5ZC-03

完全準同型暗号を用いたゲノム秘匿情報検索の bootstrap 処理における SSD 適用に向けた検討

辻 有紗 †　　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

近年クラウド上でのデータの機密性を保障したビックデ ータ解析に向けて, 完全準同型暗号 (以下:FHE) を用いた秘密計算技術の実用化に向けた研究が盛んに行われている. FHE の課題として, 時間空間計算量が膨れ上がることが挙げられるが, その中でもボトルネックは bootstrap [1] と呼ばれる処理である. bootstrap とは暗号文同士を演算する際に蓄積するノイズを取り除くため複数の鍵を用いて暗号化・復号化を行う処理を指し, 実行には膨大なメモリ容量を要する. FHE は従来, DRAM を多く使用するためクラウドで活用される手法だが, VM や Container の利用によって 1 台のリソースを分け合うため, 使用効率は 65 %程度にとどまる [2]. その上, DRAM1bit あたりの値段は SSD と大きく異なりコストが高い. そこで, SSD から随時 IO を行い DRAM 使用量を削減することで, 高効率化・低価格化が可能である. また, 並列性の高いアプリケーションであれば, CPU からの load/store 命令の宛先を, DRAM の代わりに並列実行可能な SSD へ割り当てることで高速化される場合がある. 本研究ではゲノム秘匿検索アプリケーション [3] を対象として, bootstrap を中心にアプリケーション内のそれぞれ の処理の傾向に合わせて SSD を用いることで, 実行にかかるコストを削減し, 高速化可能であるか調査を行う.

2 ゲノム秘匿情報検索

アプリケーションについて述べる. クライアントサーバ 型通信で, クライアントがサーバのゲノムビックデータ に対して特定の塩基配列が最長マッチを持つか問い合わせを行う. この時, お互いに持っているゲノム塩基配列や調べたい内容を秘匿しながら行うことが FHE により実現され, 個人情報の漏洩が防がれている. この時, FHE 処理に膨大なメモリ容量が必要な一方で, クラウドでは適当に DRAM が割り当てられないことが多く, スワップ処理が発生する場合がある [2].

Privacy-preserving string search for genome sequences with FHE bootstrapping using SSD
†Arisa Tsuji　†† Tatsuro Endo　†Masato Oguchi
† Ochanomizu University
†† Kioxia Corporation

3 評価環境

CPU	Xeon E5-2643 v3 (3.40 GHz × 6 Cores) × 2
L1 (i,d)cache	32KB, 64B/line
L3 cache	20480KB, 64B/line
DRAM	DDR4, 512GB
Docker1	memory 512GB

表 1: 評価環境

ゲノムデータベースについては 1 サンプルあたり 10000 文字のデータを 2184 サンプル用意した. また, プログラム中で bootstrap 処理の割合を大きくすることを目的に, 検索クエリの長さは 1, データベース上の検索開始ポジション数は 1 に設定した. 4 節の評価は全て Docker1 のコンテナ上で行った.

4 評価

4.1 アプリケーションの傾向

図 1 に実行時間に対する CPU 処理の内訳を示す. initialization では検索するための準備を行い, server は client から受け取った FHE コンテキストや公開鍵の読み込みに殆どの時間を要する. そのため, IO 待ち時間が発生し, idle 値が高く推移している. 続いて bootstrap は Helib ライブラリの recrypt 関数で実装されているが, 前半では IO 処理が重く, 後半は暗号化・復号化処理により引き起こされる CPU の計算処理が重い. bootstrap の内部の処理内容について今後調査を行う必要ある. lookup はゲノム配列データベースに対応するテーブルの更新や暗号化されたクエリとの照合を行う. 終

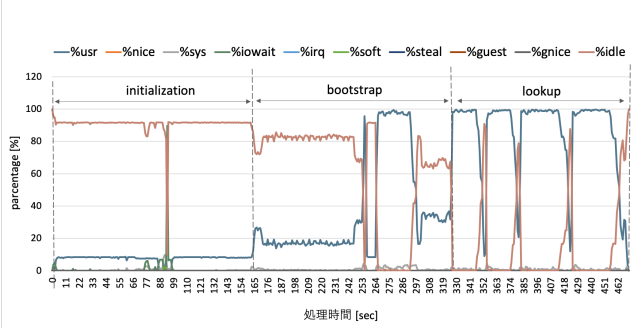

図 1: CPU 処理内訳

始 usr 値が高く推移しており, 計算処理が重いことがわかる. 処理全体を通した各値の平均値ではストレージの I/O コストを示す iowait は処理全体の 0.2 %と低く, メモリサイズが大きく swap が発生しない環境ではストレージ IO は問題とならない. ここで, idle 値が処理全体の 57.51 %を占めており CPU 使用率が低いことから, CPU の計算処理, またはメモリへの load/store 処理でボトルネックが発生していることがわかる. 予備実験として、hyperthread を有効にした結果, 実行時間は増加した. hyperthread のような仮想的な並列化では計算性能は約 2 倍となるが, メモリへの load/store は逐次処理のため, こちらがボトルネックとなり高速化されない. また, 2 スレッドが 1 つのコア上で動作するためタスクが頻繁に切り替わり, コンテキストスイッチの回数は約 2 倍になる (図 2).

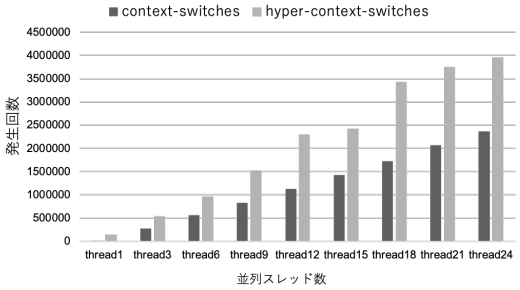

図 2: hyeprthread 有効時の context-switch の推移

4.2 並列化可能性の調査

独立性の高い計算部分を OpenMP を用いてマルチスレッド化し, 実行時間や CPU の負荷の計測を行なった. 図 3 に実行時間の推移と速度向上率を示す. ベンチマークを用いて実験を行ったところ, 計測環境では RAM ディスクへの IO 処理 は, シーケンシャルまたはランダムな読み書きそれぞれについて, 並列数 128 程度でスループットが最大になる. 一方, このアプリケーションについては論理 CPU 数 12 まで並列数に伴い減少し, その後は 増加に転じる. 並列化による CPU コストが変化し, オーバーヘッドが大きいことが原因である. また, 処理の一部は依存性が高く, 逐次処理となるため, 速度向上効率は一部の非効率部分により律速 される. 今回は initialization の並列度が低く, bootstrap と lookup の並列度は非常に高い. ここで, 実際は検索クエリ長やポジション数は大きい値の場合が多く, FHE 処理特に bootstrap 処理の割合が大きくなる. 図 4 に, 検索クエリ長の変化における bootstrap 処理の割合の増加を示す. したがって, プログラム内で並列処理が可能な割合は大きく, 並列 I/O 処理による高速化率は高いと予想される. 並列化による CPU コストの変化が顕著なものの 1 つに context-switch(図 2) があり, 並列数が 1 増加するごとに約 3000 回線形増加する. 今回処理開始時にスレッドプールを設定しており, プリエンプティブマルチタスクの側面を持つため, スレッドプールの数の増加に比例して CPU のスレッド切り替え回数が増加する. また, 並列数の増加に伴い L3 キャッシュミス率が上昇する. 並列に行われる方が短期間にアクセスされるメモリ領域が広いためメモリにないデータにアクセスする頻度が上がることが要因である. ここで, L3 load ミス率に比べ L3 store ミス率の上昇具合が大きいことから, 書き込みの方が新しいメモリ領域を使う傾向があり, 読み込みは局所性が高いことがわかる. 同様の要因により, page-fault も多発する.

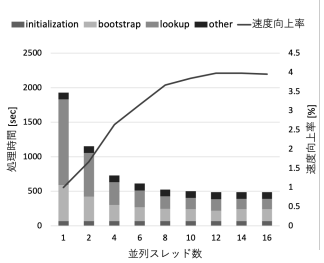

図 3: 並列化による速度向上率

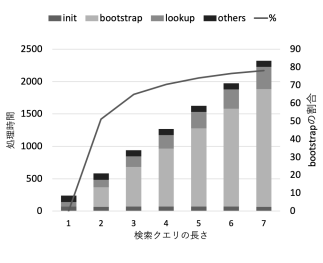

図 4: クエリ長の変化に対する処理内訳の推移

5 まとめと今後の課題

本稿では, DRAM と SSD の良い特性を引き出せるようなハードウェアの使い分けやプログラムの改良により, 高速実行できる可能性について調査を行った. メモリへの load/store 処理が重く, データサイズにより swap が発生しやすい点や, 並列処理可能な範囲が広いことから SSD 並列化が活用できる可能性が高い. 今後は今回計測したデータをもとに,CPU コストの削減方法や 最適な並列化の範囲, DRAM と SSD を階層的に活用するための検討を行う.

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.

参考文献

[1] Craig Gentry. *A FULLY HOMOMORPHIC ENCRYPTION SCHEME*. PhD thesis, Stanford University, 2009.

[2] Muhammad Tirmazi, Adam Barker, Nan Deng, Md.E Haque, Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In *Fifteenth European Conference on Computer Systems (EuroSys '20)*, pp. 1–4, 2020.

[3] Yu Ishimaki, Hiroaki Imabayashi, Kana Shimizu, and Hayato Yanama. Privacy-preserving string search for genome sequences with fhe bootstrapping optimization. In *2016 IEEE International Conference on Big Data (Big Data)*, pp. 3989–3991, 2016.