準同型暗号への In-Storage Computing 適用の検討
－準同型暗号上での畳み込みニューラルネットワークの推論処理高速化を目指して－

鈴木 拓也† 山名 早人‡

†早稲田大学大学院基幹理工学研究科 〒169-8555 東京都新宿区大久保 3-4-1
‡早稲田大学理工学術院 〒169-8555 東京都新宿区大久保 3-4-1
E-mail: †t-suzuki@yama.info.waseda.ac.jp, ‡yamana@yama.info.waseda.ac.jp

あらまし データに含まれる秘密を守ったままデータを処理する方法として，準同型暗号が注目されている．準同型暗号には，時間計算量と空間計算量が大きいという問題点がある．時間計算量への対応としては，GPU や FPGA へのオフロードによる高速化手法が提案されている．しかし，空間計算量が大きいため，オフロード時のデータ転送のオーバヘッドがボトルネックとなる．本論文では，このデータ転送オーバヘッドを in-storage computing により削減することに取り組む．特に，メインメモリにも乗り切らないほど総データサイズが大きい場合の，FPGA 等とストレージ間のデータ転送のオーバヘッドを削減する方法として，FPGA 等の演算器をストレージに載せた computational storage device を用いて，ストレージ内で演算を行う in-storage computing による効果を検証した．アプリケーションとして，準同型暗号上での畳み込みニューラルネットワークを取りあげ，in-storage computing の適用効果をシミュレーションによって評価した．

キーワード In-storage computing，Computational storage device，準同型暗号，セキュリティ，高性能計算

1. はじめに

ビッグデータは，検索や推薦システムの他にも，医療といった様々な分野において利活用されている．ビッグデータを利活用するためには，計算機資源が必要となる．そこで，近年盛んに利活用されているクラウドコンピューティングを用いることで，計算機資源を保有せずとも，ビッグデータを利活用することができる．

ビッグデータには，個人情報といった機密情報が含まれる場合がある．例えば，医療関係では，薬歴といった情報が該当する[1]．クラウド上でビッグデータを利活用する際には，このような機密情報をクラウド上へアップロードする必要がある．そのため，データに含まれる機密情報が，悪意のあるクラウド管理者や外部の第三者によって流出する危険性がある．

この機密情報が流出する危険を回避する手段の一つとして，準同型暗号を活用することが挙げられる．準同型暗号を用いることで，暗号化されたデータを復号せずに処理することができる．したがって，暗号化されたデータが流出もしくは悪用されたとしても，データに含まれる機密情報がサービス提供者を含めた他者に知られることはない．

しかし，準同型暗号には，実用する上での下記の課題点が存在する．なお，準同型暗号上での演算を準同型暗号演算と呼ぶ．

課題1. 準同型暗号演算の実行時間が長いこと
課題2. データサイズが大きいこと

課題 1 については，FPGA を用いて準同型暗号演算を高速化する手法が提案されている[2]．しかし，課題 2 のデータサイズが大きいことにより，準同型暗号アプリケーションに使用するデータが FPGA ボード上の DRAM に乗り切らない場合がある．この場合は，ホストから FPGA へデータを転送する必要がある．さらに，データがメインメモリにも乗り切らない場合は，SSD や HDD といったストレージからデータを転送する必要がある．データ転送はオーバヘッドとなり，ストレージを使用する必要があると，さらに準同型暗号演算の実行時間が長くなる．例えば，CPU 上で準同型暗号アプリケーションを実行する際に，スワップ処理によって SSD へのデータ退避が発生すると，実行時間が 16%以上増加することが明らかとなっている[3]．すなわち，大きなメモリ空間を必要とする準同型暗号アプリケーションを高速化するためには，いかにデータ転送による性能劣化を軽減しつつ，FPGA に準同型暗号演算をオフロードするかが重要となる．

ストレージと FPGA ボード間のデータ転送のオーバヘッドを軽減する方法として，in-storage computing（以下，ISC）が挙げられる．ISC は，SSD や HDD といったストレージデバイスに FPGA や ARM プロセッサ，ASIC（application specific integrated circuit）を載せ，ストレージデバイス内で演算を行う手法である．特に，ISC を行う目的で FPGA 等を載せたストレージを computational storage device（以下，CSD）と呼ぶ．ISC はニューラルネットワークといった様々な手法に適用

され，実行時間や消費電力の削減に効果があることが明らかとなっている[4,5,6,7,8,9]．しかし，準同型暗号への ISC 適用はされていない．準同型暗号と ISC を組み合わせることで，課題 2 による準同型暗号演算の実行時間が長いという問題を解消する一つの手段となり得る．

そこで，本論文では，ISC を準同型暗号へ適用する前段階として，準同型暗号アプリケーションに ISC を適用した際のシミュレーションを実施し，その結果を示す．得られた結果から，準同型暗号に適した CSD を構築するためには，どのような性能や構成が候補となり得るかについて考察する．なお，シミュレーション対象の準同型暗号アプリケーションとして，畳み込みニューラルネットワークの推論処理を用いる．畳み込みニューラルネットワークは，各層の重みパラメータや中間層のノード数が多いために，総データサイズが大きく，大きなメモリ空間を必要とする．そのため，ISC の恩恵を受けると考えられる畳み込みニューラルネットワークの推論処理をシミュレーション対象とした．

本論文は，以下の構成をとる．2 節で関連研究を示す．3 節にて，本論文で実施するシミュレーションが対象とするシステムについて説明する．4 節でシミュレーションによる評価実験について示す．最後に，5 節でまとめる．

2. 関連研究

2.1. FPGA による準同型暗号演算高速化に関する研究

準同型暗号演算を高速化する方法として，FPGA を活用する研究が存在する．FPGA を活用する研究としては，(1)準同型暗号演算を構成する一部の処理のみを FPGA 上で実行する方法と，(2)準同型暗号演算を構成する全ての処理を FPGA 内に閉じて実行する方法の 2 つに分けられる．

準同型暗号演算を構成する処理として，数論変換が存在する．数論変換は NTT（Number Theoretic Transform）とも呼ばれる．数論変換を FPGA で高速に実行する手法が提案されている[10]．そして，数論変換やその逆変換（以下，INTT）を高速化することで，準同型暗号特有の演算である key switching と呼ばれる演算を高速化することができる．しかし，FPGA 内に閉じて準同型暗号演算を実行できないと，CPU と FPGA 間でのデータ転送が発生することとなり，準同型暗号演算の実行時間短縮には不十分である．

2020 年に Riazi らが， FPGA 上で準同型暗号演算実行を完結できる実装（HEAX）を発表した[2]．例えば，key switching と呼ばれる準同型暗号特有の演算を対象に，FPGA ボード上の DRAM にデータが乗っている条件下で，CPU でのシングルスレッド実行と比較して 91.7～232.5 倍のスループットを達成した．

2.2. ISC に関する研究

文字列検索や高速フーリエ変換（以下，FFT），深層学習といった，様々なアプリケーションの実行を ISC により高速化できることが示されている．特に，ISC を盛んに研究している研究機関として，カリフォルニア大学のアーバイン校と NGD Systems, Inc.[1]が挙げられる．この 2 つの機関は共同研究を実施している．

まず，2018 年に Torabzadehkashi らは，CompStor という ISC アーキテクチャを発表した[4]. CompStor は，ファイルの圧縮や展開，文字列探索を対象に，データ転送による消費電力削減を行うことで，CPU 実行と比較して 3 倍の消費電力効率を達成した．

2019 年には，Torabzadehkashi らが発表した Catalina という ISC アーキテクチャでは，画像の類似度判定を対象に，ARM プロセッサと FPGA の両方を採用することで，ARM プロセッサのみを使用する場合と比べて，2 倍以上の高速化と消費電力効率を達成した[5]. また，2019 年に Torabzadehkashi らは，Catalina が様々なベンチマークや計算アルゴリズムで使われる FFT を対象に高速化や消費電力削減を達成できることを示した[6]. さらに，2019 年に，Torabzadehkashi らは，ソートアルゴリズムや離散フーリエ変換についても高速化や低消費電力化を達成したことを示した[7].

2020 年に HeydariGorji らは，Stannis というフレームワークを発表した[8]. Stannis は，深層学習の学習処理を対象に，消費電力に応じて負荷分散を行うことで，2.7 倍の高速化と 69%の消費電力削減を達成した．さらに，2020 年に HeydariGorji らは，Laguna という CSD を発表した[9]．連合学習を対象に，Stannis と Laguna を組み合わせることで，消費電力効率を 2.45 倍改善しつつ，処理速度を最大 3.1 倍に向上させた．

これらの 6 つの論文[4,5,6,7,8,9]において，CSD を複数台使用するほど，高速化できることが示されている．これは，使用する CSD の台数を増やした際に，CPU といったホストと各 CSD のデータ転送の帯域幅が限られているのに対して，1 台の CSD 内でのデータ転送帯域幅は使用台数に依存しないためである．すなわち，アプリケーション実行において，それぞれの CSD 上でできるだけ完結するように処理を分割して並列実行することで，CSD の恩恵をより受けることができる．

アプリケーションと ISC の親和性に関する研究も行われている．2019 年に Song らは，CSD の構成として，

[1] https://www.ngdsystems.com/

FPGA を採用する場合と ARM プロセッサを採用する場合の比較結果を発表した[11]．線形分類器やヒストグラム均等化では ARM プロセッサを採用する方が消費電力が低い一方で，k 近傍法や FFT での消費電力や処理速度では FPGA を採用する方が優れているという結果が得られた．2019 年に Ruan らは，ISC とアプリケーションの親和性を発表した[12]．Ruan らは，下記の 2 つの条件を満たすアプリケーションが ISC に適していると結論づけた．

- 演算強度（operational intensity）が低い．なお，この論文[12]における演算強度の定義は，パイプラインを考慮した上での，1 つの入力を処理するのにかかるサイクル数である．
- ホストと CSD 間の転送データサイズに対する，CSD 内での読み書きデータサイズの比が高い．

具体的なアプリケーションとしては，k 近傍法や SQL，grep コマンドが該当する．

2.3. 関連研究のまとめ

準同型暗号は，FPGA 上で実行が可能であり，高速化できることが示されている．しかし，全てのデータが FPGA の DRAM 上に乗っていることが前提である．そのため，畳み込みニューラルネットワークのような，総データサイズが大きく，ホストと FPGA 間のデータ転送がボトルネックとなる場合では，既存の FPGA 実装では不十分である．

上記を踏まえて，準同型暗号と ISC の親和性を考える．準同型暗号で多用される数論変換は FFT と同様のアルゴリズムであり，既存の ISC の研究[11]では，FFT の高速化が達成されている．したがって，数論変換と ISC の親和性が高いと推測される．また，準同型暗号上での加算（準同型加算）や乗算（準同型乗算）は，数論変換よりも演算強度が低い．したがって，Ruan らの研究結果[12]より，準同型加算や準同型乗算も ISC との親和性が高いことが期待される．

したがって，準同型暗号と ISC を組み合わせることで，準同型暗号アプリケーション実行にかかる総データサイズが大きい状況においても，HEAX[2]と異なり，FPGA による高速化の恩恵を受けられることが期待できる．

3. システム構成

本シミュレーションで対象とするシステムについて説明する．図 1 にシステムアーキテクチャの概略図を示す．このようなアーキテクチャは ISC の研究でよく用いられる構成である[7,11,12]．CPU といったホストと CSD 間は PCIe で接続されるとする．CSD には NAND フラッシュと FPGA ボード（DRAM 含む）の他に，インターフェース類や NAND フラッシュのコントローラが搭載される．準同型暗号演算は FPGA 上で実行される．また，アプリケーションで用いられるデータの内，畳み込みニューラルネットワークの重みパラメータのデータや準同型暗号の公開鍵が予め NAND フラッシュに保存される．なお，公開鍵は，準同型暗号特有の relinearization[2]や rotation[3]といった演算を実行する際に必要となる．

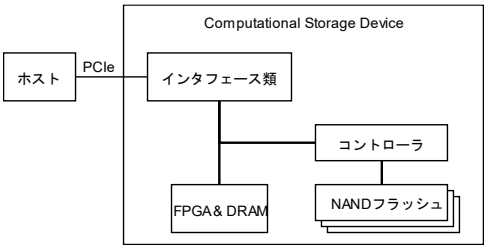

図 1　対象システムアーキテクチャの概略図

CSD のパラメータとして用いる要素を以下に示す．

- FPGA のゲート数や動作周波数
- FPGA ボード上の DRAM 容量
- PCIe の帯域幅
- NAND フラッシュからの読み込みの帯域幅
- NAND フラッシュへの書き込みの帯域幅
- NAND フラッシュの容量

シミュレーションを簡略化するために，FPGA ボードの性能は，演算ごとの実行時間をパラメータとして設定した．また，FPGA ボードの DRAM 容量は HEAX[2]を元に 8[GB]と設定した．また，PCIe や NAND フラッシュへの読み書きにおける帯域幅は理論値を採用した．準同型暗号では，平文や暗号文，公開鍵のデータサイズは数十[KB]から数十[MB]である．そして，準同型暗号では暗号文を条件とした条件分岐ができないという特徴があるため，NAND フラッシュやホストからのデータ転送の順番を予め推測することができる．すなわち，予め連続して読み出せるように NAND フラッシュ上にデータを配置することができるため，NAND フラッシュのシーケンシャルリードでの性能を活かすことができる．また，データ転送にかかるレイテンシを隠蔽することができる．NAND フラッシュの容量は十分大きいとした．

4. 評価実験

4.1. 実験目的と比較対象

評価実験では，準同型暗号に ISC を適用した際に，どの程度高速化できるかを明らかにすることを目的と

[2] 2 つの暗号文を入力とする準同型乗算の出力に使用する必要がある準同型暗号特有の演算である．

[3] 1 つの暗号文に暗号化したベクトルの要素を回転させるために必要な準同型暗号特有の演算である．

する．この目的を達成するために，3 節で示したシステムを用いた ISC 上での準同型暗号アプリケーション実行についてのシミュレーションを実施した．

CSD の性能を比較するために，CSD ではなく FPGA ボードと SSD を併用する構成や FPGA ボードのみを利用する構成についても実験を実施した．なお，FPGA のみを利用する構成では，ホストのメインメモリは十分大きな容量を持つとした．また，準同型暗号では暗号文を条件とした条件分岐ができないという特徴により，SSD からホストのメインメモリへのデータ転送の順番やホストのメインメモリから FPGA ボードへのデータ転送の順番は予め推測可能である．そのため，SSD からメインメモリへのデータ転送にかかるレイテンシやメインメモリから FPGA ボードへのデータ転送にかかるレイテンシは隠蔽できるとした．すなわち，FPGA を用いる 2 つの構成と CSD を用いる構成との違いは，FPGA を用いる構成では，PCIe の帯域幅がデータ転送に影響を与えるのに対して，CSD を用いる構成では NAND フラッシュからの読み出し帯域幅がデータ転送に影響を与える点である．

ISC 適用による高速化の効果を定量的に測定するために，CPU 実行との比較も実施した．なお，CPU 実行については，準同型暗号演算ごとの実行時間の実測値を用いて，推論処理のレイテンシを算出した．

4.2. 評価実験に用いたアプリケーション

データセットとして，MNIST データセット[4]を用いた．MNIST データセットは，手書きの数字が書かれた 28×28 のモノクロ画像とそのラベルデータの集合で構成される．

畳み込みニューラルネットワークのモデルとして，Ishiyama らが提案した手法[13]を元にする．活性化関数は，2 次の近似関数を使用した．また，本研究で使用する準同型暗号方式の一つである CKKS 方式[14]は，実部と虚部が固定小数点数表現の複素数を扱うことができるため，準同型暗号上での機械学習やニューラルネットワークにて活用されている．CKKS 方式では，パッキング[15,16]という手法が利用可能である．パッキングとは，一定の長さのベクトルを 1 つの平文や暗号文に埋め込む手法である．Ishiyama らは，特定の位置のピクセルごとに，複数画像をまとめてパッキングしていた[13]が，本シミュレーションでは，チャネルごとにパッキングする[17]．チャネルごとにパッキングすることで，スループットは低下する代わりにレイテンシを短縮することができる．チャネルごとにパッキングする場合は，畳み込み層や全結合層で rotation を使用する必要がある．本実験での準同型暗号上の畳み込みニューラルネットワークの推論処理においては，Ishiyama らの最適化手法[13]の他に，Lou ら[18]によって提案された最適化手法も適用した．ただし，本研究では，分類性能については対象外である．そのため，重みパラメータの値に依存した最適化[5]は適用しなかった．また，本論文では，学習時のエポック数や正解率については言及しない．

準同型暗号上で畳み込みニューラルネットワークの推論処理を実行するためには，準同型暗号パラメータと，relinearization や rescaling[6]といった準同型暗号特有の演算を実行するタイミングを設定する必要がある．本研究では，Dathathri らが提案した EVA[19]を用いて設定した．GitHub にて公開されている EVA のプログラム[7]を使用し，EVA への入力パラメータとして，input_scales と output_ranges を 30 とし，vec_size を 8,192 とした．

4.3. シミュレーション条件の設定

3 節で示したシステムにおけるパラメータを表 1 で設定した．CSD の FPGA や DRAM については，後述の HEAX の論文[2]で使用されていた Intel Stratix 10 の性能を元に設定した．そのため，DRAM 容量は 8[GB]と設定した．PCIe の帯域幅については，多数のデバイス（FPGA や SSD，CSD）を 1 つのホストに接続する構成を考慮し，1, 2, 4, 8, 16[GB/s]のいずれかとした．また，NAND フラッシュの容量は十分大きい値とした．

表 1　シミュレーションにおけるシステムのパラメータ設定

要素	値
CSD 上の DRAM 容量[GB]	8
CSD 上の NAND フラッシュと DRAM 間の帯域幅[GB/s]	8
ホストとデバイス間の帯域幅[GB/s]	1, 2, 4, 8, 16 のいずれか

FPGA 上での各準同型暗号演算の実行時間について説明する．本論文では，実行時間のデータとして，HEAX の論文[2]でパラメータセットの Set-C について示されている値を使用した．なお，HEAX の論文[2]では特定の準同型暗号パラメータでのスループットのみが示されている．例えば，準同型暗号パラメータの一つであるレベル（その暗号文に適用できる準同型乗算の残り回数）に応じて，準同型暗号演算の時間計算量が変化する．そのため，本論文では，HEAX の論文[2]で示されている実装方法を元に他の準同型暗号パラ

[4] http://yann.lecun.com/exdb/mnist/

[5] 例えば，0 を足し合わせる処理があった時に，この処理を実行しないことで，計算結果への影響を与えずに，実行時間を短縮できる．

[6] 準同型乗算の出力に対して使用する必要がある準同型暗号特有の演算である．

[7] https://github.com/microsoft/EVA

メータでの実行時間を推定した．

HEAX の論文[19]では，実験結果として，スループットのみが示されている．FPGA では，処理をパイプライン化することでスループットを高める．そのため，一般的にレイテンシはスループットの逆数よりも長い値となる．しかし，準同型乗算では，準同型乗算を行う回路の規模に対して，準同型乗算の入力データのデータサイズが大きいため，スループットの逆数をレイテンシとした．準同型加算については，論文中で示されていないため，レベル 0 の暗号文同士の準同型加算の実行時間を基準の 10[us]と仮定し，その他のパラメータにおける実行時間は，準同型加算の時間計算量を元に算出した．relinearization や rotation の実行時間は，HEAX の回路構成の方法により，レベルに依存せずに一定の実行時間とした．これは，relinearization や rotation に含まれる key switch と呼ばれる演算における，2 段階目の INTT の処理がボトルネックとなる回路構成であり，この 2 段階目の INTT の実行回数は入力暗号文のレベルに依存しないためである．rescaling の実行時間は，HEAX に記載されている NTT と INTT のスループットを元に理論的に算出した．なお，各準同型暗号演算の時間計算量については，Microsoft SEAL[20]の実装を参考にした．

シミュレーションを行う上でのデータの初期位置といった実験条件を下記に示す．

- 1 つの FPGA ボードもしくは CSD 上で同時に実行される推論処理は 1 つのみとし，推論処理実行中に他の推論処理が開始されることはないとした．
- 入力データはホストから転送されるが，測定開始時点で既に FPGA ボードもしくは CSD 上の DRAM 上に格納されているとした．
- 実行開始時点では，畳み込みニューラルネットワークの重みパラメータ及び公開鍵は SSD もしくは CSD の NAND フラッシュに格納されているとした．ただし，FPGA のみを利用する構成では，ホストのメインメモリ上に格納されているとした．
- 推論結果の出力データが FPGA ボードもしくは CSD 上で得られた時点で推論処理完了とした．

CPU 上での実行時間として，Intel Xeon Gold 5122 上で EVA のランタイムを用いて実行した際の実測値を用いた．ただし，EVA のランタイムで実行される演算の内，畳み込みニューラルネットワークの重みパラメータのパッキングといった事前実行可能な演算の実行時間を除外した．Intel Xeon Gold 5122 は物理コアが 4 コア存在するが，1 つの推論処理をシングルスレッドで実行した際の実行時間を測定した．また，CPU 実行で用いた計算機のメインメモリの容量は，1 回の推論処理を実行するのに十分な容量であったため，スワップメモリやストレージへのデータの退避は発生しなかった．

4.4. シミュレーション結果

シミュレーションによって得た実行時間を表 2 に示す．帯域幅が広いほど，実行時間が短くなることが分かる．特に，帯域幅が 1[GB/s]から 2[GB/s]になると，1.98 倍高速化した．しかし，帯域幅が広くなるにつれて実行速度の向上率が低下した．例えば，帯域幅が 8[GB/s]から 16[GB/s]へ広くなっても，実行速度は 1.02 倍に留まった．

表 2　シミュレーションで取得した帯域幅ごと及び CPU 実行での推論処理時間

CSD 上の NAND フラッシュから DRAM への帯域幅 もしくはホストや SSD から FPGA ボード上の DRAM への PCIe の帯域幅[GB/s]	実行時間[ms]
1	3,858
2	1,951
4	1,321
8	1,071
16	1,055
CPU 実行（Intel Xeon Gold5122）	11,359

本実験で必要な NAND フラッシュの容量は，畳み込みニューラルネットワークの重みパラメータで 3.3[GB]，公開鍵で 0.5[GB]であった．また，実行に必要な総メモリサイズは 17.3[GB]であった．不要になったデータや公開鍵を随時 FPGA ボード上の DRAM から削除したため，CSD 上の DRAM 容量は 8[GB]で足りたため，FPGA ボード上の DRAM から NAND フラッシュ等への書き出しは不要であった．

4.5. 考察

4.5.1. CSD の構成に関する考察

帯域幅が 8[GB/s]の実行時間と 16[GB/s]の実行時間を比較すると，差は 2%である．したがって，帯域幅は 8[GB/s]あれば十分であると考えられる．この 8[GB/s]は 16 チャネルの NAND フラッシュからの読み出し帯域幅の理論値[7]であるため，従来の NAND フラッシュの構成で十分な性能が得ることができると考えられる．

今回の実験では，1 回の実行に必要な NAND フラッシュの容量は，畳み込みニューラルネットワークの重みパラメータと公開鍵を合わせて 3.8[GB]であった．実環境を考えると，1 台の CSD で実行できる畳み込みニューラルネットワークのモデルごとに重みパラメータで 3.3[GB]の容量が必要となる．ただし，より複雑なモデルを使用する場合は，必要な容量が大きくなる．

同様に，クライアント・サーバモデルを対象とすると，1 台の CSD で対応できるクライアントごとに公開鍵で 0.5[GB]の容量が必要となる．クライアントから

推論処理の依頼が来てからクライアントに推論結果を送るまでのレイテンシの観点から，1 台の CSD でできるだけ多くのクライアントの公開鍵を保有しておく必要がある．例えば，クライアント総数が 10,000 人で，2 台の CSD を活用する場合を考える．1 台の CSD に 5,000 人分の公開鍵しか保存しないとすると，その 5,000 人の内の 2 人から同時に推論処理の依頼を受けた際に，もう一方の CSD が利用可能であるにも関わらず，片方のクライアントのレイテンシが 2 倍となってしまう．もしくは，CSD 間でのデータ転送が発生することになり，レイテンシの悪化が想定される．これらの問題を避けるには，$0.5[\mathrm{GB}] \times 10{,}000[\text{人}] = 5{,}000[\mathrm{GB}]$の容量が 1 台の CSD 上の NAND フラッシュに必要となる．

4.5.2. 複数台の CSD 利用に関する考察

Torabzadehkashi らの論文[4,5,6,7]や HeydariGorji らの論文[8,9]により，CSD の台数を増やすことで，さらなるスループット向上が期待できる．台数を増やすと，ホストとの CSD 間の帯域幅は狭くなる．しかし，CSD の場合は，CSD 上の NAND フラッシュからデータを読み出すため，台数を増やしても CSD 上の NAND フラッシュから CSD 上の DRAM への帯域幅を 8[GB/s]に保つことができる．したがって，HEAX の方法を複数台の FPGA で活用する場合と比べて，実行時間の削減やスループットの向上が期待できる．なお，ボトルネックとなり得る FPGA 間や CSD 間のデータ転送を無くすために，クエリレベル並列を用いるとした．

具体例を用いて考察する．前提条件として，PCIe レーンを FPGA や CSD，SSD に合計で 32 レーン使用できる場合を想定する．下記の 3 種類の構成について比較する．図 2 に下記の 3 種類の構成を図示する．

1. FPGA と SSD を併用する構成
2. FPGA のみを利用する構成
3. CSD のみを利用する構成

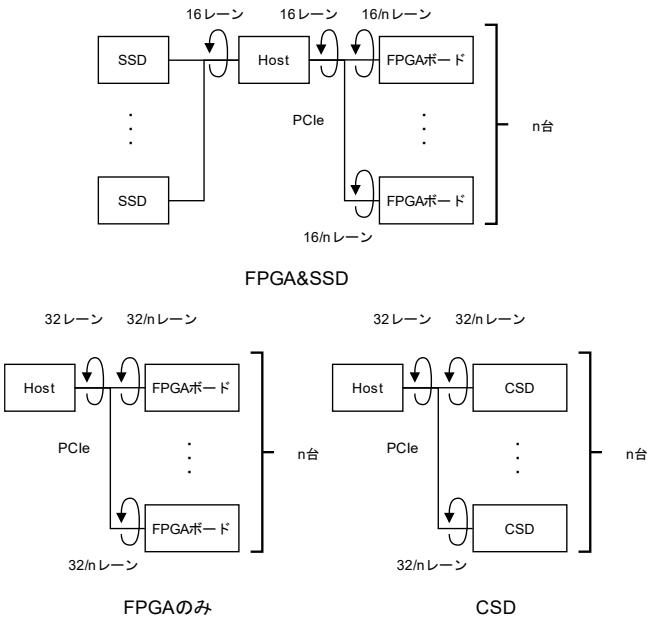

図 2　複数台利用で比較する各構成

なお，FPGA と SSD を併用する構成では，FPGA ボード群と SSD 群で半分の 16 レーンずつ使用するとした．これは，一方のデバイス群の帯域幅がボトルネックとなることを避けるためである．また，FPGA のみを利用する場合は，ホストの DRAM 容量が十分に大きく，直接ホストから FPGA へデータを送ることができるとした．CSD 上の NAND フラッシュと CSD 上の DRAM 間の帯域幅は 8[GB/s]とした．なお，簡単のため，畳み込みニューラルネットワークの入力データや出力データの転送については考慮していない．

表 3　FPGA や CSD の台数ごとの推論処理の実行時間とスループット

FPGA もしくは CSD の台数	実行時間[ms]			スループット[/s]		
	FPGA &SSD	FPGA のみ	CSD	FPGA &SSD	FPGA のみ	CSD
1	1,055	1,055	1,071	0.948	0.948	0.934
2	1,071	1,055	1,071	1.867	1.896	1.867
4	1,321	1,071	1,071	3.028	3.735	3.735
8	1,951	1,321	1,071	4.100	6.056	7.470
16	3,858	1,951	1,071	4.147	8.201	14.939
32	—	3,858	1,071	—	8.294	29.879

FPGA や CSD の台数ごとの実行時間とスループットを表 3 に示す．台数を増やすことで，CSD を利用する構成ではスループットが向上することが分かる．一方で，FPGA と SSD を併用する構成では，FPGA を 4 台以上利用するとスループットの向上率が 30%以上低下していく．同様に，FPGA のみ利用する構成でも，8 台以上を利用すると，スループットの向上率が 30%以上低下していく．これらの結果は，台数が増えることにより，FPGA ボード 1 台あたりに割り当てられる帯域幅が狭くなるために，実行時間が長くなったことが原

因である．また，16 台使用した場合を想定すると，FPGA と SSD を併用する構成と比べて CSD を利用する構成では，実行時間を 72.2%削減しつつ，スループットを 3.60 倍に向上できると推定される．

今回対象外とした畳み込みニューラルネットワークの入力データや出力データの転送を考慮すると，FPGA と SSD を併用する構成や FPGA のみを利用する構成では，さらに性能が劣化すると考えられる．一方で，CSD を利用する構成では，ホストとの主な通信は畳み込みニューラルネットワークの入力データと出力データのみであるため，ホストと CSD 間の帯域幅が狭くなっても性能への影響は比較的小さいと考えられる．

4.5.3. CPU との比較に関する考察

CPU（Intel Xeon Gold 5122）と比較すると，帯域幅が 8[GB/s]であれば，CPU 実行と比べて 10.6 倍高速化することが分かる．しかし，本シミュレーションでは，畳み込みニューラルネットワークの入力データや出力データに関するホストと CSD 間の転送する処理を対象外とした．そのため，実環境では CSD の性能は今回のシミュレーションで得られた性能よりも劣化する．具体的には，実環境では，クライアントから入力暗号文の受信から出力暗号文の送信までを考慮する必要がある．CSD 実行は，CPU 実行と比べて，クライアントとサーバ間の転送は同じであるが，ホストと CSD 間の入出力暗号文の転送の分だけ劣化する．

この性能劣化を軽減する方法として，イーサネットワークモジュールを搭載した FPGA ボードを用いて CSD を実装することが挙げられる．すなわち，ネットワーク経由で受け取った入力暗号文を直接 FPGA ボード上の DRAM に格納し，かつ出力暗号文を FPGA ボードからそのまま送り返すことができるようになる．その結果，入出力暗号文に関するホストと CSD 間の通信を削減し，データ転送のオーバヘッドを削減することができるようになる．

5. まとめ

本論文では，準同型暗号演算高速化を目的として，準同型暗号への ISC 適用について検証した．畳み込みニューラルネットワークを対象としたシミュレーション評価を実施した．

本論文で示したシミュレーション結果より，16 台の CSD を活用することで，従来の FPGA 実装を用いた 16 台の FPGA 利用時と比べて，実行時間を 72.2%削減しつつ，スループットを 3.60 倍向上する効果を期待できることが分かった．また，1 コア利用時の CPU（Intel Xeon Gold 5122）上での実行と比べ，10.6 倍の高速化を期待できることが分かった．

今後の課題は，今回のシミュレーション結果を元に，実機上で評価検証を行うことである．また，より大規模な畳み込みニューラルネットワークモデルに対する適用が挙げられる．これに伴い，CSD 上の DRAM 容量が不足した際の NAND フラッシュへの退避処理を考慮することも今後の課題である．また，メニーコア CPU や GPU との比較や，CSD の構成部品の性能と価格の関係を明らかにすることも今後の課題である．

謝　辞

本研究の一部はキオクシア株式会社と早稲田大学との組織連携活動の一環として実施したものです．

参　考　文　献

[1] Y. Jiang, T. Noguchi, N. Kanno, Y. Yasumura, T. Suzuki, Y. Ishimaki, and H. Yamana, “A Privacy-Preserving Query System using Fully Homomorphic Encryption with Real-World Implementation for Medicine-Side Effect Search”, In Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, pp. 63–72, 2019.

[2] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, “HEAX: An Architecture for Computing on Encrypted Data”, In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 1295–1309, 2020.

[3] 廣江彩乃，圓戸辰郎，小口正人，“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”，第 13 回データ工学と情報マネジメントに関するフォーラム，G25-5，2021.

[4] M. Torabzadehkashi, S. Rezaei, V. Alves, and N. Bagherzadeh, “CompStor: An In-storage Computation Platform for Scalable Distributed Processing”, In 2018 IEEE International Parallel and Distributed Processing Symposium Workshops, pp. 1260–1267, 2018.

[5] M. Torabzadehkashi, S. Rezaei, A. HeydariGorji, H. Bobarshad, V. Alves, and N. Bagherzadeh, “Catalina: In-Storage Processing Acceleration for Scalable Big Data Analytics”, In 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 430–437, 2019.

[6] M. Torabzadehkashi, A. HeydariGorji, S. Rezaei, H. Bobarshad, V. Alves, and N. Bagherzadeh, “Accelerating HPC Applications Using Computational Storage Devices”, In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems, pp. 1878–1885, 2019.

[7] M. Torabzadehkashi, S. Rezaei, A. HeydariGorji, H. Bobarshad, V. Alves, and N. Bagherzadeh, “Computational storage: an efficient and scalable platform for big data and HPC applications”, Journal of Big Data, vol. 6, no. 100, pp. 1–29 2019.

[8] A. HeydariGorji, M. Torabzadehkashi, S. Rezaei, H. Bobarshad, V. Alves, and P. H. Chou, “Stannis: Low-Power Acceleration of DNN Training Using Computational Storage Devices”, In the 57th ACM/IEEE Design Automation Conference, pp. 1–6, 2020.

[9] A. HeydariGorji, S. Rezaei, M. Torabzadehkashi, H.

Bobarshad, V. Alves, and P. H. Chou, “HyperTune: dynamic hyperparameter tuning for efficient distribution of DNN training over heterogeneous systems”, In Proceedings of the 39th International Conference on Computer-Aided Design, no. 88, pp. 1–8, 2020.

[10] R. Paludo and L. Sousa, “Number Theoretic Transform Architecture suitable to Lattice-based Fully-Homomorphic Encryption”, In 2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors, pp. 163–170, 2021.

[11] X. Song, T. Xie, and S. Fischer, “A Near-Data Processing Server Architecture and Its Impact on Data Center Applications”, ISC High Performance, LNCS, vol. 11501, pp. 81–91, 2019.

[12] Z. Ruan, T. He, and J. Cong, “Analyzing and Modeling In-Storage Computing Workloads On EISC — An FPGA-Based System-Level Emulation Platform”, In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 1–8, 2019.

[13] T. Ishiyama, T. Suzuki, and H. Yamana, “Highly Accurate CNN Inference Using Approximate Activation Functions over Homomorphic Encryption”, In 2020 IEEE International Conference on Big Data, pp. 3989–3995, 2020.

[14] J.H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “A Full RNS Variant of Approximate Homomorphic Encryption,” In Proceedings of SAC 2018, LNCS, vol.11349, pp.347–368, 2019.

[15] N. P. Smart and F. Vercauteren, “Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes”, In Public Key Cryptography – PKC 2010, LNCS, vol 6056, 2010.

[16] N. P. Smart and F. Vercauteren. “Fully homomorphic SIMD operations”, Designs, Codes and Cryptography, vol. 71, pp. 57–81, 2014.

[17] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, “CHET: an optimizing compiler for fully-homomorphic neural-network inferencing”, In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019), 142–156, 2019.

[18] Q. Lou. and L. Jiang, “HEMET: A Homomorphic-Encryption-Friendly Privacy-Preserving Mobile Neural Network Architecture”, In Proceedings of the 38th International Conference on Machine Learning, in Proceedings of Machine Learning Research, vol. 139, pp. 7102‑7110, 2021.

[19] R. Dathathri, B. Kostova, O. Saarikivi, W. Dai, K. Laine, and M. Musuvathi, “EVA: an encrypted vector arithmetic language and compiler for efficient homomorphic computation”, In Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020), pp. 546–561, 2020.

[20] Microsoft SEAL (release 3.6), https://github.com/Microsoft/SEAL, Nov. 2020.