Vol.2015-DPS-162 No.27
Vol.2015-CSEC-68 No.27
2015/3/5

メニーコア・コプロセッサXeon Phiでの並列暗号実装

白勢 政明[1,a)]

概要：本報告は，1つのディレクトリに共通鍵暗号で暗号化すべきファイルが複数 (10 ないし 100 個) 入っている場合，マルチコア CPU やメニーコア・コプロセッサを用いてファイル単位で並列的に暗号化・復号を行う場合の実装評価を提示し，その問題点を考察する．なお，暗号暗号アルゴリズムとして CBC モードの 128 ビット AES を採用し，OpenSSL ライブラリ及び OpenMP を使用してプログラミングを行った．

1. はじめに

本報告の目的は，1つのディレクトリに共通鍵暗号 AES で暗号化すべきファイルが複数入っている場合，マルチコア CPU やメニーコア・コプロセッサを用いてファイル単位で並列的に暗号化・復号を行う場合の実装評価を行うことであり，その経緯は次の 4 つの背景に関係する．

1.1 背景 1: プロセッサのマルチコア化・メニーコア化

トランジスタの微細加工技術の進展のため，1つのプロセッサに搭載できるトランジスタ数 (ゲート数) は，2 年で 2 倍になる，というムーアの法則が知られており，現在も継続している [3]．トランジスタの微細化は，2005 年ぐらいまではプロセッサのクロック周波数の向上の要因であったが，それ以降はプロセッサのコア数の増大の要因となっている [13]．実際に，近年のパソコン用 CPU においては，コア数が 2〜12 程度のマルチコア CPU が一般的になっており，2013 年には Intel Xeon Phi のようなコア数が 60 程度のメニーコア・コプロセッサの流通が開始した．また，Graphic Processing Unit (GPU) においては，コア数が 2,000 以上のものも存在する．

微細化により CPU のクロック周波数の上昇がなされた時期においては，新しい CPU はプログラムの変更なしでそのプログラムの処理速度の向上を達成できたが，微細化が CPU コア数増大をもたらす現在は，プログラムの処理速度の向上には (コンパイラに依るか人力に依るかは別にして) 並列プログラムへの書き換えが必要である．CPU メーカは，少ない負担で並列化コードを生成できるコンパイラを開発している [4]．

既存のプログラムにプラグマを追加して並列プログラムに変更する OpenMP は，既存のプログラムの並列化における比較的容易な手法であり，本報告での実装も OpenMP を利用する．

1.2 背景 2: メニーコア・コプロセッサ Intel Xeon Phi

2013 年に流通が開始した Intel Xeon Phi は，57 から 61 のコアを持つメニーコア・コプロセッサであり，インテル・メニー・インテグレーテッド・コア (MIC) アーキテクチャと呼ばれるアーキテクチャを採用している [13]．インテル MIC アーキテクチャのためのコードは，c 言語や Fortran で記述したプログラムに OpenMP 等による並列処理の指示を追加し，Intel コンパイラで “-mmic” オプションでコンパイルすることで，生成される．また，インテル MIC アーキテクチャは x86 アーキテクチャをベースにしているため，並列化が不要ならば大抵の x86 アーキテクチャのコードを変更無しで Intel Xeon Phi で実行することができる[*1]．このように，既存のプログラムやアセンブリコードから Intel Xeon Phi 用コードを容易に生成する環境が整っている．

1.3 背景 3: 電子図書の著作権保護

近年急速に市場が拡大している電子図書のフォーマットの 1 つである EPUB[1] では，コンテンツはページ単位または章単位で xhtml ファイル化され，それらは 1 つのディレクトリに置かれる．1 つのコンテンツの xhtml ファイル数は数十から数百となる．

EPUB のための基準システムの開発を目的とするプロジェクト Readium において，電子書籍のための軽量な著

[1] 公立はこだて未来大学
Future University Hakodate, 116-2 Kamedanakano, Hakodate, Hokkaido, 041–8655, Japan
[a)] shirase@fun.ac.jp

[*1] 命令セットが完全には一致していないため，x86 アーキテクチャのコードを Intel Xeon Phi で実行できない場合もある．例えば，アセンブリで与えられた GMP ライブラリ [2] の関数のいくつかは，アセンブリを書き換えてコンパイルし直さなければならない．

作権保護技術の実装の必要性から Readium Lightweight Content Protection (LCP) が提案された [11]．Readium LCP は，販売者は電子書籍コンテンツを，ファイル単位で，ユーザ毎の共通鍵で 128 ビット (以上の)AES で暗号化することを推奨している．

1.4 背景 4: OpenSSL

OpenSSL は SSL/TLS プロトコルのためのソフトウェアや共通鍵暗号や公開鍵暗号，ハッシュ関数アルゴリズムのライブラリを提供している [10], [12]．OpenSSL 暗号ライブラリを用いることで，様々な種類の暗号機能を実システムに容易に実装することができる．

1.5 本報告の寄与

本報告では，2 つのディレクトリ名を引数として，1 つ目のディレクトリに入っているファイルすべてに対して，同じ鍵の 128 ビット AES による暗号化/復号ファイルを 2 つ目のディレクトリに出力するプログラムを OpenSSL ライブラリを利用して作成した．次に，このプログラムを OpenMP を利用して並列化プログラムに改良した．最後に，1 つ目のディレクトリに 1 つのサイズが 128B から 4MB のテキストファイルを 10 または 100 個を置き，コア数が 12 の CPU である Intel Xeon やコア数が 60 のコプロセッサ Intel Xeon Phi を使いスレッド数を変えて，並列プログラムを実行したときの処理時間の評価と問題点をまとめる．

2. AES について

共通鍵暗号はブロック暗号とストリーム暗号に大別される．Readium LCP で使用が推奨されている AES はブロック暗号の一種であり，広く普及している．AES のブロックサイズは 128，192，256 ビットである．一般にブロック暗号では，ブロック長より長い平文を暗号処理する場合は，以下で例を挙げる各モードによって，平文をブロック長のブロックに分割し暗号処理を行う．

Electronic Code Book (ECB) モードは，各ブロックを独立に暗号処理する．ECB モードではブロック単位で並列的に暗号処理を行うことができるが，値が同じ平文ブロックは暗号文も同じブロックとなるため，安全でない場合がある．

CounTeR (CTR) モードは，カウンターの値をブロック暗号で暗号化し，それを鍵ストリームとして平文との XOR をとる．CTR モードは暗号処理の並列化が可能である．

Cipher-Block Chaining (CBC) モードは，前後のブロックの暗号文に依存関係を持たせることで ECB モードの欠点を克服し，最も利用されている暗号モードである．但し並列的に暗号化することはできない．

パラメータの種類	パラメータの値
使用プロセッサ	Intel Xeon，Intel Xeon Phi
ファイル 1 つのサイズ (B)	128，256，512，1K，2K，4K，8K，16K，32K，64K，128K，256K，512K，1M，2M，4M
スレッド数	1，2，4，10，20，50，100

表 1 実装実験のパラメータ

3. 関連研究

メニーコア CPU や GPU を利用しての AES の並列化の研究は数多くなされているが，並列化の対象は

(1) 1 回の AES 暗号化の高速化のためのラウンド処理の並列処理
(2) 1 つのファイルの AES 暗号化の高速化のためのブロック単位での並列処理

に大別される．例えば，[7] や [6] は (1) に関する研究で，[5] や [8]，[9] は (2) に関する研究である．なお，(2) では研究では CBC モードは対象とはならない．

しかしながら，電子書籍のコンテンツの暗号化のように複数ファイルを同時に暗号処理を行うという状況もあるため，

(3) ファイル単位での並列処理

も意義のある方針である．本報告は (3) の立場をとる．

4. 本報告の実装

本報告は，2 つのファイル名を引数として 1 つ目のファイル名のファイルを，CBC モードの 128 ビット AES による暗号化/復号ファイルを 2 つ目のファイル名のファイルとして生成する関数 **AES** を作成した．

次に **AES** 関数を使って，2 つのディレクトリ名を引数として，1 つ目のディレクトリに入っている平文ファイルすべてに対して，その暗号化ファイルを 2 つ目のディレクトリに出力する関数 **AES_enc_folder** を作成した．**AES_enc_folder** の復号版の **AES_dec_folder** も作成した．

その次に，**AES_enc_folder** から OpenMP を利用して並列処理できる関数 **AES_enc_folder_parallel_OMP** を作成した．

最後に，表 1 のようなパラメータをそれぞれ組み合わせて暗号化と復号を並列処理しその実行時間を計測した．(紙面の都合上，本報告は暗号化の結果のみを掲載する．) 最後に，この実装結果から考察をまとめる．

なお本報告の実装では，Intel Xeon Phi サーバ Express5800HR120a–1 を使用した．このサーバは，ホスト CPU として Intel Xeon E5-2640(6 コア，2.5GHh) を 2 つとコプロセッサとして Intel Xeon Phi 5110P(60 コア，1.05GHz) を搭載しており，OS は Red Hat Enterprise 6.3

である．コプロセッサの動作周波数は，ホスト CPU の半分以下であることに注意されたい．

4.1 関数 AES

関数 **AES** は 2 つのファイル名を引数として，1 つ目のファイル名のファイルの，CBC モードの 128 ビット AES による暗号化/復号ファイルを 2 つ目のファイル名のファイルとして生成する関数である．これは OpenSSL ライブラリで提供される関数 **EVP_CIPHER_CTX_init** (暗号コンテキストの初期化)，**EVP_CipherInit_ex** (暗号コンテキストの設定)，**EVP_CipherUpdate** (暗号化)，**EVP_CipherFinal_ex** (最終処理)，**EVP_CIPHER_CTX_cleanup** を使って，[10] の Crypto/EVP_Encrypto ページにあるサンプルプログラムを参考に作成された．**AES** 関数の引数は，第 1 ファイル名，第 2 ファイル名，鍵，初期ベクトルの 4 つである．

AES 関数では，**EVP_CipherInit_ex** を

```
1  EVP_CipherInit_ex(&ctx, EVP_aes_128_cbc(),
       NULL, key, iv, do_encrypt);
```

のように使用したが，第 2 引数の EVP_aes_128_cbc() を変えることで，暗号のタイプ，鍵長，モードを変更することができる．

また，EVP_CipherUpdate は引数として，平文のサイズと暗号文を書き込むためのバッファ (のポインタ) を与える必要がある．本報告のプログラムでは，AES 関数内で平文のサイズを計測し，バッファのためのメモリ領域を動的に確保している．

4.2 関数 AES_enc_folder

2 つのディレクトリ名を引数として，1 つ目のディレクトリに入っている平文ファイルすべてに対して，その暗号化ファイルを 2 つ目のディレクトリに出力する関数 **AES_enc_folder** を，**AES** 関数を使って以下のように記述した．

```
1  void AES_enc_folder(char *infolder, char *
       outfolder, unsigned char *key, unsigned
       char *iv){
2      DIR *dir;
3      struct dirent *dp;
4      char original[100];
5      char operated[100];
6      dir=opendir(infolder);
7      for(dp=readdir(dir);dp!=NULL;dp=readdir(
           dir)){
8          strcpy(original,infolder);
9          strcat(original,"/");
10         strcat(original,dp->d_name);
11         strcpy(operated,outfolder);
12         strcat(operated,"/");
13         strcat(operated,dp->d_name);
14         if(*dp->d_name!='.')
15         AES(original, operated, key, iv, 1);
16     }
17     closedir(dir);
18 }
```

以下はこのプログラムの補足である．

- **1 行目，引数について**
  - infolder: 処理したいファイルが入っているディレクトリ名
  - outfolder: 処理後のファイルを入れるディレクトリ名
  - key: 鍵
  - char: 初期ベクトル
- **8〜10 行目**
  “original =処理したいファイル名” となる
- **12，13 行目**
  “operated =処理結果のファイル名” となる
- **14 行目**
  ディレクトリ内のファイル名以外にも “ . ” や “ .. ” があるように動作するため，これらを除外する
- **15 行目**
  関数 **AES** の呼び出し．復号の時は第 5 引数=0

4.3 関数 AES_enc_folder_parallel_OMP

この節の目的は，関数 **AES_enc_folder** を OpenMP を利用して並列プログラムに改良することである．

OpenMP を使って，

```
1  for(i=0;i<n;i++){
2      func()
3  }
```

のような for 文内を並列化するには，

```
1  #include<omp.h>
2  #pragma omp parallel for
3  for(i=0;i<n;i++){
4      func()
5  }
```

のように，ヘッダとプラグマの追加のみで良い．しかしながら，関数 **AES_enc_folder** 内の for 文は上記のような形をしていないため，並列化するには

```
1  for(i=0;i<ディレクトリ内のファイル数;i++){
2      func()
3  }
```

の形にする必要がある．勿論，関数 **AES_enc_folder_parallel_OMP** 内でディレクトリ内のファイル数をカウントする必要がある．その結果，**AES_enc_folder_parallel_OMP** は以下のようになった．

```
1  void AES_enc_folder_parallel_OMP(char *
       infolder, char *outfolder, unsigned
       char *key, unsigned char *iv){
2      DIR *dir;
3      struct dirent *dp;
4      char original[1024][128];
5      char operated[1024][128];
```

```
6     int num=0;
7     char file_list[1024][128];
8     int i;
9     dir=opendir(infolder);
10    for(dp=readdir(dir);dp!=NULL;dp=readdir(
          dir)){
11       num++;
12    }
13    closedir(dir);
14    num=num-2;
15    dir=opendir(infolder);
16    i=0;
17    for(dp=readdir(dir);dp!=NULL;dp=readdir(
          dir)){
18       if(*dp->d_name!='.'){
19          strcpy(file_list[i],dp->d_name);
20          i++;
21       }
22    }
23    closedir(dir);
24
25 #pragma omp parallel for schedule(static)
26    for(i=0;i<num;i++){
27       strcpy(original[i],infolder);
28       strcat(original[i],"/");
29       strcat(original[i],file_list[i]);
30       strcpy(operated[i],outfolder);
31       strcat(operated[i],"/");
32       strcat(operated[i],file_list[i]);
33       AES(original[i], operated[i], key, iv
             , 1);
34    }
35 }
```

以下はこのプログラムの補足である．

- **1 行目**: 引数は **AES_enc_folder** と同じ
- **4 行目**: **AES_enc_folder** と同じく “original =処理したいファイル名” であるが，処理したいファイル名をすべて記憶する
- **5 行目**: **AES_enc_folder** と同じく “operated =処理結果のファイル名” であるが処理結果のファイル名をすべて記憶する
- **6 行目**: “num = ディレクトリ内のファイル数” となる
- **7 行目**: “file_list” に処理したいファイルのファイル名を記憶する
- **25 行目**: OpenMP による並列化の指示

4.4 実装実験

表 1 のようなパラメータに対して，関数 **AES_enc_folder_parallel_OMP** によりディレクトリ内の平文の暗号化を別のディレクトリ内に出力するのに要する時間を計測した．なお，main 関数は以下のようにし，実行時間を計測する方法は [13] を参考にした．

```
1  int main(int argc, char *argv[]){
2     char infolder[] = "plain";
3     char outfolder[]= "enc";
4     unsigned char key[] ="1234567890123456";
5     unsigned char iv[]  ="0123456789abcdef";
6     double start,end;
```

	Xeon				Xeon Phi			
ファイルサイズ \ スレッド数	1	2	4	10	1	2	4	10
128〜256B	0	0	0	0	0	0	0	0
512KB	0	0	0	3	0	0	0	0
1MB	0	0	0	6	0	0	0	0
2MB	0	0	3	10	0	0	0	0
4MB	0	0	2	6	0	0	0	0

表 2 ファイル数 10 での異常発生回数 (100 回中)

	Xeon					Xeon Phi						
ファイルサイズ \ スレッド数	1	2	4	10	20	1	2	3	10	20	50	100
128〜128KB	0	0	0	0	0	0	0	0	0	0	0	0
256KB	0	0	0	0	6	0	0	0	0	0	0	0
512KB	0	0	0	6	21	0	0	0	0	0	0	0
1MB	0	0	0	23	25	0	0	0	0	0	0	0
2MB	1	1	1	33	34	0	3	0	0	0	0	0
4MB	1	0	1	12	23	0	6	0	0	0	1	1

表 3 ファイル数 100 での異常発生回数 (100 回中)

```
7     int threadnum;
8     threadnum=atoi(argv[1]);
9     omp_set_num_threads(threadnum);
10    start=dtime();
11    AES_enc_folder_parallel_OMP(infolder,
          outfolder, key, iv);
12    end=dtime();
13    printf("exection time is %f s\n",end-
          start);
14    return 0;
15 }
```

以下はこのプログラムの補足である．

- **9 行目**: ここで使用するスレッド数を指定する

付録に実装結果を記載するが，暗号処理を 100 回行い，処理時間の平均値，最小値，最大値をまとめている．また，同じパラメータでも平均より暗号処理に数秒以上かかる**異常**が発生した．今回は同パラメータでの処理時間の平均値と 1 秒以上を多く要したものを異常と判定した．異常が存在するパラメータに対しては，100 回中の異常でない (**正常**な) 回数，正常のみの処理時間の平均値と最大値も与える．なお，– の欄は処理時間の計測を行っていない[*2]．

各パラメータに対して異常発生回数を表 2(ファイル数 10 の場合) と表 3(ファイル数 100 の場合) にまとめた．ファイルのサイズが大きいほど異常となる場合が多くなり，また Xeon Phi より Xeon で処理している時の方が異常となる頻度がずっと高い．

ファイルサイズ毎に，暗号処理時間の平均時間 (異常発生の時は正常の平均時間) が最小となるスレッド数とそのスレッド数での平均処理時間を，表 4 にまとめた．ファイルサイズが大きくなると最適なスレッド数は大きくなることが分かる．また，本報告での実装では Xeon Phi より

[*2] 1 つのコアに対してスレッドは 2 つまで，という著者の先入観のため，Xeon(総コア数 12) ではスレッド数 50 と 100 の場合の処理時間を計測しなかった．

	ファイル数 10				ファイル数 100			
	Xeon		Xeon Phi		Xeon		Xeon Phi	
ファイルサイズ (B)	A	B	A	B	A	B	A	B
128	1	2.672	1	20.84	2	8.744	1	40.42
256	1	2.685	1	20.81	2	8.727	1	41.54
512	1	2.783	1	20.97	2	8.755	1	45.95
1K	1	2.949	1	21.32	2	9.018	1	51.05
2K	1	3.145	1	22.23	2	9.873	1	56.03
4K	2	3.288	1	23.99	2	10.22	2	60.84
8K	2	3.857	1	26.99	4	12.34	4	65.33
16K	2	4.815	1	33.89	4	16.16	4	77.49
32K	4	6.469	4	40.74	10	20.99	4	110.9
64K	10	8.541	10	44.74	10	25.17	20	122.0
128K	10	12.19	10	53.31	10	40.73	20	203.5
256K	10	18.11	10	67.72	10	65.87	20	294.6
512K	10	23.50	10	99.17	20	99.05	50	440.3
1M	10	41.98	10	154.7	20	173.3	100	686.7
2M	10	67.44	10	259.6	20	318.1	100	1086.7
4M	10	117.4	10	456.8	20	529.0	100	1588.1

A: スレッド数，B: 暗号処理平均時間 (ms)
注意: Xeon ではスレッド数 50 と 100 を計測していない

表 4 各パラメータに対する最良のスレッド数と，そのスレッド数での暗号処理平均時間

Xeon の方が処理時間が (数倍も) 高速であった．

4.5 考察

処理時間を計測する前は，

(1) Intel Xeon Phi は Intel Xeon よりコア数は 5 倍あるがクロック周波数は半分以下であるため，Intel Xeon Phi の処理時間の相対的速さはわずかである，

(2) 並列処理に必要なオーバヘッド (スケジューリングと同期) のため，ファイルサイズが小さい時は，スレッド数が多いことは効果がないが，ファイルサイズが大きくなると効果が出てくる，

と予想していた．(2) は予想通りであったが，(1) は予想外であった．また，

(3) **異常**が発生すること

は全く想定していなかった．(1) や (3) は本報告で作成したプログラムが原因なのか，プロセッサの構造が原因なのか，今後調査しなければならない．現時点では，並列処理する関数が動的メモリ確保を行っていることが何かしらの影響を与えているのではないかと考えている．

5. まとめと今後の課題

マルチコア CPU やメニーコア・コプロセッサを使って，ファイル単位での AES による並列暗号処理を行った場合処理時間がどうなるか調べた．その結果，少なくとも本報告で作成したプログラムでは，メニーコア・コプロセッサのコア数に匹敵する処理速度は全く得られなかった．また，4.4 節で述べた異常に時間がかかる場合があることが判明した．メニーコア・コプロセッサではこの異常はあまり生じなかったので，安定的に暗号処理を行うときはメニーコア・コプロセッサが適しているかもしれない．

本報告を「ファイル単位での暗号処理」の研究の第一報と位置づけ，今後もこのテーマを継続していく．

謝辞 本研究は JSPS 科研費 25330156 の助成を受けたものです．Intel Xeon Phi への OpenSSL 等のライブラリのインストール作業の補助に対して公立はこだて未来大学の吉田努君にお礼を申し上げます．

参考文献

[1] EPUB，http://idpf.org/epub (2015.02.08).
[2] GMP，https://gmplib.org/ (2015.02.08).
[3] Harris, D. M., Harris, S. L., (鈴木貢，中條拓伯，天野英晴，永松礼夫. 訳)：ディジタル回路設計とコンピュータアーキテクチャ，翔泳社 (2009).
[4] Intel コンパイラ，http://www.intel.co.jp/content/www/jp/ja/developer/software-products.html (2015.02.09).
[5] Iwai, K., Nishikawa, N., and Kurokawa, T.: “Acceleration of AES encryption on CUDA GPU,” Int. J. Networking and Computing, vol.2, no.1, pp.131–145 (2012).
[6] Le, D., Chang. J., Gou, X., Zhang A., and Liu, C.: “Parallel AES Algorithm for Fast Data Encryption on GPU,” 2nd International Conference on Computer Engineering and Technology, *ICCET* (2010).
[7] Manavski, S. A.: “Cuda Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography,” IEEE International Conference on Signal Processing and Communication, *ICSPC*, pp.65–66 (2007).
[8] Nishikawa, N., Iwai, K., and Kurokawa, T.: “High-performance symmetric block ciphers on multicore CPU and GPUs,” Int. J. Networking and Computing, vol.2, no.2, pp.251–268 (2012).
[9] 西川尚紀，岩井啓輔，田中秀磨，黒川恭一：GPU を用いたブロック暗号設計のための性能予測フレームワーク，電子情報通信学会論文，Vol. J97–D No. 12, pp.1740-1754 (2014).
[10] OpenSSL，https://www.openssl.org/ (2015.02.08).
[11] Readum LCP，http://readium.org/projects/readium-lcp (2015.02.08).
[12] Viega. J，Messier M.，Chandra P.，(齋藤 孝道. 訳)：OpenSSL–暗号・PKI・SSL/TLS ライブラリの詳細，オーム社 (2013).
[13] ジム・ジェファース，ジェームス・レインダース，(すが

わらきよふみ. 訳)：インテル Xeon Phi コプロセッサーハイパフォーマンス・プログラミング，カットシステム (2013).

付　　録

A.1　実装結果の詳細

この節の表の見方は 4.4 節の第 1，2 段落を参照．時間の単位は ms.

A.1.1　ファイル数 10 の場合

ファイルサイズ 128B

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	**2.672**	2.929	3.601	6.221	**20.84**	28.85	34.06	44.34
C	**2.438**	2.651	3.186	4.742	**20.06**	27.22	31.28	40.31
D	7.226	**3.104**	4.271	30.58	**22.39**	34.31	39.24	49.10

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 256B

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	**2.685**	2.959	3.640	**20.81**	29.70	33.50	43.36	
C	**2.501**	2.803	3.244	5.848	**19.99**	27.40	31.08	39.51
D	4.063	**3.667**	4.833	9.959	**22.50**	33.52	37.27	49.44

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 512B

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	**2.783**	2.987	3.711	5.766	**20.97**	28.43	34.73	43.20
C	**2.532**	2.815	3.307	4.791	**20.21**	27.27	31.48	39.68
D	8.352	**3.222**	4.906	10.43	**21.97**	30.67	41.05	49.33

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 1KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	**2.949**	3.026	3.660	6.071	**21.32**	29.21	33.63	43.27
C	2.690	**2.357**	3.304	4.777	**20.46**	26.70	31.48	39.40
D	11.76	**3.425**	4.492	24.86	**23.92**	34.00	38.44	48.36

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 2KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	**3.145**	3.153	3.706	6.246	**22.23**	28.70	34.45	43.63
C	2.892	**2.861**	3.317	4.757	**21.38**	27.10	31.39	39.86
D	6.303	**3.462**	4.899	28.71	**23.54**	31.49	39.76	47.86

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 4KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	3.639	**3.288**	3.707	5.875	**23.99**	29.41	32.16	38.55
C	**2.626**	2.925	3.334	4.606	**22.96**	25.97	29.43	34.91
D	5.904	**3.494**	5.826	23.62	**28.12**	32.70	37.74	42.98

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 8KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	4.846	**3.857**	4.100	5.869	**26.99**	31.29	35.11	39.90
C	4.539	**3.378**	3.519	4.445	**26.00**	27.27	31.23	36.02
D	9.905	**4.626**	6.154	14.54	**28.66**	35.08	41.27	46.54

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 16KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	7.105	**4.815**	5.015	6.000	**33.89**	34.87	35.70	39.99
C	6.915	**4.002**	4.336	4.792	32.87	**31.32**	32.88	36.65
D	9.341	**5.246**	6.216	9.848	**38.91**	39.34	41.25	45.14

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 32KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	11.51	6.909	**6.469**	6.873	47.05	42.20	**40.74**	42.46
C	10.14	5.176	**4.987**	5.663	46.14	38.06	**34.60**	37.86
D	19.42	**7.459**	8.558	16.96	51.22	**45.42**	45.74	49.24

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 64KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	17.87	10.63	9.097	**8.541**	74.36	55.52	48.67	**44.74**
C	15.81	7.474	6.972	**6.152**	72.94	51.58	45.17	**40.25**
D	20.05	**13.13**	13.86	15.18	78.32	60.40	56.46	**49.24**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 128KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	26.49	15.45	13.89	**12.19**	130.9	82.94	68.83	**53.31**
C	23.55	12.95	9.451	**9.355**	129.2	80.96	63.54	**49.07**
D	29.73	21.60	19.69	**17.93**	133.1	89.82	73.14	**61.70**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 256KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	43.28	23.63	21.75	**18.11**	242.0	141.8	103.0	**67.72**
C	39.22	21.68	15.36	**12.78**	240.1	137.7	98.69	**62.54**
D	47.45	37.95	29.69	**22.19**	251.6	155.7	112.8	**77.19**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 512KB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	80.44	42.25	**32.76**	263.2	460.6	252.9	168.7	**99.17**
C	79.17	40.21	25.40	**18.11**	458.6	248.8	165.3	**94.28**
D	83.64	48.06	**48.36**	12034.6	464.7	282.7	184.3	**106.2**
E	100	100	100	97	100	100	100	100
F	80.44	42.25	32.76	**23.50**	460.6	252.9	168.7	**99.17**
G	83.64	48.06	48.36	**34.35**	464.7	282.7	184.3	**106.2**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

ファイルサイズ 1MB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	131.1	67.05	**60.59**	645.9	896.3	475.2	324.8	**154.7**
C	128.8	64.61	48.34	**28.75**	894.1	469.2	305.4	**147.7**
D	154.0	**81.86**	88.12	14136.9	917.1	479.4	353.1	**169.9**
E	100	100	100	94	100	100	100	100
F	131.1	67.05	60.59	**41.98**	896.3	475.2	324.8	**154.7**
G	154.0	**81.86**	88.12	247.1	917.1	479.4	353.1	**169.9**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

ファイルサイズ 2MB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	249.0	**130.0**	279.8	683.2	1763.0	940.2	626.0	**259.6**
C	244.4	125.7	79.85	**48.00**	1740.7	909.3	573.1	**247.8**
D	291.2	**152.7**	6085.5	10368.3	1767.7	1048.9	658.4	**280.5**
E	100	100	97	90	100	100	100	100
F	249.0	130.0	103.3	**67.44**	1763.0	940.2	626.0	**259.6**
G	291.2	152.7	162.4	**82.93**	1767.7	1048.9	658.4	**280.5**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

ファイルサイズ 4MB

	Xeon				Xeon Phi			
A	1	2	4	10	1	2	4	10
B	491.4	**252.5**	314.8	452.8	3462.4	1771.9	1121.4	**456.8**
C	479.0	246.8	151.1	**77.40**	3439.1	1755.5	1084.1	**422.3**
D	1080.1	**289.9**	6085.0	6234.9	3472.3	2041.0	1251.3	**487.9**
E	100	100	98	94	100	100	100	100
F	491.4	252.5	198.9	**117.4**	3462.4	1771.9	1121.4	**456.8**
G	1080.1	289.9	268.9	**148.0**	3472.3	2041.0	1251.3	**487.9**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

A.1.2 ファイル数 100 の場合

ファイルサイズ 128B

	A	1	2	4	10	20	50	100
Xeon	B	9.730	**8.744**	12.13	19.46	27.89	–	–
	C	9.080	**5.175**	9.034	15.56	17.55	–	–
	D	15.01	**13.21**	13.57	56.47	87.48	–	–
Xeon Phi	B	**40.42**	55.20	73.36	149.2	148.3	171.7	222.9
	C	**39.43**	52.99	67.65	143.2	141.9	165.9	214.0
	D	42.85	**59.00**	78.84	155.8	153.3	181.3	238.6

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 256B

	A	1	2	4	10	20	50	100
Xeon	B	10.08	**8.727**	11.92	18.86	28.54	–	–
	C	9.783	7.700	**7.547**	14.35	17.49	–	–
	D	15.91	**11.01**	13.16	37.53	67.24	–	–
Xeon Phi	B	**41.54**	59.04	71.59	149.2	148.6	173.7	222.4
	C	**40.35**	56.81	68.24	143.5	144.1	166.8	213.5
	D	**43.07**	62.15	78.52	159.3	154.4	182.3	241.0

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 512B

	A	1	2	4	10	20	50	100
Xeon	B	10.62	**8.755**	12.72	19.29	27.28	–	–
	C	**6.756**	6.789	9.208	16.04	17.53	–	–
	D	13.46	**10.97**	23.08	37.76	66.68	–	–
Xeon Phi	B	**45.95**	58.53	73.06	148.2	148.6	173.0	223.3
	C	**42.63**	53.83	67.65	143.8	143.4	166.4	214.0
	D	**48.51**	63.88	79.22	154.2	156.1	186.2	237.8

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 1KB

	A	1	2	4	10	20	50	100
Xeon	B	11.97	**9.016**	12.04	19.39	28.90	–	–
	C	8.508	**6.424**	9.399	15.34	17.37	–	–
	D	**13.55**	16.87	15.11	48.09	76.26	–	–
Xeon Phi	B	**51.05**	55.84	73.78	148.1	148.7	173.3	222.8
	C	**47.60**	51.49	67.45	143.7	142.0	164.1	211.1
	D	**52.99**	61.59	79.92	154.1	156.0	187.5	238.1

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 2KB

	A	1	2	4	10	20	50	100
Xeon	B	14.50	**9.873**	12.20	19.47	28.45	–	–
	C	11.00	**7.447**	9.120	15.48	17.42	–	–
	D	21.37	19.48	**18.19**	39.42	76.45	–	–
Xeon Phi	B	**56.03**	59.64	73.79	148.6	148.5	171.9	221.4
	C	**54.97**	56.18	69.58	143.1	144.6	166.0	212.5
	D	**61.72**	66.84	79.07	156.3	154.2	180.2	243.5

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 4KB

	A	1	2	4	10	20	50	100
Xeon	B	19.42	**10.22**	10.97	15.95	26.18	–	–
	C	14.06	7.900	**6.843**	14.50	16.34	–	–
	D	**20.02**	21.67	24.83	26.54	86.29	–	–
Xeon Phi	B	69.66	**60.84**	63.38	102.4	102.0	127.8	175.5
	C	68.90	**57.45**	59.04	98.91	95.90	119.4	168.1
	D	70.47	**65.65**	68.67	109.4	109.4	139.5	188.7

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 8KB

	A	1	2	4	10	20	50	100
Xeon	B	22.59	13.29	**12.34**	16.57	24.39	–	–
	C	17.69	10.75	**8.902**	14.77	16.55	–	–
	D	26.17	23.25	**20.11**	32.85	60.15	–	–
Xeon Phi	B	104.5	74.55	**65.33**	101.5	104.2	131.7	180.8
	C	101.1	67.01	**58.72**	95.86	98.10	122.6	169.6
	D	116.1	82.49	**75.53**	111.2	112.1	144.1	196.1

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 16KB

	A	1	2	4	10	20	50	100
Xeon	B	33.16	18.28	**16.16**	17.85	24.79	–	–
	C	31.90	16.01	**11.24**	16.09	16.57	–	–
	D	42.20	39.08	**22.60**	37.91	66.47	–	–
Xeon Phi	B	172.5	108.7	**77.49**	104.5	105.4	133.9	186.8
	C	166.4	100.6	**72.86**	98.52	100.8	129.0	176.8
	D	196.7	118.8	**88.71**	109.2	110.6	142.7	201.4

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 32KB

	A	1	2	4	10	20	50	100
Xeon	B	58.42	28.60	23.12	**20.99**	26.94	–	–
	C	51.86	26.58	16.41	**13.62**	19.37	–	–
	D	59.01	32.94	34.04	**31.19**	77.73	–	–
Xeon Phi	B	301.0	174.7	**110.9**	112.1	111.2	135.6	191.4
	C	297.2	166.2	**104.7**	106.3	105.9	127.5	181.3
	D	306.1	180.7	**120.2**	118.1	119.0	144.4	204.6

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 64KB

	A	1	2	4	10	20	50	100
Xeon	B	99.67	51.15	36.03	**25.17**	29.49	–	–
	C	97.34	48.80	27.50	18.85	**18.24**	–	–
	D	107.2	56.71	51.87	**31.32**	74.56	–	–
Xeon Phi	B	568.5	319.6	192.4	127.0	**122.0**	142.0	205.7
	C	563.5	302.1	176.6	119.1	**118.1**	135.5	194.9
	D	597.2	353.5	198.1	133.2	**130.4**	159.1	239.6

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 128KB

	A	1	2	4	10	20	50	100
Xeon	B	160.4	82.43	63.92	**40.73**	45.28	–	–
	C	158.3	79.76	49.83	29.91	**29.40**	–	–
	D	190.3	95.57	90.55	**51.85**	83.19	–	–
Xeon Phi	B	1130.2	601.3	345.9	215.2	**203.5**	224.8	278.5
	C	1123.3	589.5	323.6	207.7	**191.2**	212.1	264.8
	D	1135.0	683.7	370.7	224.7	**210.7**	262.5	342.3

A: スレッド数，B: 平均値，C: 最小値，D: 最大値

ファイルサイズ 256KB

	A	1	2	4	10	20	50	100
Xeon	B	311.0	159.9	97.88	**65.87**	868.4	–	–
	C	305.9	155.1	82.87	56.07	**49.93**	–	–
	D	355.0	185.8	145.0	**90.91**	14055.0	–	–
	E	100	100	100	100	94	–	–
	F	311.0	159.9	97.88	**65.87**	66.21	–	–
	G	355.0	185.8	145.0	**90.91**	114.9	–	–
Xeon Phi	B	2227.4	1160.3	653.7	335.5	**294.6**	324.5	345.3
	C	2220.6	1150.6	620.8	314.4	**277.4**	304.0	331.8
	D	2230.9	1322.6	703.2	349.2	**319.7**	403.6	421.1
	E	100	100	100	100	100	100	100
	F	2227.4	1160.3	653.7	335.5	**294.6**	324.5	345.3
	G	2230.9	1322.6	703.2	349.2	**319.7**	403.6	421.1

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

ファイルサイズ 512KB

	A	1	2	4	10	20	50	100
Xeon	B	615.0	315.9	**194.1**	768.1	1684.7	–	–
	C	610.9	308.4	163.3	90.45	**87.37**	–	–
	D	642.2	534.2	**281.0**	14089.7	14137.7	–	–
	E	100	100	100	94	79	–	–
	F	615.0	315.9	194.1	109.3	**99.05**	–	–
	G	642.2	534.2	281.0	**122.0**	137.6	–	–
Xeon Phi	B	4415.7	2285.5	1259.5	516.3	443.8	**440.3**	464.2
	C	4410.1	2253.3	1196.4	493.9	424.3	**412.5**	444.2
	D	4431.0	2599.0	1357.7	568.0	**514.9**	526.9	638.0
	E	100	100	100	100	100	100	100
	F	4415.7	2285.5	1259.5	516.3	443.8	**440.3**	464.2
	G	4431.0	2599.0	1357.7	568.0	**514.9**	526.9	638.0

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

ファイルサイズ 1MB

	A	1	2	4	10	20	50	100
Xeon	B	1214.7	630.7	**387.6**	1575.2	1778.4	–	–
	C	1209.8	612.2	320.3	164.0	**162.6**	–	–
	D	1237.4	1035.7	**580.6**	11525.4	11932.4	–	–
	E	100	100	100	77	75	–	–
	F	1214.7	630.7	387.6	203.9	**173.3**	–	–
	G	1237.4	1035.7	580.6	227.8	**203.2**	–	–
Xeon Phi	B	8727.2	4506.6	2465.1	857.9	747.0	695.7	**686.7**
	C	8716.7	4441.1	2369.0	840.1	710.2	671.4	**637.4**
	D	8765.5	5107.1	2697.2	1005.2	**904.9**	968.6	1069.4
	E	100	100	100	100	100	100	100
	F	8727.2	4506.6	2465.1	857.9	747.0	695.7	**686.7**
	G	8765.5	5107.1	2697.2	1005.2	**904.9**	968.6	1069.4

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

ファイルサイズ 2MB

	A	1	2	4	10	20	50	100
Xeon	B	2454.8	1275.9	**800.9**	2392.5	2551.1	–	–
	C	2356.8	1225.8	641.2	351.7	**287.0**	–	–
	D	5409.2	**2368.6**	6559.2	12033.9	12116.7	–	–
	E	99	99	99	66	67	–	–
	F	2425.0	1264.9	742.7	397.0	**318.1**	–	–
	G	3265.6	1812.3	1113.6	443.4	**361.7**	–	–
Xeon Phi	B	17454.5	8987.8	4704.0	1601.7	1357.0	1291.0	**1086.7**
	C	17090.8	8783.7	4589.2	1557.4	1314.8	1170.4	**986.0**
	D	17497.9	10141.1	5265.0	1922.2	1742.5	**1718.3**	1719.5
	E	100	97	100	100	100	100	100
	F	17454.5	8952.4	4704.0	1601.7	1357.0	1291.0	**1086.7**
	G	17497.9	9024.3	5265.0	1922.2	1742.5	**1718.3**	1719.5

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値

ファイルサイズ 4MB

	A	1	2	4	10	20	50	100
Xeon	B	4821.3	2472.3	1447.9	**1340.2**	1952.5	–	–
	C	4745.4	2404.6	1248.2	577.6	**496.3**	–	–
	D	5831.4	3390.8	7155.0	**6818.5**	12054.5	–	–
	E	99	100	99	88	77	–	–
	F	4811.1	2472.3	1390.3	660.0	**529.0**	–	–
	G	4880.2	3390.8	2223.4	738.4	**592.0**	–	–
Xeon Phi	B	34429.8	17475.8	9067.4	2806.7	2315.8	1986.0	**1602.3**
	C	34156.8	17143.6	8806.5	2751.4	2231.7	1937.7	**1461.6**
	D	34511.8	19582.7	10009.1	3257.4	**2934.4**	3048.2	3008.3
	E	100	94	100	100	100	99	99
	F	34429.8	17373.6	9067.4	2806.7	2315.8	1975.2	**1588.1**
	G	34511.8	18201.1	10009.1	3257.4	2934.4	2024.4	**1647.1**

A: スレッド数，B: 平均値，C: 最小値，D: 最大値
E: 正常数，F: 正常データの平均値，G: 正常データの最大値