GPGPU でのパスワードクラックツールの実装と評価

笠原 竜大† 村上 智祐† 杉浦 寛†† 羅 鏡栄†† 大釜 正裕†† 齋藤 孝道†
†明治大学 ††明治大学大学院

1 はじめに

従来，GPU (Graphics Processing Unit) は，画像処理を専門に行うサブプロセッサとして利用されてきたが，その高い演算能力やメモリアクセスの性能に着目した，汎用的な用途への利用が広がっている．GPU には複数のプロセッサが搭載されており，それぞれのプロセッサが並列して処理を行うため，並列化できる処理においては，CPU よりも高い性能を発揮する場合がある．

他方，UNIX/Linux のログインパスワードの解析は，複数のパスワード候補に対して同じ処理をそれぞれ独立に行うため，それらの処理を GPU 上で並列化することが可能であり，GPU によるパスワード解析の高速化が期待できる．

そこで，本論文では，NVIDIA 社製の GPU である GeForce GTX 280 を利用して，GPU 上で辞書攻撃によるパスワード解析を行うツールを実装し，その評価を行う．

2 GPU

2.1 概要

GPU は，画像処理を高速に行うことを目的として開発されたマイクロプロセッサである．GPU 上に搭載された複数のプロセッサが同時に並列して演算を行う．従来 CPU 上で処理されていた 3 次元から 2 次元への座標変換，レンダリング，テクスチャの張り込みなどの処理を GPU 上で処理することで，処理時間の短縮，CPU の負荷軽減が可能となる．

代表的な GPU として，NVIDIA 社の GeForce シリーズ，AMD 社の Radeon シリーズなどがある．

2.2 アーキテクチャ

ここでは，本論文で利用した NVIDIA 社の GPU である GeForce GTX 280 のアーキテクチャについて示す．

GeForce GTX 280 は，内部に並列演算器である Streaming Multiprocessor（以降，SM）を 30 個搭載している．各 SM は，スカラ演算器である Streaming Processor（以降，SP）8 個，SP 間で共有される共有メモリ，CPU からの命令を解釈し各 SP へ命令を送信する Instruction Unit などから構成される．GeForce GTX 280 の構成の概念図を図 1 に示す．

2.3 GPGPU

GPGPU（General Purpose computing on GPUs）[1] とは，GPU を画像処理以外の汎用的な数値演算に用いる技術の総称である．初期の GPU アーキテクチャは固定機能のパイプライン処理を行うものであったが，プログラマブルシェーダと呼ばれる技術が導入されると，パイプライン処理がプログラム可能となり，GPU を画像処理以外の処理に用いることができるようになった．また，GPU 向けの開発言語が提供されるようになると，低級言語ではなく，C 言語や C++を用いたプログラミングが可能となった．

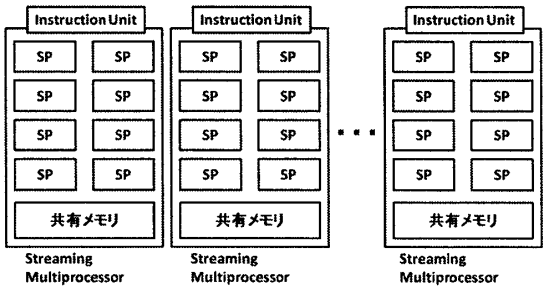

図 1: GeForce GTX 280 の構成の概念図

本論文で利用した開発環境 CUDA（Compute Unified Device Architecture）は NVIDIA 社の提供する GPGPU の 1 つである．CUDA を利用することで，GPU 向けのプログラムを C 言語を拡張した言語で記述することができる．

3 John the Ripper

3.1 UNIX/Linux のパスワード管理

パスワードをそのままファイル上に保存してしまうと，そのファイルが漏洩した際にパスワードも同時に漏洩してしまう．これを防ぐため，UNIX/Linux ではパスワードそのものを保存するのではなく，ハッシュ関数を用いてパスワードをハッシュ処理したものを保存する方法を採用している．このハッシュ処理されたパスワードをパスワードハッシュと呼ぶ．

古い UNIX/Linux では，パスワードハッシュをパスワードファイル（/etc/passwd）という root 以外のユーザも参照できるファイルに記録していた．しかし最近では，パスワードファイルにはパスワード以外の情報を記録し，パスワードハッシュは root のみが参照できるシャドウパスワードファイル（/etc/shadow）に記録する方式が主流となった．この方式はシャドウパスワードと呼ばれ，現在の UNIX/Linux の殆どがこの方式を採用している．[2]

パスワードファイルには，ユーザ名，ユーザ ID，ログインシェルなどの情報が，シャドウパスワードファイルには，ユーザ名，パスワードハッシュ，パスワードの変更期限などの情報が保存されている．

3.2 John the Ripper の概要

John the Ripper[3] は，ログインパスワードの解析を行う，オープンソースのソフトウェアである．解析には，パスワードファイルとシャドウパスワードファイルを結合したものを使用する．以降では，この結合したものをパスワードファイルと呼ぶ．

パスワード解析のモードとして，ユーザ名やホームディレクトリのパス等の情報をもとに解析を行う single モード，辞書攻撃による解析を行う wordlist モード，パスワードと成り得るすべての文字列を試す incremental モードがあり，実行時にオプションを指定することで，これらのモードでパスワード解析を行うことができる．特にオプションを指定しない場合はすべてのモードを順番に実行する．

wordlist モードを実行する場合，辞書ファイルと呼ばれる，パスワード候補となる単語が列挙されたファイルを指定する必要がある．John the Ripper にはあ

† Ryuta KASAHARA, Tomosuke MURAKAMI, Takamichi SAITO
†† Kan SUGIURA, Keang-Weng LO, Masahiro OHGAMA
{kasahara,tomo47,kan_s,oscar_weng,ohgama,saito}@cs.meiji.ac.jp
Meiji University (†)
Graduate School of Meiji University (††)
1-1-1, Higashimita, Tama-ku, Kawasaki-shi, Kanagawa, 214-8571, JAPAN(†) (††)

らかじめ，3000 語程度のパスワード候補が収録された辞書ファイル password.lst が用意されている．また，より単語の収録数が多い辞書ファイルを利用することで解析の精度を上げることができる．

John the Ripper を wordlist モードで実行すると以下の手順で解析が行われる．

(1) パスワードファイルからパスワードハッシュ，ユーザ名などの情報を読み込む．

(2) 辞書ファイルから単語を 1 つ読み込む．

(3) 読み込んだ単語にハッシュ処理を行う．

(4) (3) の結果と，(1) で読み込んだパスワードハッシュを比較する．

(5) 比較した結果が一致した場合，その単語を出力する．

(6) (1) で読み込んだパスワードハッシュを全て解析し終えるか，辞書ファイルを最後まで読み込むと処理を終了する．それ以外の場合は (2) に戻る．

4 GPU を利用するパスワード解析ツールの実装

4.1 実装の概要

John the Ripper の wordlist モードではハッシュ処理を逐次処理で行うため，辞書ファイルのサイズによっては膨大な処理時間を必要とする場合がある．そこで，本論文での実装では処理時間を短縮するため，ハッシュ処理を GPU 上で並列して実行するよう John the Ripper 1.7.3.4 に改変を行った．

また，本論文での実装は MD5 で暗号化されたパスワードのみを解析の対象とし，辞書攻撃によるパスワード解析のみを行う．

4.2 動作

本論文での実装の動作概要を図 2 に示す．以下の動作説明の先頭に振った番号は，図 2 内の番号と対応している．

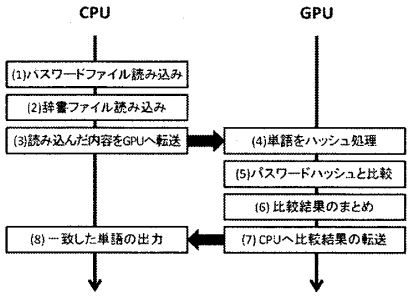

図 2: 実装ツールの動作

(1) パスワードファイルからパスワードハッシュ，ユーザ名などの情報を読み込む．

(2) 辞書ファイルから単語を複数読み込む．

(3) (1)，(2) で読み込んだ内容を CPU から GPU へ転送する．

(4) (3) で転送した複数の単語に対して，GPU 上で並列してハッシュ処理を行う．

(5) (3) で転送したパスワードハッシュと，(4) の結果を GPU 上で比較する．

(6) 比較した結果が一致した単語を，GPU 上でリストにまとめる．

(7) (6) で生成したリストを GPU から CPU へ転送する．

(8) (7) で転送したリストの内容を表示する．リストが空の場合，何もしない．

(9) (1) で読み込んだパスワードハッシュを全て解析し終えるか，辞書ファイルを最後まで読み込むと処理を終了する．それ以外の場合は (2) に戻る．

5 評価

5.1 評価の方法

本論文での実装の評価のため，実験用に作成したパスワードファイルを解析し，その処理時間を計測した．また，比較対象として John the Ripper 1.7.3.4 を使用した場合の処理時間も計測し，計測結果の比較を行った．

5.2 計測

5.2.1 計測環境

計測に用いたマシンのプロセッサは Intel Core 2 Quad 3GHz, メモリは 2GB RAM, GPU は GeForce GTX 280 を 3 枚搭載している．OS は FedoraCore8 kernel2.6.23 である．また，開発環境として CUDA2.1 を使用した．

計測に使用した辞書ファイルは，password.lst に変更を加え，9 番目に現れる文字列”qwerty”を，3079 番目に移動したものである．

計測に使用したパスワードファイルに含まれるログインユーザのユーザ名とパスワードの組を表 1 に示す．ただし，password.lst に収録されておらず，解析できなかったパスワードは”不明”と表示した．パスワードはいずれも MD5 を用いてハッシュ処理されている．

これらの辞書ファイル，パスワードファイルを使用し，処理時間の測定を行った．ここでは，辞書ファイルを読み込み始める直前の時刻と，すべての単語を処理し終えた直後の時刻を計測し，その差を処理時間とした．

表 1: ログインユーザのユーザ名とパスワード

ユーザ名	パスワード
root	”qwerty”
kan	”qwerty”
murakami	”qwerty”
kasahara	”energy”
oscar	不明
user	不明

5.3 計測結果

計測結果を表 2 に示す．本論文での実装の処理時間は 477microsec であり，John the Ripper の約 0.03%の処理時間となった．

表 2: 計測結果

計測対象	処理時間 [micro sec]
本論文での実装	477
John the Ripper	1436552

6 まとめと今後の課題

GPU を利用したパスワード解析ツールの実装と，その評価を行った．今後の課題として，MD5 以外のアルゴリズムへの対応が挙げられる．

参考文献

[1] Hubert Nghuyen 編，中本浩 監訳，"GPU Gems 3 日本語版" 株式会社ボーンデジタル
[2] IPUSIRON 著，"パスワード解析 基礎と実践" DATA HOUSE
[3] Openwall Project, http://www.openwall.com/