4K-6

暗号化ZIPファイルにおけるパスワード解析のGPUによる高速化

表 雅之† 大津 金光† 大川 猛† 横田 隆史† 馬場 敬信†
†宇都宮大学工学部情報工学科

1 はじめに

CPUの処理性能の向上が難しい現状から，CPUと比べ高い演算性能を持つGPUを汎用的な目的のために用いるGPGPUが注目されている．GPUのアーキテクチャの特性上，データ並列処理に対しては大幅な速度向上が期待できるが，タスク並列処理に対しては大幅な速度向上は見込めない．よって，GPUを用いたプログラムを書く際には，データ並列処理はGPUで実行し，タスク並列処理については，GPU，CPUどちらで実行するか熟考の上プログラムを作成しなければならない．CPUでタスク処理を行わせる際にはGPUとCPUに協調処理をさせることが出来れば，資源を最大限に活用することができる．協調処理を行わせる場合には，GPU，CPU間で片方に処理が偏らず負荷がなるべく均等になるように処理を分散して割り当てなければ，GPU，CPUのどちらか一方が片方の処理を待つ時間が生まれてしまう．

本稿では，暗号化ZIPファイルのパスワード解析を対象としてGPUとCPUの協調処理による高速化の方法を検討する．

2 GPU

本稿では，NVIDIA社のGPUを扱う[1]．GPUはCPUに比べ数多くの演算器を持つ．また，この演算器をCUDAコア (以前のStreaming Processor) と呼ぶ．CUDAコア1つ1つに複雑な回路を搭載することは困難なため，GPUによって異なるが，8〜48個のCUDAコアに対して1つの命令管理機構を搭載している．そのため，この8〜48個のCUDAコアは全く同じ演算をすることになる．つまりCUDAコア8〜48個がSIMD形式で実行される．また，このCUDAコア8〜48個と命令管理機構をまとめてSM (Streaming Multiprocessor) と呼ぶ．この1つのSMを複数動作させることにより，大量の演算を一度に処理することが可能となっている．しかし，8〜48個のCUDAコアはSIMD形式で実行しているため，CUDAコア毎に異なる処理を実行させると，性能が落ちる．よって，GPUで行う処理はデータ並列処理であることが好ましい．

また，GPUで処理を行うためには図1のようにホストメモリとデバイスメモリ間でデータの転送を行わなければならないので，データ転送を頻繁に行うようなプログラムは性能を向上させにくい．

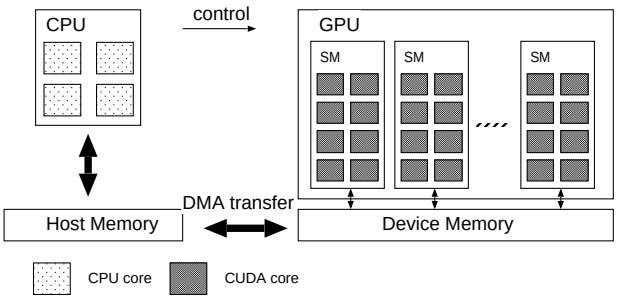

図 1: GPU アーキテクチャ

3 ZIPのパスワード解析

本稿で扱うZIPのパスワード解析ツールは全探索により図2の手順で行われる．まず候補となるパスワードの文字列生成を行う．このパスワード生成はa→b→…→z→aa→…のような順で総当たりでパスワードの解析を試みる．次に生成したパスワードから，キーを生成する．その後ZIPファイルのヘッダ情報を，生成したキーを用い復号する．そして，復号したヘッダの情報から，正解の可能性があるパスワードかどうかの選別を行う．明らかに誤ったヘッダ情報ならば，パスワードの生成へ戻る．正解可能性のあるものならば，生成したキーを用い，ZIPファイルを実際に展開する．その後，CRCの整合性を確認し，正しいものであればパスワードが発見されたことになる．CRCが一致しない場合は，パスワードの生成から再度，解析処理をやり直す．

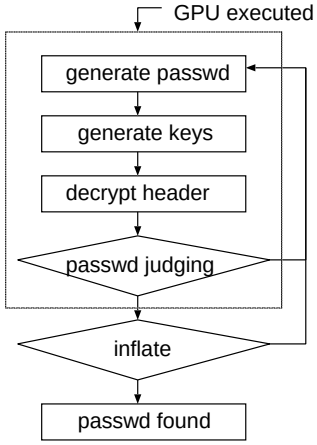

図 2: パスワード解析処理の流れ

Acceleration of password analysis for encrypted ZIP files using GPU
†Masayuki Omote, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota and Takanobu Baba
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

4 GPUによるZIPのパスワード解析の高速化

前述の通りGPUでは，データ並列処理を行うことが望ましい．プログラムの構造を調査した結果，図2の点線で囲まれたパスワードの生成，キーの生成，ヘッダ情報の復号化，およびパスワードの選別はデータ並列処理のため，GPUで実行する．また，タスク並列処理であるZIPファイルの展開処理はCPUにより実行する．この際，GPUとCPUによるパイプライン処理を実現することにより無用な待ち時間を減少させる．図3に示す通り，CPUでZIPファイルの展開処理を行っている間に，GPUを用い，次のパスワードの候補群を生成することで，パイプライン処理を実現している．しかしGPUが行う処理の重さにより，GPUもしくはCPUの一方が早い段階で処理を終了してしまい，待ち時間が発生してしまう．よって，GPUとCPU間でパイプライン処理を動作させる際，適切に負荷分散しなければならない．

次の節でGPUが行う処理の重さを変更し，どの程度速度向上が見込めるか評価する．

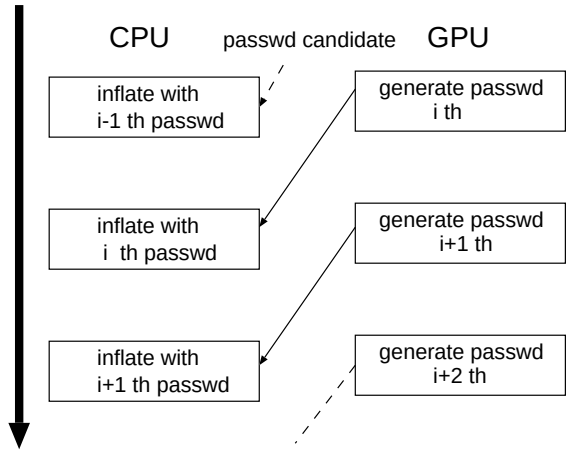

図3: CPUとGPUによるパイプライン処理

5 評価

GPUとCPU間の実行時間が均等になるよう，GPUに割り当てる処理の重さを変えながら10^8通りのパスワードを解析した．以下の表1にツールの評価に使用したGPUとCPUの情報を記載する．また，本稿で使用するTesla C1060はSM内に8個のCUDAコアを搭載しているため，SMの数は30となる．

表1: 実行環境

CPU	プロセッサの数	動作周波数
Intel Core i7 920	4	2.67 GHz
GPU	CUDAコアの数	動作周波数
Tesla C1060	240	1.296 GHz

CPU単体で実行した場合と，GPUとCPUで協調処理を行わせた際の速度向上比を以下のグラフに示す．また，グラフに記載されている1ブロックとは，128個のデータ並列処理のまとまりである．以下の図4から判るようブロック数が1200〜1600の時，最も効率的な負荷分散が出来ることを確認した．

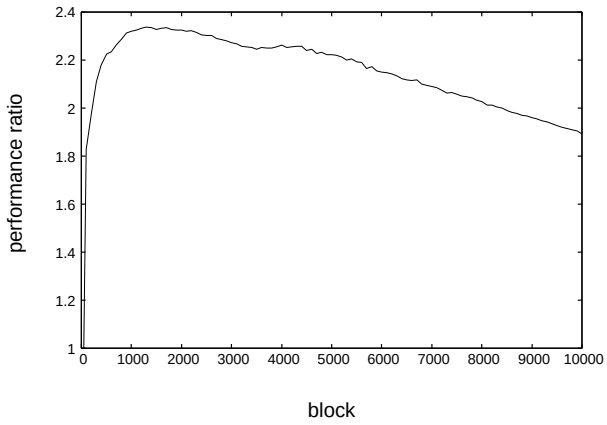

図4: ブロック数に対する実行時間

6 おわりに

本稿では，GPUとCPUの協調処理により，暗号化ZIPファイルのパスワード解析ツールを高速化した．協調処理をさせる際，どちらか一方に待ち時間が生じてしまうと，高速化を妨げてしまうので，待ち時間が発生しない方法を検討した．その結果，GPUに割り当てるブロック数を調整することにより，GPUとCPU間の最も効率的な負荷分散が出来ることを確認した．このブロック数は，このツールを動かす際のGPUとCPUの性能などの違いにより，異なった値となるため，今後は最も効率的なブロック数算出方法を考え，実装し，実行環境に合わせた最適なブロック数で，プログラム実行できるよう改変する．また，今回はTeslaアーキテクチャで動かすことを前提にプログラムを作成したが，Fermiアーキテクチャで動作させることも考え，プログラムを変更していく予定である．

謝辞

本研究は，一部日本学術振興会科学研究費補助金 (基盤研究 (C)21500050，同 (C)21500049) の援助による．

参考文献

[1] NVIDIA CUDA C Programming Guide Version 4.0, http://developer.download.nvidia.com/compute/cuda/4_0/toolkit/docs/CUDA_C_Programming_Guide.pdf