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

— Noah : Hypervisor-Based Darwin Subsystem for Linux—

1.　背景

近年のWebアプリケーションの隆盛やクラウド技術の発展により、LinuxのOSとしての重要性は年々増大している。一方で、開発者の多くは、実際のデプロイ先や業務開発環境であるLinuxではなく、自身の手元の開発環境としてmacOSやWindowsを選択している状況にある。Stack Overflowの2016年度のDeveloper Surveyによれば、現在開発者のデスクトップオペレーションシステムの52.2%をWindows系が、26.2%をmacOS系が占めている。これらの開発者は、Linux上で使用するミドルウェアの各OSへの移植版を使用することで、実際の業務環境であるLinuxとの差異を吸収する努力をしてきた。しかし、この状況には、

1. ミドルウェアのLinuxからの移植がなされるまで時間を要する
2. 移植作業そのものを行うミドルウェアの開発者に負担となる

という2つの問題が依然として存在していた。

2.　目的

本プロジェクトの目的は、全てのOSでLinuxアプリケーションがそのまま動くようにするというアプローチで、上記の2つの問題の解決を図ることである。2016年にMicrosoftが“Windows Subsystem for Linux”によりWindows上でLinuxアプリケーションが動作するようになることを発表し、また、FreeBSDにも以前からLinuxulatorというLinuxアプリケーション動作環境があるなど、近年多くのOSでLinuxアプリケーションが動作する環境が整いつつある。ここに我々が本プロジェクトでmacOS上でLinuxアプリケーションを動作させるアプリケーションを開発することで、Linuxアプリケーションを主要なOS全てで動作するようにする。それにより、先程の問題は

1. ミドルウェアの移植がなくとも、そのままあらゆるOSで動作させることができる
2. ミドルウェアの開発者は移植作業から解放される

という解決を見ることになる。これによりLinuxは次世代の標準開発環境と呼べるものになり、現在その立ち位置にあるPOSIXを置き換える現代的でリッチな次世代開発規格となるであろう。我々は本プロジェクトで、このような環境の実現を目指した。

3.　開発の内容

本プロジェクトでは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、Noahと名付けたソフトウェアを構築した。具体的には、Noah（以下本システム）はmacOS向けのコマンドラインツールとして実装してある。本システムは、コマンドラインの引数でLinuxアプリケーションを受取り、それを一切の修正を加えずにその

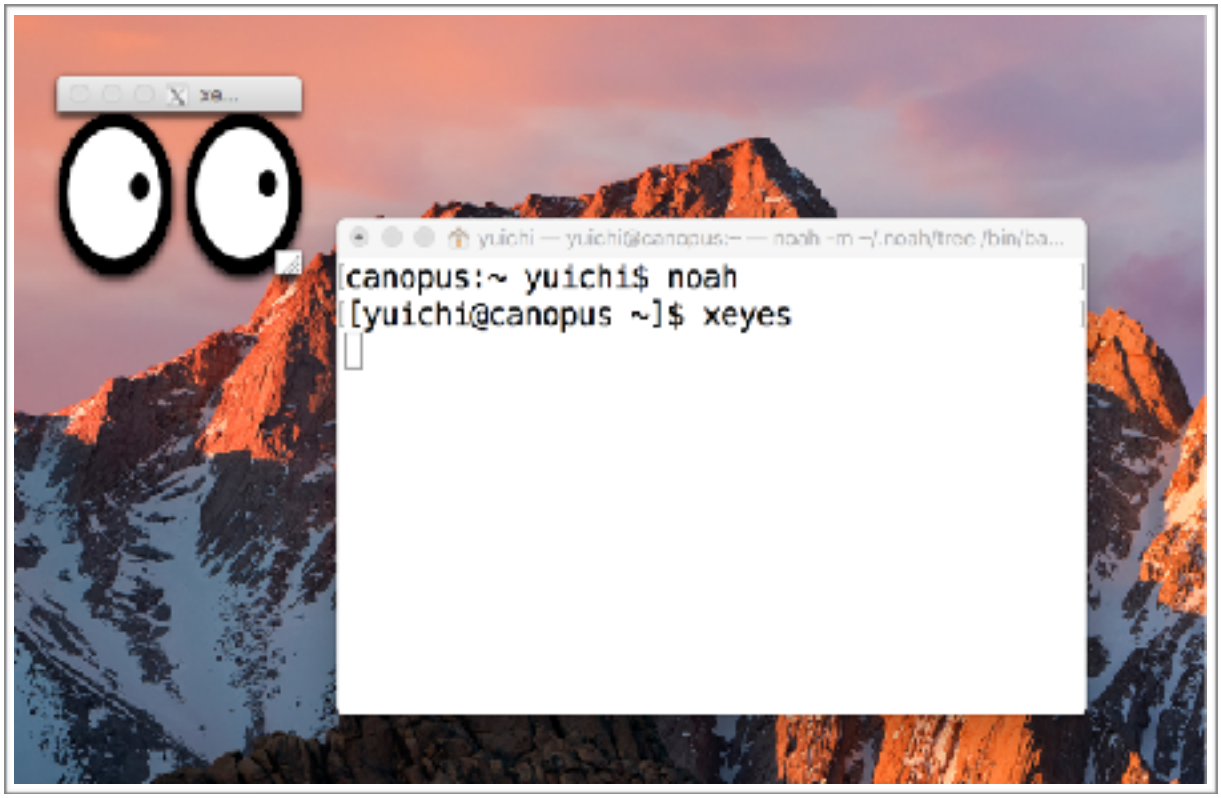

図1. Noah（本システム）でxeyesを起動している様子

ままmacOS上で動作させることができる（なお引数が何も与えられなかった場合はBashシェルが起動する）。図1は、本システムを用いて、Linux向けアプリケーションであるxeyesを起動している様子である。ここではXサーバとしてmacOSの半公式アプリであるXQuartzを使用した。

本システムは次のようなメカニズムでLinuxアプリケーションのmacOS上での実行を可能にする。まず、ホストプログラムは起動すると直ちにApple Hypervisor Frameworkを経由してIntel VT-x技術によって高速化されたハイパーバイザを一つ立ち上げる。その後ホストプログラムは起動時に指定されたLinux用のELFバイナリをそのハイパーバイザ内に本プロジェクトで独自実装したELFローダを用いてロードする。ELFローダが処理を終えるとハイパーバイザ内にはLinux用のELFバイナリが実行可能な状態で待機している。このときハイパーバイザ内にカーネルコードが一切読み込まれていないことに注意されたい。ハイパーバイザの実行を開始すると、Linux用のELFバイナリがCPU上で直接実行される（ハイパーバイザ内で実行されるプログラムを以降、ゲストプログラムと呼ぶ）。ゲストプログラムがシステムコールを発行すると、ハイパーバイザはそれを検知し実行を一時停止し、ホストプログラムへと通知する。ホストプログラムはハイパーバイザ内を検査し、どのようなシステムコールが発行されたかを調べる。ホストプログラムはLinuxカーネルがそのシステムコールを受け取ったときの動作をmacOSのユーザ空間内でエミュレーションし、結果をハイパーバイザ内に返却する。その後ゲストプログラムの動作を再開することで、ゲストプログラムからはあたかもLinuxカーネルがその場に存在してシステムコールをハンドルしたかのように見える。システムコールのエミュレーションを後述するように注意深く設計することで、ゲストプログラムの動作はmacOSに適切な形で反映されるようにできる。このような仕組みで、Linux向けのプログラムが無変更で高速かつ高親和性を持った形でmacOS上で実行可能となる。

本システムはLinuxディストリビューションの一つであるUbuntuのユーザランドを動かすことができ、例えば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上で動作させるほどの完成度を持っている。また、当初の目標に加えて、Ubuntuだけではなくパワーユーザに人気のディストリビューションであるArch Linuxの環境のアプリケーションも動作させることができた。さらにはGUIを実現するのに必要であるXアプリケーションの一部も安定動作させることができ、これらにより、本システムの実際の応用の幅が大きく広がっている。

4. 従来の技術（または機能）との相違

先の節でも述べたように、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macOS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という新しいアーキテクチャを我々は採用している。このアーキテクチャには、次のような3つの利点がある。

1. 高度なエミュレーション
   ハイパーバイザを用いたことで、ハイパーバイザの下では仮想的なカーネル権限を行使することができる。これにより、ホストのカーネルを修正することなくELFアプリケーションを直接ロードすることができ、コピー・オン・ライトなど通常macOSのカーネル権限が必要な機構も実現が可能で、結果として高速な実装が可能である。

2. 高親和性
   Linuxカーネルが独立して動作する完全な仮想マシンとは違い、macOSとの高い親和性を実現することができる。本システムはmacOSから見ればただのアプリケーションであり、例えばネットワークやメモリ、ファイルシステムなどの資源はmacOSによって通常のアプリケーションと同様に管理されている。よって仮想マシンのようにメモリ割り当ての量を考えたり、仮想マシンのネットワーク構成やディスク割当を独立して考えたりする必要はない。また、ゲストプロセスとホストプロセスが1対1に対応するので、シグナルやパイプなど、通常の仮想マシンでは決して共有できなかった資源もスムーズに共有することができる。

3. 高移植性
   本システムのハイパーバイザモジュール部分を除いた大部分はユーザ空間で動作する通常のアプリケーションであり、普通のアプリケーションを移植するのと同じ労力で他のOSに移植することが可能である。これはカーネルの機能として実装されているWindowsのWindows Subsystem for LinuxやFreeBSDのLinuxulatorにはない特徴である。

これらの特徴を踏まえ、関連する他のシステムと比較すると表1のようになり、本システムがアーキテクチャとして優れていることがわかる。

	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*	🚫	✔	✔

表1. 関連システムとの比較

4.　期待される効果

本システムにより、究極的にはエンドユーザは最新のサーバソフトなどのLinux向けミドルウェアをmacOSに移植されるのを待たずに使うことができるようになり、また逆に、ミドルウェアの開発者はNoahをバンドルすることでLinuxアプリケーションから簡単にmacOS用アプリケーションを作成することができる。現状でもPOSIXに準拠しているmacOSに対してはLinuxアプリケーションの多くが移植されると思いがちだが、2017年3月3日現在GitHubで“macOS or osx build fail”で検索すると1,116,686件のissueがヒットすることからも分かるように、macOSへの移植は決して簡単なものでも確実なものでもない。

また、さらなる効果として、本システムの登場と以前から存在するプロダクトの組み合わせにより、macOS, Windows, FreeBSD, Linuxという主要な4つのOSでLinuxアプリケーションが動作することになり、POSIXに替わる次世代の標準開発環境としてのLinuxのエコシステムの活性化につながることが期待される。

5.　普及（または活用）の見通し

今後国際カンファレンスでの発表などを通じて積極的にプロモーションを行う予定である。本プロジェクト期間終了後にはまず初めに2017年5月に開催されるLinuxの国際会議LinuxConでのプレゼンテーションを予定している。その後、技術的詳細をまとめた論文を国際学会に今年の夏に投稿予定である。

6. クリエータ名（所属）

佐伯 学哉（東京大学大学院）
西脇 友一（東京大学大学院）

（参考）関連URL

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