卒業論文　2019 年度（平成 30 年度）

卒論

慶應義塾大学 環境情報学部

石川 達敬

徳田・村井・楠本・中村・高汐・バンミーター・植原・三次・中澤・武田
合同研究プロジェクト

2020 年 1 月

卒業論文　2019 年度（平成 30 年度）

卒論

論文要旨

アブスト

キーワード

OS

慶應義塾大学 環境情報学部

石川 達敬

Abstract Of Bachelor's Thesis　Academic Year 2019

Title

Summary

abst

Keywords

OS

Bachelor of Arts in Environment and Information Studies
Keio University

Tatsunori Ishikawa

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

コンピュータの管理者は，動作中，あるいはカーネルパニックなどによって停止したコンピュータの情報を監視・解析することが必要となる場面がある．動作中のコンピュータ自身に対しては，同一ホスト内の top コマンドや ps コマンドを用いて，プロセスの一覧を得たり，gdb コマンドを用いてプロセスをトレースし，プロセスの状態を把握する．論理的に停止したコンピュータに対しては，kdump と呼ばれる機構を通してメモリダンプを静的に解析し，原因の究明をする．また，状態を監視したいホストを物理的なマシンではなく，仮想マシンとして起動し，qemu や Xen などの基盤上で libvmi などを通して状態を解析する手法も存在する．

仮想マシン上に起動したホストに対する解析手段としては，上述の通り手段が豊富に揃っている．

これらの課題を解決する手法の一つとして，RDMA NIC を用いた解析手法が，（sora さんの論文で）発表された．

1.2　着目する課題

コンピュータの状態は，コンピュータ内部におけるレジスタの値および，内部から参照できる仮想アドレス空間上に保持されている．

例えばコンテキストスイッチでは，task$_s$*truct* 構造体から辿れる退避されたメンバから値を取り出すことでプロセス空間および状態の復元を行なっている．

task$_s$*truct* 構造体をはじめとして，*Linux* カーネルの変数や型，関数は，様々なアーキテクチャやカーネルコンフィグに対応するため，マクロによって分岐されている．この分岐が確定するのは，*Linux* カーネルをビルドするときであり，構造体のメンバへのアクセス，関数のアドレスなどはコンパイラが保証している．

実際のカーネルのバイナリは，vmlinux としてコンパイルされた後，strip され bzImage となる．ユーザーが作成したカーネルモジュールなどで関数を呼び出す際は，シンボルとアドレスの変換表である ‘/boot/System.map‘を参照し，実際の

背景で述べた RDMA NIC を用いた解析手法では，メモリの物理アドレスを指定し，逐次的に値を取得し外部から復元していくが，CPU レジスタの現在の値は直接知ることができないため，例えばプロセスの一覧を取得したい場合は，コンテキストスイッチ時に退避された値を辿っていく必要がある．しかし，上述の通り task$_s$*truct* はビルドされた際のカーネルコンフィグによって，どのメンバが先頭アドレスからどのオフセットに保持されているかは変動する．

Linux カーネルのバージョン，System.map の情報および config の情報を知らなければ，メモリのみから正しくコンテキストを復元していくことはできない．

libtlp の論文（もうちょっとちゃんと書く）で紹介されている process-list.c は System.map を引数として渡し，init$_t ask$ の行を $read$ することで $init_t ask$ の仮想アドレスを得ている．また，カーネルコンフィグに依存するマクロの値や，使用する関数などもハードコーディングされており，論文の環境以外で動かすことが容易ではない．

1.3 目的・アプローチ

そこで本研究の目的として，問題の章であげた 3 つの情報，Linux カーネルのバージョン，System.map の情報およびカーネルコンフィグのうち，System.map およびカーネルコンフィグの情報を RDMA NIC を用いて復元する．また，Linux カーネルのバージョンを知ることさえできれば，どのようなカーネルコンフィグを持つコンピュータに対しても，RDMA NIC を通してプロセスリストの一覧を取得できることを実証する．

この目的を達成するため，本研究では，いくつかの段階に分けてネットワーク越しにあるコンピュータに対して監視･解析を行っていく．第一の工程として，RDMA NIC を用いて，監視対象ホストのメモリを全探索し，メモリに落ちている System.map のうち，init$_t ask$ が配置されている仮想アドレス空間に関する情報と，$Linux$ カーネルにおける$_{phys_addr}$ 関数，$task_struct$ 型を決定するためのカーネルコンフィグに関する情報を収集

次の工程として，収集したカーネルコンフィグを元に手元のコンピュータで Linux カーネルのソースコードに対してプリプロセスの処理を行い，task$_s truct$ 型を確定する．さらに，ソースコード上にある$_{phys_addr}$ の実体を収集する．

最後に，この工程で得られた情報をもとに，libtlp で提唱されている手法を用いて，プロセスの一覧を正しく取得できることを確認する．

1.4 本論文の構成

2 章では、本研究で使用する libtlp と，その基盤として使用している RDMA について述べる．

3 章では、RDMA を通してネットワーク越しにあるコンピュータを監視・解析を行うための実行環境の構成に関して述べる．

4 章では、本研究で実装したものについて述べる（加筆）

5 章では、本研究における評価として，Linux カーネルのバージョンのみが与えられた状態でプロセスリストの一覧を取得できることを示す．

6 章では、本研究に関する結論と，今後の課題について述べる．

第2章　関連技術

本章では，本研究で基盤技術として使用する RDMA（Remote Direct Memory Address），および libtlp に関して述べる．

2.1　RDMA

ここに RDMA の説明をかく

2.2　libtlp

ここに libtlp の説明，とりわけ process-list.c で行われている処理に関して書く．

libtlp 本来の目的は，PCIe デバイスの開発プラットフォームであるが，その機能の一つとして，DMA message と ethernet パケットを相互変換する機能があり，それを利用して，RDMA を行っていく，みたいなことをかく

第3章　設計

本章では，本研究におけるコンピュータの構成と，それぞれのコンピュータでどのプログラムを実行するかに関して述べる．

3.1　コンピュータの構成

本研究で実装を行う環境は，RDMA NIC が刺さった監視対象ホストと，本研究における実装を実行するホストの 2 台で構成する．

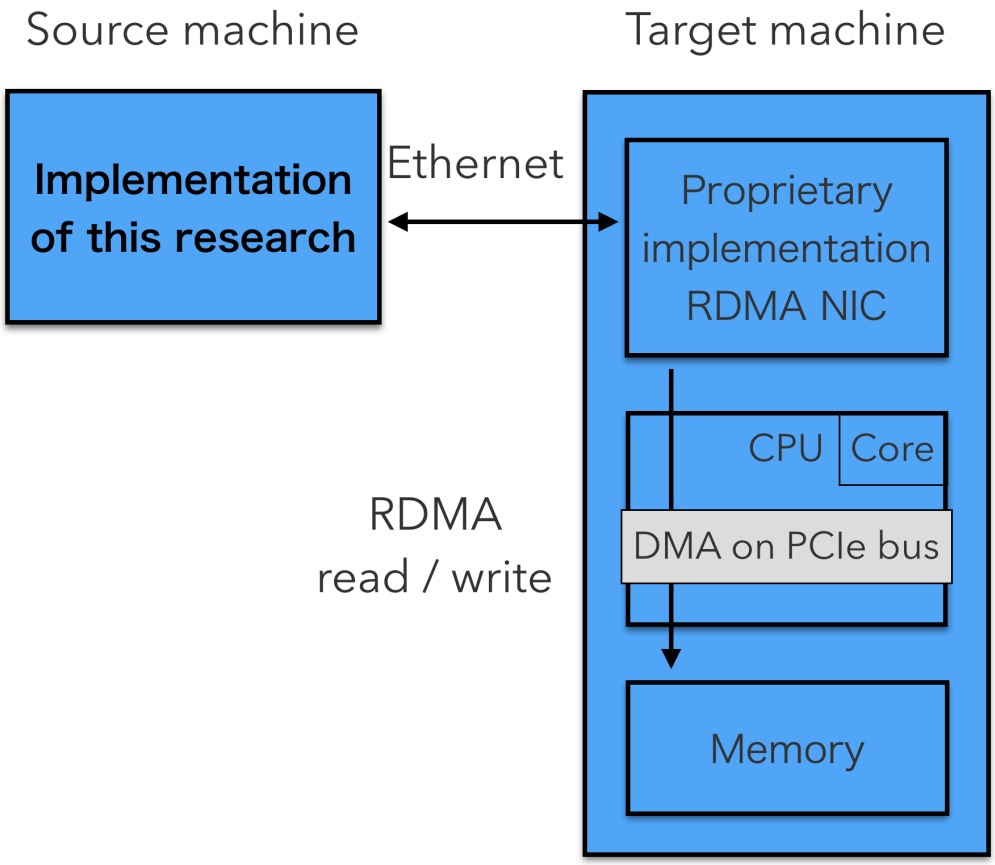
監視対象ホストは，Linux 4.15.0-72-generic の ubuntu であり，PCIe デバイスとして，FPGA ボードが刺さっている．この FPGA ボードは，2 章にて述べたように，dma message と ethernet パケットを相互変換する機能を有しており，インターフェースとして，dma$_{r}ead$ 関数と $dma_{w}rite$ 関数が $libtlp$ に用意されている．また，この $FPGA$ ボードには，IP アドレスとして，192.168.10.1 を静的に振ってある．

実装を実行するホストは，Linux 4.19.0-6-amd64 の Debian buster であり，光ファイバーケーブル (名称はあとで修正) が刺さる NIC を刺している．この NIC には IP アドレスとして，192.168.10.3 を静的に振ってある．監視対象ホストに対して RDMA を実行する際は，dma$_{r}ead$ 関数，あるいは $dma_{w}irte$ 関数を通して 192.168.10.1 に対して IP パケットを送信している．

図 3.1 にて，全体図を書く

3.2　物理アドレスを指定したメモリの値の取得

ここにどのように叩くと値が返ってくるかを書く


Source machine
Target machine
Implementation of this research
Ethernet
Proprietary implementation RDMA NIC
CPU
Core
DMA on PCIe bus
Memory
RDMA
read / write

第4章　実装

実装がまだ完成してないので，空想です．

4.1　実装の全体

ダンプしてくるということを書く．物理アドレスのマッピングに関しても

次の工程として，収集したカーネルコンフィグを元に手元のコンピュータで Linux カーネルのソースコードに対してプリプロセスの処理を行い，task$_s$*truct* 型を確定する．さらに，ソースコード上にある$_p$*hys*$_a$*ddr* の実体を収集する．

最後に，この工程で得られた情報をもとに，libtlp で提唱されている手法を用いて，プロセスの一覧を正しく取得できることを確認する．

4.2　工程 1

第一の工程として，RDMA NIC を用いて，監視対象ホストのメモリを全探索し，メモリに落ちている System.map のうち，init$_t$*ask* が配置されている仮想アドレス空間に関する情報と，*Linux* カーネルにおける$_p$*hys*$_a$*ddr* 関数，*task*$_s$*truct* 型を決定するためのカーネルコンフィグに関する情報を収集することは 2 章で述べた．

（本セクションでは，その実装を詳しく書く．ダンプしまくるやつの説明をここに書く）

4.3　工程 2

収集したカーネルコンフィグを元に手元のコンピュータで Linux カーネルのソースコードに対してプリプロセスの処理を行い，task$_s$*truct* 型を確定する．さらに，ソースコード上にある$_p$*hys*$_a$*ddr* の実体を収集する部分に関する実装をより詳しく書く．

4.4　工程 3

最後に，集められたデータをもとに，process-list を改造したものに関する説明をここに書く

第5章　評価

評価をしたい

5.1　評価手法

カーネルのバージョンのみわかる状態から，正しく ps aux と同じような出力を得られるかどうかを評価とする．

5.2　評価

未評価

第6章　結論

アブスト

6.1　本研究の結論

結論

6.2　今後の課題

課題はたくさんある

謝辞

アドバイスをくれた全員に感謝