平成 29 年度 卒業論文概要			
所　　属	機械情報工学科	指導教員	光來 健一
学生番号	14237017	学生氏名	金本 颯将
論文題目	リモートホストの異常を検知するための GPU との直接通信機構		

1　はじめに

システムの異常には，ハードウェアやソフトウェアの障害をはじめ，ヒューマンエラーが原因の障害や外部からの攻撃といった様々なものが存在する．システムの異常はできるだけ早く検知する必要があるため，障害から復旧するための障害検知，システム性能を維持するための性能監視，攻撃の被害を最小化するための侵入検知などが行われている．従来，ソフトウェアやハードウェアによる検知手法が用いられてきたが，いずれの手法にも問題点があり，高信頼・低コスト・高性能の 3 つを満たすのは難しかった．これらを満たす手法として，汎用ハードウェアである GPU を用いて異常検知を行う GPUsec [1] が提案されている．GPUsec は GPU からメインメモリ上の OS のデータを解析することで異常検知を行う．しかし，GPUsec は検知結果を外部に通知するために OS のネットワーク通信機能を利用している．そのため，OS の内部で障害が発生したり，攻撃によってネットワーク通信を阻害されると，検知結果を通知することができなくなる可能性がある．

本研究では，OS を介さずに GPU と直接通信を行い，検知結果を取得することができるシステム GRASS を提案する．GRASS では，GPU およびネットワークカード（NIC）が正常に動作していれば通信が可能であり，検知結果の通知が障害や攻撃の影響を受けにくい．

2　システムの異常検知

ソフトウェアを用いた異常検知手法では，OS 上や OS 内部で動作する検知システムを用いてシステムの異常を検知することが多い．例として，システムの状態を SNMP で送信して障害を検知する手法や，アンチウィルスを用いてコンピュータウィルスへの感染を検知する手法などが挙げられる．しかし，このような異常検知手法では，OS 内部に異常が発生すると検知を行うことができなくなる可能性がある．例えば，OS が異常停止すると検知システムは動作しなくなる．また，OS の内部にカーネルルートキットがインストールされると，OS 上のセキュリティソフトウェアに偽の情報が返され，以降の攻撃を検知できなくなる．

一方，ハードウェアを用いた異常検知手法では，専用ハードウェアや汎用 CPU の機能を利用してシステムの異常を検知する．専用ハードウェアを用いた例として，PCI バス経由でメインメモリにアクセスして攻撃を検知する手法が提案されているが，ハードウェアのコストが高いという問題がある．また，Intel 製 CPU のシステムマネジメントモード（SMM）を用いて監視を行う手法は，高信頼・低コストという利点があるが，SMM での実行は低速であり，実行中はシステム全体が停止するなど性能面に問題がある．

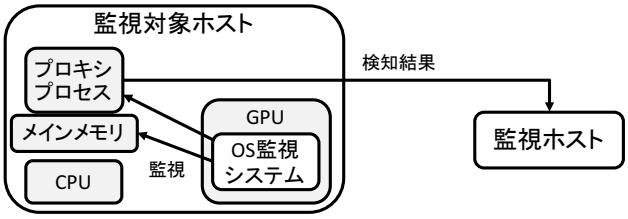

図 1　GPUsec における検知結果の通知

高信頼・低コスト・高性能の 3 つを満たす手法として，GPU 上で異常検知システムを動作させる GPUsec [1] が提案されている．GPU は専用ハードウェアと同様に CPU やメインメモリから独立しているため，異常検知の信頼性が高い．また，GPU は多くの計算機に標準的に搭載されていることから低コストであり，GPU 内の多数の演算コアを用いて並列処理が可能であるため高性能である．GPUsec は図 1 のように，監視対象ホストの GPU 上で OS 監視システムを実行し，メインメモリ上の OS データを解析することで異常検知を行う．検知結果はネットワーク経由でリモートの監視ホストに通知される．しかし，GPU は能動的にネットワーク通信を行うことができないため， GPUsec では OS 上のプロキシプロセスを介して通信を行う．そのため， ソフトウェアを用いた異常検知手法と同様に，OS 内部に異常が発生すると検知結果を通知できなくなる可能性がある．

3　GRASS

本研究では，異常検知対象の OS 等を介さずに監視対象ホストの GPU と直接通信して検知結果を取得するシステム GRASS を提案する．GRASS のシステム構成は図 2 のようになる．GRASS では，GPUDirect RDMA を利用することで，監視対象ホストの OS 等に異常が発生しても GPU および NIC が正常に動作していれば，監視ホストから異常を検知することができる．GPU や NIC に異常が発生した場合には通信を行うことができなくなるが，応答がない場合には異常が発生していると判断することができる．ただし，その場合には異常に関する詳細な情報を取得することはできない．

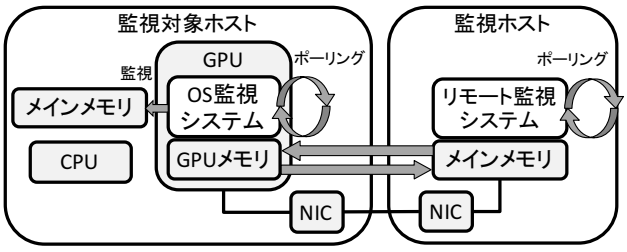

図 2　GRASS のシステム構成

3.1 GPUDirect RDMA の利用

GPUDirect RDMA は，CPU を介さずにリモートホスト上の GPU メモリに直接アクセスするためのハードウェア機構である．GPU の機能を用いて，GPU メモリをメインメモリにマッピングし，NIC の機能を用いてリモートホストからマッピングしたメモリに直接アクセスすることができる．RDMA Write 機構を用いて指定したリモートホストの GPU メモリに直接データを書き込み，RDMA Read 機構を用いて指定したリモートホストの GPU メモリからデータを読み込む．

GPUDirect RDMA を用いるために，GRASS では異常が発生する前に，監視対象ホストと監視ホストの間で接続を確立しておく．まず，監視対象ホストにおいて，RDMA Read/Write に使用するメモリ領域を GPU 上に確保する．次に，監視ホストのアドレスおよびポート番号を指定することで監視対象ホストから RDMA 接続を行い，RDMA Send 機構を用いて GPU メモリのアドレスおよび鍵の情報を送信する．RDMA Send は RDMA Write とは異なり，送信先の CPU を用いた受信処理を必要とする．以後，監視ホストは受信したアドレスおよび鍵を用いて RDMA Read/Write を行うことができる．

3.2 検知結果の要求・取得

検知結果の要求を行う際に，監視ホストは RDMA Write を用いて，監視対象ホストの GPU メモリに対して図 3 のように要求を書き込む．書き込みが完了すると，RDMA Write を用いて GPU メモリ上にある書き込みフラグをセットし，要求の書き込み完了を GPU 上の OS 監視システムに通知する．GPU 上の OS 監視システムは多数ある演算コアの 1 つを利用し，書き込みフラグがセットされるまでポーリングを用いてチェックを続ける．ポーリングを行うのは，RDMA Write の完了を GPU に通知するハードウェア機構が存在しないためである．書き込みフラグがセットされたことを OS 監視システムが検知すると，GPU メモリに格納された要求を取得し，書き込みフラグをクリアする．

受信した要求に従って，OS 監視システムは異常の検知結果を GPU メモリに格納する．その後で GPU メモリ上にある読み込みフラグをセットし，検知結果の書き込み完了を監視ホストに通知する．監視ホストはポーリングを用いて読み込みフラグに対して繰り返し RDMA Read を行う．監視ホストの負荷を下げるために，ポーリングは一定の間隔をあけて行う．読み込みフラグがセットされたことを検知すると，監視ホストは GPU メモリから RDMA Read で検知結果を取得して，RDMA Write で読み込みフラグをクリアする．

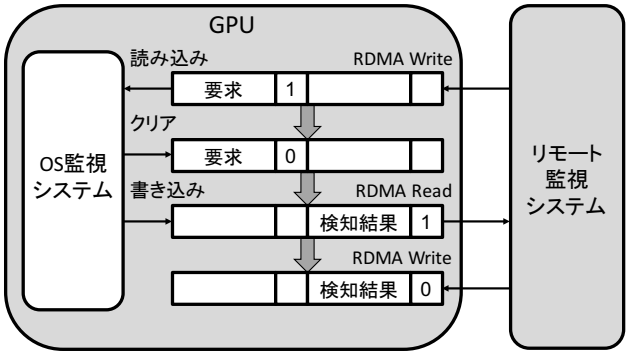

図 3　GPU との通信の流れ

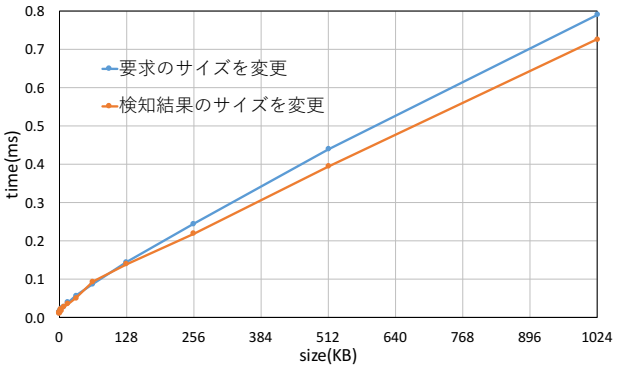

図 4　検知結果の要求・取得にかかる時間

4　実験

GRASS を用いた GPU との直接通信の性能を調べる実験を行った．監視対象ホストとして，Intel Xeon E5-1603 v4 の CPU，8GB のメモリ，NVIDIA Quadro M4000 の GPU を搭載したマシンを用いた．監視ホストとして，Intel Xeon E3-1270 v3 の CPU，8GB のメモリを搭載したマシンを用いた．これらのホストでは Mellanox ConnectX-4 の NIC を用い，100 ギガビットイーサネットで接続した．OS には Linux 4.10 を用い，監視対象ホストでは CUDA 8.0 および nvidia-peer-memory 1.0.5 を用いた．

まず，GPU 上の OS 監視プログラムが応答することを確認するためのハートビートにかかる時間を測定した．ハートビートでは，監視ホストは書き込みフラグのみをセットし，GPU が読み込みフラグをセットするのを待つ．実験の結果，39 マイクロ秒でハートビートを行えることが分かり，十分に短い時間であることが確認できた．

次に，要求および検知結果のサイズを変えた場合の通信時間を調べた．この実験ではそれぞれのサイズを 64 バイトから 1MB まで変化させた．実験結果を図 4 に示す．実験結果から，通信時間は要求と検知結果のサイズに比例し，検知結果のサイズが大きい時より要求のサイズが大きい時のほうが時間がかかることが分かった．

5　まとめ

本研究では，異常検知の対象である OS 等を介さずに監視対象ホストの GPU と直接通信を行い，検知結果を取得するシステム GRASS を提案した．GRASS では，GPUDirect RDMA と呼ばれるハードウェア機構を利用することにより，OS 等に異常が発生しても GPU および NIC が正常に動作していれば，検知結果を通知することができる．

今後の課題としては，CPU 使用率やプロセス情報などの実際の検知結果を通知できるようにすることが挙げられる．また，GPU で検知を行う代わりに，OS データを監視ホストに送って監視ホストで検知を行うことも検討している．

参考文献

[1] 山本裕明. GPU を用いた安全なシステム監視. 九州工業大学 情報工学部機械情報工学科卒業論文. 2016 年 2 月.