IPA

2016 年度　未踏 IT 人材発掘・育成事業　採択案件評価書

1. 担当 PM

竹迫　良範　PM（株式会社リクルートマーケティングパートナーズ　専門役員）

2. 採択者氏名

クリエータ（代表）：佐伯　学哉（東京大学大学院）
クリエータ：西脇　友一（東京大学大学院）

3. 委託金支払額

2,304,000 円

4. テーマ名

ハイパーバイザ技術を用いたクロス OS な Linux バイナリ互換プラットフォームの構築

5. 関連 Web サイト

https://github.com/linux-noah/noah

6. テーマ概要

本プロジェクトでは、近年の CPU に搭載されているハードウェア仮想化支援機構 Intel VT-x 技術を応用することで、Linux アプリケーションを事前の変換や動的な変換を用いずに直接 macOS 上で動かすプラットフォームを構築した。具体的には、Intel VT-x 技術によって作られたハイパーバイザの内部に、カーネルではなく Linux アプリケーションそのものをロードし、ハイパーバイザの機構を用いてシステムコールをトラップ、さらにユーザ空間に通常のアプリケーションとして実装した仮想的なカーネルがこれをホストオペレーティングシステムのシステムコールに変換することで Linux アプリケーションを動作させる。従来のバイナリ互換技術とは異なり、ユーザレベルでシステムの実装が可能であり、また高速かつホストオペレーティングシステムと親和性のあるプラットフォームを実現できるのが特徴である。

7. 採択理由

本プロジェクトの提案時は Bash on Ubuntu on Windows の発表があったタイミングと近しいが、本プロジェクトの手法はそれとは異なる技術的アプローチを用いており、OS X でも Linux バイナリ互換性を達成できるハイパーバイザフレームワークを開発することを目標としている。Intel VT-x 対応の x86_64 アーキテクチャを想定し、独自の ELF ローダで sysenter システムコールをフックし、底の抜けたハイパーバイザとやりとりする。従来このような技術はマルウェアの解析やセキュリティ分野のハックとして知られてきたが、バイナリエミュレーションの分野に応用することで高いバイナリ互換性を実現することができる。開発途中で技術的に困難な課題が発生すると予想されるが、クリエータは過去にいくつかの要素技術を開発しており、システムプログラミング能力も高いため、本プロジェクトを完遂できると考え、採択した。

8. 開発目標

本プロジェクトは Linux ディストリビューションの一つである Ubuntu のパッケージマネージャである apt-get を一切の変更を加えず macOS 上で動作させることを目標とした。独自の ELF ローダやシステムコール変換層を開発した後、適用範囲を広げるため、perl，ruby，gcc などの言語処理系を動かすことや、X11 アプリケーションを実行できるようにすることも新たな目標として再設定した。

9. 進捗概要

Linux 向けにビルドされた xeyes が macOS 上で動作している様子を図 1 に示す。ここでは X サーバとして macOS の半公式アプリである XQuartz を使用している。xeyes は Arch Linux の公式リポジトリからビルド済みパッケージを、pacman を使用してインストールしたものである。Linux のみで動くことを想定されているグラフィカルアプリケーションが一切の変更無しに macOS 上で動作していることが確認できる。Linux 向けにビルドされた 3D グラフィカルゲーム Doom 3 のオープンソースエンジンである dhewm3 を、無修正で macOS 上で動作させることにも成功した（図 2)。

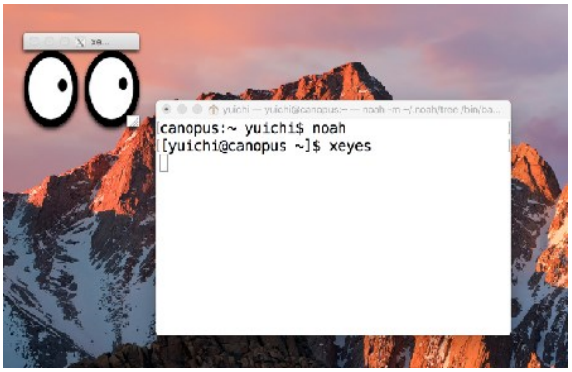

図 1 Arch Linux の xeyes バイナリが動作

図 2 Doom 3 のゲームも無修正で動作

本システムはゲストプログラムからのシステムコールをトラップしてエミュレートする（図 3)。ゲストプログラムがシステムコールを発行するとそれがハイパーバイザによってトラップされゲストプログラムの動作は一時停止し、処理はホストプログラムに移る。ホストプログラムはトラップの原因となったゲストプログラムのシステムコールを解析し（図 3 の上側の write システムコール)、それと等価なシステムコールを macOS に向かって発行する（図 3 の下側の write システムコール)。ホストプログラムでのシステムコールの発行が完了するとその戻り値に応じて、同じ状況で Linux システムコールが返すと想定される戻り値をゲストプログラムに返却する。その後ゲストプログラムを再開するとゲストプログラムからはあたかも Linux カーネルが存在してシステムコールをハンドルしたように見える。

本システムの設計では、ゲストプログラムのプロセスとホストプログラムのプロセスが一対一に対応する。ゲストプロセスがハイパーバイザ内で fork を発行すると、ホストプロセスもホストオペレーティングシステム上で fork を発行し、実際にホストプロセスの数が 2 つに増える。fork によって新しくできたホストプロセスは、新しいハイパーバイザを一つ作り、内部で fork されたゲストプロセスを実行することになる。この設計により、仮想マシンによる完全なカーネルのエミュレーションと違い、本システムはゲストプロセスのスケジューリングや、プロセス間の通信などをホストオペレーティングシステムの機構に任せることができ、非常に軽量な設計を保つことができる。例として「cat file | grep mitou」というコマンドが、本システム上の Bash で実行されたときの動作イメージを示す（図 4)。

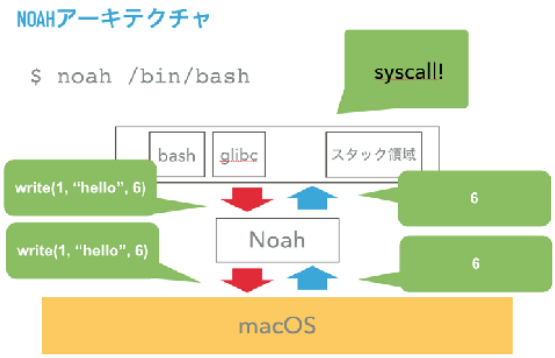

図 3 システムコールトラップの動作

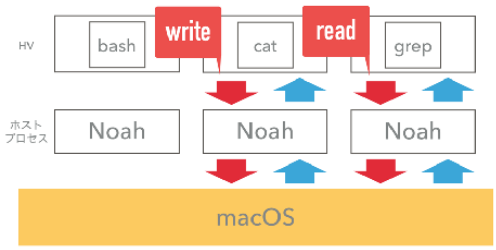

図 4 cat file | grep mitou の動作

Linux アプリケーションを他のオペレーティングシステム上で動作させるシステムとしては、本プロジェクトの他に、Windows Subsystem for Linux, Linuxulator、flinux, User Mode Linux といったシステムが存在する。さらに、Linux アプリケーションを他の OS 上で使用するという目的では、完全な仮想マシンを使って実際の LinuxOS を動かすことも広く行われている。また、Linux とある程度互換性のある POSIX 互換性を達成するシステムとして、Cygwin, MinGW + MSYS2 などが広く用いられている。バイナリ互換、カーネルの修正不要、高親和性の 3 つ観点からこれらのシステムの比較を行うと、本システムは表 1 に示すように多くのシステムに対して優位性がある。

表 1 本システムと関連システムとの比較

	対象OS	バイナリ互換	カーネルの修正不要	高親和性
Noah（本システム）	macOS*	✔	✔	✔
仮想マシン	全てのOS	✔	✔	🚫
WSL	Windows	✔	🚫	✔
Linuxulator	FreeBSD	✔	🚫	✔
Cygwin	Windows*	🚫	✔	✔
MinGW+MSYS2	Windows*	🚫	✔	✔

10. プロジェクト評価

本プロジェクトの掲げる目標はかなり壮大で、クリエータがあれもこれもやりたいとアイディアが発散してしまうと、プロジェクト期間中の実装時間が足りなくなってしまい、トイプログラムのまま実用的な品質まで達せず、プロジェクトが中途半端に失敗してしまうリスクを心配した。PM としてはクリエータに対して「あれもこれもやりたいだろうけど今は我慢しよう」「あえてこの選択肢をやらないことによってコードも綺麗になって開発のスピードも維持できる」など選択と集中のアドバイスに徹することにした。結果、メインの技術以外の周辺のエミュレーションレイヤの開発にも時間を割けるようになり、Linux の X11 グラフィックアプリケーションを macOS 上で実行できるまで完成度を高めることができた。実行パフォーマンスについても他プロダクトとの定量比較を実施して、アーキテクチャの先進性と競合優位性を確保した。開発合宿や他 PM と共同で開催した進捗報告合宿では、都度新しいデモを動かすために必要な機能を追加開発していくというデモ開発駆動の手法が非常に役に立った。大きなプロジェクトを複数人チームで進めていくには、適切なマイルストーンを都度設定し、達成目標のゴールを明確にしておくことが重要ということも学べたかもしれない。

11. 今後の課題

本プロジェクトはオープンソースで開発は進めているものの、開発者である本クリエータ 2 人以外の外部のコミッタやユーザコミュニティが存在しないため、今後どのようにして持続的な開発を続けていくかが課題である。今後も対応するシステムコールの数やオプションの種類を増やし、バイナリ互換の完成度を高めていく必要がある。技術的には非常にモダンで興味深いアーキテクチャのため、開発協力者をさらに増やせるかどうかに鍵がかかっている。英語圏向けの Web サイトを作ったり、論文を投稿したり、Linux カーネル技術者向けの国際会議で積極的に対外発表するなどして、本プロジェクトに興味を持ってくれる開発者仲間を集めて欲しい。本プロジェクトのキラーアプリケーションは何か、という問いに対してもユーザコミュニティと一緒に考え、提案当初は思いもつかなかった新しい使い方を発明して欲しい。