の課題に対する取り組みについて論じる. 完全準同型暗号を用いると暗号文同士の演算が可能となるため，個人情報を含むビッグデータの演算をクラウドに委託して実行する際のデータ漏洩リスクを低減できる. しかし実行には膨大なメインメモリを必要とし，処理時間がかかる. そのためストレージの観点からこの課題に取り組むことが一つの有効な手段であると考えられる. 本研究ではストレージ IO が比較的多い秘匿データマイニングアプリケーションについて低レイテンシ SSD を用いた性能評価を行う. 今回行う実験では，完全準同型暗号アプリケーションを実行する際に使用する帯域幅と swap 領域に用いる記憶装置の関係に注目する.

# Investigation of a Privacy Preserving Data Mining Application with Fully Homomorphic Encryption on Low Latency SSDs

AYANO HIROE[1] TATSURO ENDO[2] MASATO OGUCHI[1]

## 1. はじめに

近年個人情報を含むビッグデータを扱う処理をクラウドコンピュータに委託する取り組みが増えている. 企業などの様々な組織は顧客から個人情報や技術情報などの機密情報を保持しており，解析技術の発展に伴ってこれらのデータを利用することでさらに有用なデータを得ることができる. そのためには，膨大なデータから有益な情報や特徴的なパターンを抽出することができるような処理能力が高い計算機での演算が必要となる. そこで各企業が保有する膨大なデータをクラウドやデータセンタなどの大型の計算機とストレージを保持する機関に委託し，その機器に対して利用者が問い合わせをすることで統計処理を行っていくことが考えられる. 一方で，クラウドコンピュータ上でこれらのデータから解析結果を得る際には処理依頼データの送信から演算途中，そして演算結果を送信するまでの過程においてデータ漏洩リスクがある. 例えばゲノムデータなどの生体情報や医薬品の開発における技術情報は利用価値が高く盗聴対象となりやすい. このように秘匿情報を扱う際には強固なセキュリティ管理が必要不可欠となる. そこで有用と考えられているのが，データを秘匿したまま処理を行うことができる完全準同型暗号 [1] である. 完全準同型暗号とは暗号文同士の加法と乗算が成立する暗号手法であり，2 章で説明する. クライアント・サーバ構成の秘匿データを扱うシステムに使用する暗号方式には従来の共通鍵暗号などによる暗号化手法も考えられる. しかし暗号化したデータ同士の複雑な演算を行うためにはクライアントの秘密鍵をサーバ側に渡す必要がある. この際にはサーバ側でその秘密鍵を使って暗号化されて送られてきたデータを復号して平文のデータに戻して処理を行う必要があり，この際の盗聴リスクがある. 本研究では，高性能 SSD を活用することで完全準同型暗号アプリケーション実行時における演算時間の課題に対する取り組みについて論じる. 完全準同型暗号を用いると暗号文同士の演算が可能となるため，個人情報を含むビッグデータの演算をクラウドに委託して実行する際のデータ漏洩リスクを低減できる. しかし実行には膨大な実行時間がかかる. そのためストレージの観点から

[1] お茶の水女子大学
Ochanomizu University
[2] キオクシア株式会社
Kioxia Corporation

この課題に取り組むことが一つの有効な手段であると考えられる. 一方で，比較的高性能な低レイテンシ SSD を用いた際に短縮できる実行時間と，それにかかるコストの費用対効果は，実行アプリケーションや実行環境に大きく依存する. よって本研究ではストレージ IO が比較的多い完全準同型暗号アプリケーションについて低レイテンシ SSD を用いた性能評価を行う.

以上の理由から盗聴のリスクが低いのが完全準同型暗号であるが、この暗号化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. アルゴリズムの高速化は進められているものの，依然として演算量が多いためである. それに加えて完全準同型暗号を用いて暗号化するとデータの大きさは元の大きさより更に膨大になる [2]. その結果高価なメインメモリが大量に必要になったり，演算中にメインメモリが不足して swap 処理が発生し，処理速度がメインメモリに遥かに劣る HDD などのストレージ領域を用いる必要が出てきたりする. クラウドコンピュータを使用することを考慮すると特に，用いるコンピュータリソースによって費用がかなり左右される. そのため上で述べたように完全準同型暗号などを用いて秘匿する必要があるビッグデータを使うアプリケーションをメインメモリ上のみで実行することは費用面で現実的ではなく，実行に際して分散するストレージ領域を用いることが想定される. そこで本研究では，近年開発が進む高性能な SSD を活用することを考える. 異なる特徴を持つ複数の SSD を秘匿検索処理の際に用いて，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. そして実行効率とコストの課題を解決することを目的として，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.

## 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 [3].

### 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 [4] によって提唱され，2009 年に Gentry[5] 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 [2].

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 [6].

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 [7].

### 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[8] 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[9] や SEAL[10] 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

### 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

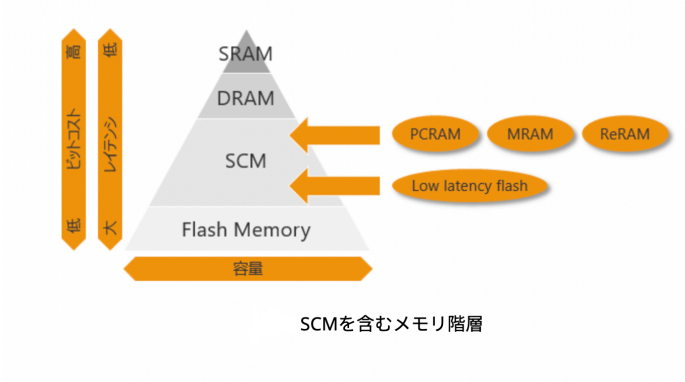


図 1: メモリ階層

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 [11].

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 [12] 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 3DNAND フラッシュメモリ・チップを発表している．3DNAND 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 [13]．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は，実験環境の章にまとめる．

### 3.3 I/O キュー

本研究では，コンピュータリソースへの負荷を計測する上で I/O キューの長さに注目した．キューとは最も基本的なデータ構造の一つであり，先に入れた要素から順に取り出すという規則で出し入れを行うので待ち行列とも言われる．I/O とは入出力の Input/Output の略称であり，I/O キューは，入出力処理の要求を格納して順に取り出すデータ構造である．完全準同型暗号を用いるアプリケーションの実行時には演算処理がボトルネックとなり，IO を行うストレージの性能にキューの状況が左右される．そこで，本研究ではに iostat コマンド [14] を用いて，avgqu-sz の項目で出力される I/O キューの長さを計測した．

## 4. 先行研究

本研究では先行研究 [15] で実装された，完全準同型暗号を用いた秘匿データマイニングアプリケーションを用いる．このアプリケーションには Apriori アルゴリズムが採用されている．

### 4.1 秘匿データマイニング

ビッグデータの主要な活用方法のひとつとして，その膨大なデータの中から特定のパターンやデータを抽出することが挙げられる．その際に用いられるのは個人の身体情報や購買データといった，プライバシが担保される必要がある情報であることが想定される．よって，データを秘匿化したままデータをマイニングする手法が研究されている．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．この手法は，0 と 1 で表現された購買データを対象として 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た．こ

のシステムでは，クライアント側がバイナリデータをひとつずつ暗号化し，それをサーバ側に送信する．サーバ側は受け取ったデータを平文に戻すことなく完全準同型暗号を用いた Apriori 計算を行い，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るサポート値を算出する．このようにサーバ側では受け取ったデータが暗号化された状態のまま演算を行うことができるため，クライアントが保持するデータの中身をサーバ側が知ることはない．よってこの手法は，クラウドなどの外部に演算を委託する際にもデータの秘匿化を保持するという点で有効である．なお完全準同型暗号は加算と乗算の機能を持つが，比較は極めて困難な暗号である [16]．そのため，クライアント側が持つミニマムサポート値との大小比較についてはサーバ側から送られてきたサポート値をクライアント側で復号して比較を行う．

### 4.2 Apriori アルゴリズム

Apriori アルゴリズムはデータの中から頻出パターンを効率的に抽出するためのアルゴリズムであり，Agrawal ら [17] によって 1933 年に提案された．購買データを対象として特定のパターンをマイニングするケースでは，トランザクション毎に各アイテムを購入したかどうかを 0 と 1 でバイナリ表現したデータベースを用いる．そしてその中から各アイテムの部分集合であるアイテムセットを購入したトランザクションの出現頻度を計算する．出現頻度とは，全トランザクション数に対する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数から求めるもので，これをサポート値と呼ぶ．Apriori アルゴリズムでは，徐々にアイテムセットの長さを伸ばしながら幅優先的に探索を繰り返してサポート値を求め，頻出なアイテムセットを導出する．探索は幅優先的に行われる．
このアルゴリズムの流れを図 2 に示す [15]．図 2 の Sup(N) とはアイテム N のサポート値を示し，minSup とは与えられた閾値である，最小サポート値のことである．閾値である minSup よりもサポート値が小さいアイテムが探索対象から抜けていく．頻出ではない集合を含む集合は頻出ではないことから，これらを探索対象から除外することによって計算量を削減している．

## 5. 実験

### 5.1 使用アプリケーション

本研究の実験で用いる秘匿データマイニングアプリケーションのクライアント・サーバ型システムの概要を説明する．システムの概観は図 3 に示す．

アプリケーションの流れは以下に示す通りである．

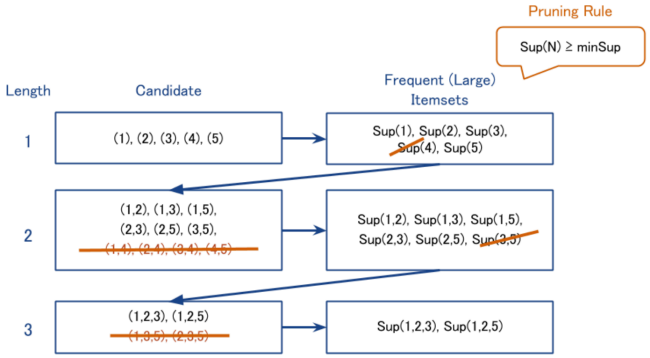


図 2: Apriori アルゴリズム

(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) クライアントはデータを秘密鍵で暗号化しサーバ側に公開鍵と共に委託

(3) クライアントはサーバ側に長さ 1 のアイテムセットに対するサポート値の計算を依頼

(4) サーバ側では完全準同型暗号を用いたサポート値計算を行い，結果をクライアントへ転送

(5) 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を比較

(6) クライアントは閾値を超えたアイテムセットをサーバ側に連絡

(7) 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

(8) 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の長さを 1 ずつ増やしていき，それが最長にな

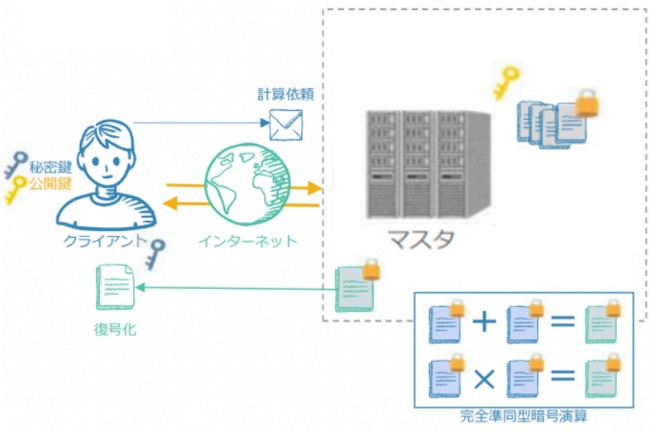


図 3: 秘匿データマイニングアプリケーション概観

るまで，(3)〜(7) の処理の繰り返し

なお本アプリケーションは C++で実装され，完全準同型暗号ライブラリとして HElib を用いている．

### 5.2 検証方法

本研究では上で述べた完全準同型暗号を用いた秘匿データマイニングアプリケーションを用いて，I/O キューの長さなどコンピュータリソースへの負荷を計測する．そして，実行環境の違いによる分析を行う．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るため，swap 処理に着目する．本実験ではクラウド環境を想定し，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．そして，iostat コマンドを 1 秒ごとに実行して，avgqu-sz という項目で取得できる I/O キューの長さを計測し，swap 先メモリに高性能 SSD を指定した各条件について比較・検証していく．なお使用する頻出パターンマイニング用のデータセットは IBM AlmadenQuest research group が開発したジェネレータで生成した．今回実験に用いた購買データはアイテム数 30，トランザクション数 2000 のデータセットでありデータの大きさは 118KB である．

### 5.3 パラメータ選定

本研究ではクラウドコンピュータのインスタンスを利用する際に，限られたメモリ容量の中でビッグデータマイニングを行うとメモリが不足して swap 処理によるストレージ領域の利用が発生することを想定している．一般的なインスタンスではせいぜい 128GB 程度のメモリが搭載され，数テラバイトのデータが用いられる．本実験で用いるデータサイズは 118KB であるが，FHE で暗号化するとそのサイズは一万倍程度に膨れ上がるため 1 GB 程度になる．実験環境の章で示す通り本実験では使用可能メモリを 0.2GB〜0.6GB に制限している．これらの 118KB のデータサイズ 0.2GB〜0.6GB の使用可能メモリは実験の都合上，実際の実行条件を縮小して行っているものであり，実用上に起きうる現象を再現している．

### 5.4 実験環境

表 1 に示すスペックを持つサーバ上でデータマイニング処理を行う．表 2 の 3 種類の SSD を swap 先として指定し，表 3 に示す 3 つの条件下でアプリケーションを実行し，計測していく．

表 1: サーバ

| | |
|---|---|
| CPU | Intel®Xeon®Processor 6 Cores × 2 Sockets |
| DRAM | DDR4 512GB 2133MT/s |
| HDD | HGST SATA 2TB |
| PCIe Gen | 3.0 |

表 2: 比較対象 SSD

| | Kioxia EXCERIA PLUS SSD[22] | Samsung 980 PRO[23] | Intel OPTANE SSD 800P[24] |
|---|---|---|---|
| capacity | 1TB | 500GB | 118GB |
| memory type | NAND Flash | NAND Flash | 3D XPoint |
| sequential R/W (MB/s) | 3,400/3,200 | 6,900/5,000 | No Official Info. |

表 3: 実行条件

| | メインメモリ | Kioxia SSD[22] | Samsung SSD[23] | Intel SSD[24] |
|---|---|---|---|---|
| 条件 (1) | ○ (※) | ○ | - | - |
| 条件 (2) | ○ (※) | - | ○ | - |
| 条件 (3) | ○ (※) | - | - | ○ |

(※) cgroup によって使用可能メモリを制限

### 5.5 実験結果

条件 (1) から条件 (3) において，swap 領域にそれぞれ異なる高性能 SSD を用いて取得した I/O キューの長さを比較する．I/O キューの長さは，iostat コマンドにおける avgqu-sz で取得できる I/O キューの長さの平均値である．

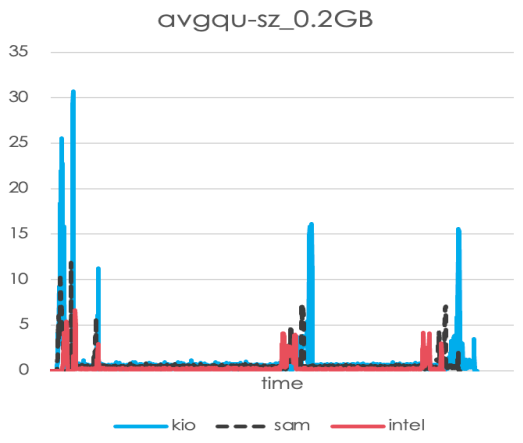


図 4: 使用可能メモリ 0.2GB

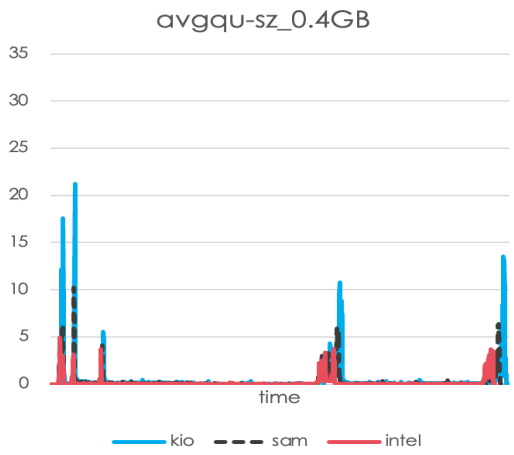


図 5: 使用可能メモリ 0.4GB

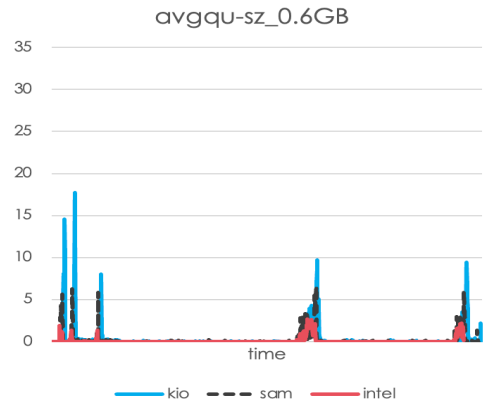


図 6: 使用可能メモリ 0.6GB

図 4〜図 6 にそれぞれ，アプリケーション実行に使用可能なメインメモリを 0.2GB，0.4GB，0.6GB に制限した条件 (1)〜(3) のときの I/O キューの長さをグラフにしたものを示している．これらを見ると，条件 (1),(2),(3) の順に全体的に値が小さくなっていることがわかる．今回用いている SSD の性質を比較してみると，条件 (1) と条件 (2) で用いている SSD は同じ NAND フラッシュメモリを搭載している一方で，条件 (3) の SSD は 3D XPoint というメモリを搭載している．この 3D XPoint メモリは読み書きの時間が短い，低レイテンシという特長がある．よって，条件 (3) が条件 (1) と (2) に比べて比較的処理が高速に行われ，I/O キューの長さが短くなっている．このことから，DRAM が不足する量が多くなるほど SSD の性能が重要になることが分かる．

次に，条件 (1) と条件 (2) について比較する．前述の通り，この 2 つの条件で用いられている SSD は同じ種類のメモリを搭載しているが，図 4〜図 6 を見ると，I/O キューに命令が溜まるタイミングにおいて，I/O キュー長は 2 倍程度異なっている．表 2 に示す通り，条件 (1) と条件 (2) で用いた Kioxia の SSD と Samsung の SSD では，一度に読み書き可能なデータ量を示す Sequential Read/Write の性能に二倍程度の差がある．しかし今回の実験ではこの性能差が出るほどの Read/Write は発生していないため，SSD 内部のコントローラの性能による違いが出たものと考えられる．

## 6. まとめと今後の課題

今回の実験では，完全準同型暗号を用いたデータマイニングアプリケーション実行の際にメインメモリが不足し swap 処理が発生することを想定し，その swap 領域に各種高性能 SSD を使用した際の I/O キューの長さを比較した．その結果，低レイテンシな SSD を用いたときが最もキューの長さが短く，レイテンシ性能が完全準同型暗号のような重い演算処理に有効であることが分かった．今後は，完全準同型暗号を用いた重い処理のアプリケーションの実行時間の長さの課題に対して低レイテンシな性能をもつ SSD を活用していくために，swap 領域以外への適用も模索していきたいと考えている．

## 謝辞

本研究の一部は、キオクシア株式会社の支援をうけて実施したものです。

### 参考文献

[1] Craig Gentry, et al., ”Fully homomorphic encryption using ideal lattices.” In STOC, Vol. 9, pp. 169–178, 2009
[2] 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
[3] 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
[4] R. L. Rivest et al., “On data banks and privacy homomorphisms,” Foundations of secure computation, vol. 4, no. 11, 1978, pp. 169 – 180
[5] Craig Gentry, et al., ”Fully homomorphic encryption using ideal lattices.” In STOC, Vol. 9, pp. 169–178, 2009
[6] 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
[7] 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
[8] Shoup V. and Halevi S., http://shaih.github.io/HElib/index.html
[9] PALISADE, https://palisade-crypto.org/software-library/
[10] Microsoft SEAL, https://github.com/Microsoft/SEAL
[11] KIOXIA，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
[12] IBM Journal of Research and Development , ”Storage-

class memory: The next storage system technology”, Volume: 52 , Issue: 4.5 , July 2008
[13] 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
[14] Linux Manual of iostat https://ysatoautonomous.github.io/jm-draft/man2html/sysstat/sysstat-12.2.0/draft/man1/iostat.1.html
[15] http://www.is.ocha.ac.jp/ oguchi_lab/Publications/paper2018 /dicomo2018_yamamotoy.pdf
[16] https://www.tcs.com/content/dam/tcs/pdf/research-innovation/insights/fully-homomorphic-encryption-data-privacy.pdf
[17] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining asso- ´ ciation rules between sets of items in large databases. In Acm sigmod record, Vol. 22, pp. 207–216. ACM, 1993.
[18] 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.
[19] 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.
[20] David Wai-Lok Cheung, Vincent TY Ng, and Benjamin W Tam. Maintenance of discovered knowledge: A case in multilevel association rules. In KDD, Vol. 96, pp. 307–310, 1996.
[21] Micron, https://investors.micron.com/static-files/8ce1925c-f488-47c1-be33-15ba6049b747
[22] KIOXIA, https://personal.kioxia.com/ja-jp/ssd/exceria-plus-nvme-ssd.html
[23] SAMSUNG, https://www.samsung.com/semiconductor/minisite/jp/ssd/consumer/980pro/
[24] 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
[25] Intel, https://www.intel.co.jp/content/www/jp/ja/architecture-and-technology optane-technology/balancing-bandwidth-and-latency-article-brief.html