5H-04

Xeon Phi プロセッサ上での共通鍵暗号 AES の並列処理

Parallelization of AES Cryptography in Xeon Phi Processor

小梁川 龍 † 吉田 明正 †
Ryu Koyanagawa Akimasa Yoshida

1 はじめに

近年，データ通信においてなりすましや改ざんなどセキュリティ脅威は巧妙化の一途をたどり，個人情報をはじめとした重要なデータを安全にやり取りするために，暗号化技術に関心が高まっている．またプロセッサの性能も年々向上しており，より安全性の高い暗号方式が必要になっている．現在では AES が広く利用されており，大規模なデータを暗号化および復号化するためには多くの時間を要するため，高速化の需要が高まっている．AES の並列処理に関する研究として，メニーコアによるファイル単位の暗号化 [1][2]，CPU/GPU を用いてオンデマンドによる AES 暗号化 [3] が提案されている．本稿では Java マルチスレッドおよび OpenMP による AES-CTR の高速化手法を提案する．Intel Xeon Phi 上で行った性能評価の結果，64MB データの暗号化および復号化の並列処理時間が大幅に短縮されており，提案手法の有効性が確認された．

2 共通鍵暗号 AES

AES は DES の後継暗号である．DES は 1977 年に発効してから標準暗号として利用されてきたが，現在では DES のアルゴリズムに関する研究が進み DES の解読に要する時間が短くなったため，実用に耐える強度を維持することが困難になっている [4]．従ってより高い安全性を持つ標準暗号の需要が生まれ AES が 2001 年に発効されて以降，現在でも広く利用されている．AES は長期間の利用に耐えられるようにするために鍵長を 128 ビット，192 ビット，256 ビットの 3 種類から選択できるようになっている．また，一般に AES では以下で例を挙げる各暗号利用モードにより暗号処理を行う．

Electronic Code Book(ECB) モードは最も基本的な暗号利用モードであり，各ブロックを独立に暗号処理する．同一の平文ブロックからは常に同じ暗号文が生成され，並列処理が可能であるが選択平文攻撃の脆弱性があるため，現在では使用が推奨されていない暗号利用モードである．

Cipher Block Chaining(CBC) モードは前後のブロックに依存関係を持たせることによって ECB の問題点を克服した暗号利用モードである．一般に並列的に暗号処理を行うことはできないが，複数の初期ベクトルを用いることで並列実行が可能である [5]．

CounTeR(CTR) モードは乱数を基にしたカウンタを利用し ECB の問題点を克服している．CBC とは前後のブロックに依存関係がない点で異なり並列処理が可能である．

Galois/CounterMode(GCM) は CTR と Galois mode を組み合わせた認証付き暗号のひとつで並列化可能なアルゴリズムである．

†明治大学大学院先端数理科学研究科ネットワークデザイン専攻
Department of Network Design, Meiji University

```
01: static class Threadfunc implements Runnable {
02:   Threadfunc(int core, int allcore){
03:     this.core = core;
04:     this.allcore = allcore;
05:   }
06:   public void run() {
07:     if (core==0)
08:       コア0の処理;
09:     else
10:       コア1の処理;
11:   }
12: }
```

図 1 Java マルチスレッド実装による並列化コードの一部．

3 AES-128-CTR の並列処理

本章では，AES-128-CTR の Java マルチスレッドおよび OpenMP による並列処理について述べる．

一般に AES の並列化を行う際，暗号化処理内部のラウンド処理を並列化する手法や AES の処理単位であるブロック単位での並列処理を行う手法がある．本稿で実装したプログラムでは AES のブロック単位での並列化を行っている．

3.1 Java マルチスレッドによる並列処理

並列コードの実装方法としては，Java マルチスレッドによる並列処理の実装を行なった．Java ではマルチスレッドの実装に Thread クラスを利用する方法と Runnable インターフェース用いて実装する 2 つのの実装方法を用意している．本研究では Runnable インターフェースを用いて Java マルチスレッドを実装した．図 1 は各スレッドの動作を記した Java コードの一部である．図 1 のように各スレッドにそれぞれ並列実行したい処理を割り当てることで並列処理が可能になり処理時間の高速化を行うことができる．

3.2 OpenMP による並列処理

OpenMP による並列コードの実装では，プログラムの並列化したいところに並列化指示文を挿入することで複数個のスレッドが生成され実行される．図 2 は OpenMP による並列化を行うコードの一部である．

4 Intel Xeon Phi 上での AES の性能評価

本章では，Java マルチスレッドおよび OpenMP による AES の並列プログラムの性能評価について述べる．

```
01:  int i;
02:  #pragma omp parallel for lastprivate(encrypt, decrypt)
03:  for(i = 0; i < 1024; i++){
04:    //暗号化関数
05:    encrypt = encryptDecryptCTR(data, key, counter);
06:    //復号化関数
07:    decrypt = encryptDecryptCTR(encrypt, key, counter);
08:  }
```

図 2　OpenMP 実装による並列化コードの一部.

表 1　性能評価に用いる Intel Xeon Phi 搭載マシン.

マシン	Intel Xeon Phi
CPU	Intel Xeon Phi 7250(1.4GHz) (68 コア)
メモリ	48GB
OS	CentOS7.4
Java 処理系	JDK1.8
C コンパイラ	Intel Parallel Studio XE icc 18.0.1

4.1　性能評価環境

本性能評価では Java マルチスレッドおよび OpenMP を用いて実装した AES-128-CTR による 64MB のデータの暗号化および復号化プログラムの性能評価を行う．評価に用いたマシンは表 1 に示した Intel Xeon Phi 搭載マシンである．性能評価方法としては，それぞれのプログラムにおいて 1，4，8，16，32，64 スレッドでの並列実行を行い実行時間を計測した．

4.2　Java マルチスレッドによる並列処理の性能評価

Intel Xeon Phi 上での Java マルチスレッドによる並列処理の速度比較を図 3 に示す．シングルスレッドでの実行時間と比較して 4，8，16，32，64 スレッドにおいて，3.8 倍，7.3 倍，14.0 倍，21.7 倍，33.6 倍の速度向上が得られた．この測定結果から，Intel Xeon Phi 上での Java マルチスレッドによる AES-128-CTR のブロック単位での並列化による暗号化および復号化の高速化において高い実効性能が確認された．

4.3　OpenMP による並列処理の性能評価

Intel Xeon Phi 上での OpenMP による並列処理の速度比較を図 4 に示す．シングルスレッドでの実行時間と比較して 4，8，16，32，64 スレッドにおいて，3.8 倍，7.5 倍，15.0 倍，28.1 倍，57.2 倍の速度向上が得られた．この測定結果から，Intel Xeon Phi 上での OpenMP による AES-128-CTR のブロック単位での並列化による暗号化および復号化の高速化において高い実効性能が確認された．

また，図 3 に示した Java 実装の結果と比較することで C 言語による実装の方が Java 言語による実装よりも実行時間が短く，オーバーヘッドが小さいことが確かめられた．これは Java 言語では Java 仮想マシン用のコードにコンパイルされ，HotSpot 最適化 (JIT コンパイル) が適応されるのに対して，Intel C 言語コンパイラでは，最適なネイティブコードが生成されていることが挙げられる．

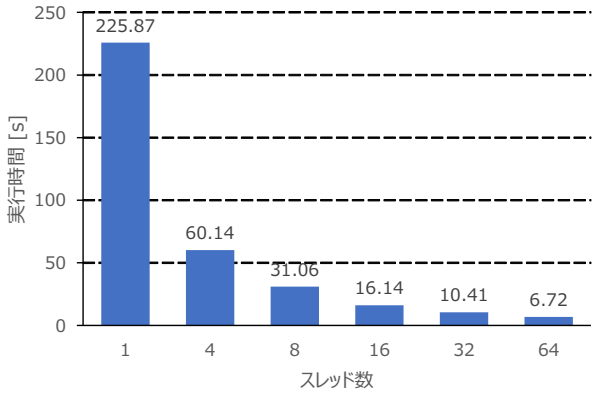

図 3　Java マルチスレッド実装による実行時間.

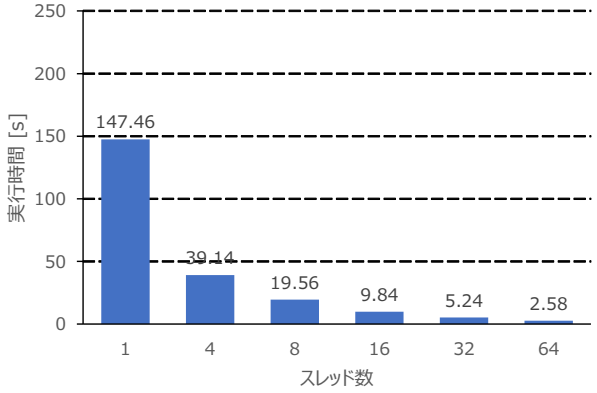

図 4　OpenMP 実装による実行時間.

5　おわりに

本稿では，Intel Xeon Phi 上での共通鍵暗号 AES(AES-128-CTR) の並列処理による高速化を提案する．Java マルチスレッドによる並列プログラムの実装と OpenMP による並列プログラムの実装を行い，スレッド数を変えて実行速度の計測を行った．Java マルチスレッドによる実装ではシングルスレッド実行と比較して 64 スレッド実行時に最大 33.6 倍，OpenMP による実装では 64 スレッド実行時に最大 57.2 倍の速度向上が得られた．

以上の結果から Intel Xeon Phi 上での共通鍵暗号 AES の並列処理において，Java マルチスレッドおよび OpenMP の両方に高い実効性能が確認された．今後の課題としては，AES-256-CTR で並列処理を実現し性能評価を行うことが挙げられる．

参考文献

[1] 白勢政明．メニーコア・コアプロセッサ Xeon Phi での並列暗号実装，情報処理学会，SIG Technical Report, Vol.2015-CSEC-68，No.27，2015.

[2] 白勢政明，吉田努．メニーコア・コアプロセッサによる複数ファイルの並列共通鍵暗号実装，情報処理学会フォーラム，FIT2015，L-041，2015.

[3] 菊池丞太，山口実靖．CUDA による AES 暗号化処理についての考察，情報処理学会，S0731A，2015.

[4] 神永正博，山田聖，渡邊高志．Java で作って学ぶ暗号技術，森北出版株式会社，2008.

[5] 福永武志，平木敬．AES-CTR モードを用いたセキュアな高速シングルストリーム通信，電子情報通信学会，CPSY2015-27，DDC2015-23，2015.