B-029

仮想マシン環境を用いたオペレーティングシステム構成法

An Operating System With Virtual Machine

山本 茂樹†　　**早川 栄一**‡
Shigeki Yamamoto　　Eiichi Hayakawa

1．はじめに

1.1 背景

現在セキュリティを考慮した実行環境が重要になってきている。これは Java などのインストラクションセットを提供する仮想マシン（以下 VM）などの環境では、実行時にプログラムの動作を制御することができ、プログラムの不正な動作を止めることができるからである。

従来 VM はアプリケーションプログラム（以下 AP）の実行環境として、セキュリティが重要なサーバなどのプログラムで主に使用されてきた。しかし、現在のセキュリティの問題は AP だけでなく、ミドルウェアやシステムのメカニズムのバグによる問題によっても起きており、VM による AP だけの保護では、従来 C 言語などで記述されていたプログラムまでも含めたシステム全体の安全な実行環境を提供することはできていない。

特に現在のオペレーティングシステムは動的なモジュールの追加、削除をすることができ、この機能を使ったシステムコールの改ざんの問題などシステムレベルでのセキュリティの問題が起きている。

この問題を解決するため、JavaOS[1]や JX[2]などシステムを VM 上で実現するシステム構成が考えられているが、VM をネイティブコードで記述しており、VM を使ったシステム全体の保護は実現できてはいない。

1.2 目的

ミドルウェア、システムのポリシ、メカニズムを含めたシステム全体の保護ができる環境を構築する。

また、ソフトウェアによる保護によって AP だけでなくシステムのポリシに対しても動的な追加、削除からシステムが保護されることができる環境を構築することを目的とする。

2．設計

2.1 全体構成

本システムは VM を使用してプログラムの実行環境と資源管理を実現する。

システムはハードウェアインタフェース、VM 環境、AP で構成される。システム構成の役割は次のとおりである。

- AP はバイトコードで記述され、AP ごとにスタック、ヒープ、AP の動作をシステムが制限することができるように名前空間を持つ。
- VM 環境はシステムの実行環境と管理機能を提供する。従来の OS が提供していたファイル管理、タスク管理、メモリ管理、デバイス管理などの資源管理と実行環境としてインタプリタ、Just-In-Time Compiler（以下 JIT）を提供する。
- ハードウェアインタフェースは CPU やメモリへのアクセスなど、ハードウェアに依存した処理を VM に提供する

本システムの全体構成図を図 1 に示す。

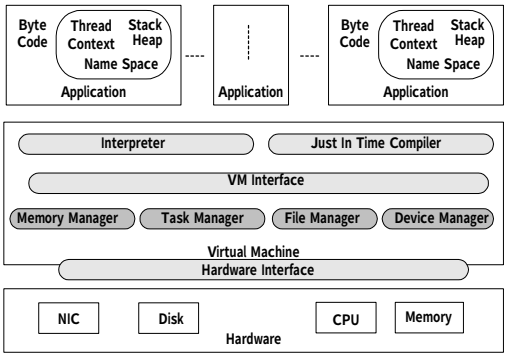

図１　全体構成図

2.2 実行環境

本システムではバイトコードで記述されたプログラムを動作させ、ネイティブコードのプログラムは動作させない。そのため、バイトコードで記述されたアプリケーションを実行するために、実行環境としてインタプリタ、JIT を提供する。

バイトコードで記述されたプログラムをシステムがタスクとして認識する。タスクはスレッドを 1 つ以上持つ。

実行環境をシステムに含めることで、バイトコードで記述された AP のコンテキストを持ち、バイト コードで記述された AP の複数動作を可能とさせる。

2.3 Java 言語による JavaVM

本システムはシステムを高水準言語で記述する。プログラミング言語には Java 言語、VM は JavaVM を使用する。

Java 言語で記述された JavaVM を直接 CPU で動作させるために、Java のバイトコードをネイティブコードへ静的にコンパイルする。これは Java の動作、例外処理や抽象メソッドなどの機能を実現したまま、ネイティブコードに変換する。

また、Java 言語で記述できないもの、メモリやレジスタへのアクセスに関してはハードウェアインタフェースを使ってハードウェアへのアクセスをする。

これにより Java で記述され、Java の機能を持ち実 CPU 上で動作する JavaVM を実現する。

2.4 ハードウェアへのアクセス

VM に対してハードウェアアクセスのインタフェースを提供する。これは Java のクラスとして実装し、Java のインタフェースを提供する。ハードウェアへのアクセスはアセンブリ言語で記述し、これを Java 言語で記述されたインタフェースから呼出すことにより、ハードウェアへのアクセ

†拓殖大学大学院工学研究科
Graduate school of Engineering, Takushoku University

‡拓殖大学工学部情報工学科
Takushoku University

スを実現する。これにより、ハードウェアアクセスコードの集中化をすることができ、システムの移植性を高めることができる。

2.5 システムサービスの呼出し

VM が提供するシステムのサービスは Java のクラスとして VM が提供する。

タスクはオブジェクトを生成し、メソッドを呼出すことでシステムのサービスを得る。

システムのサービスとしては、

・タスク管理
・メモリ管理
・ファイル管理
・デバイス管理

を VM が提供する。

3．保護機構

3.1 ソフトウェアによる保護

本システムではハードウェアに依存しないシステム保護を提供する。ソフトウェアによる保護は、

・メモリへの自由なアクセスを記述することができない言語
・VM による実行時の例外検出

により実現する。

プログラムのメモリへの自由なアクセスができないことにより、ネットワークからの信頼できないプログラムをダウンロードし、実行したとしても、システムを破壊することができない安全な実行環境を提供することができる。

3.2 VM を使ったシステム環境

本システムではシステムの機能を VM が管理する。これは VM がモジュールの動的追加、削除を提供することにより実現する。VM に追加することができるプログラムはバイトコードのプログラムを使用する。これは次の理由からである。

・不正なメモリの書換えができない
・実行前にプログラムが書き換えられていないかを判断できる

このため、VM に対して動的なシステムのポリシ、メカニズムの追加はバイトコードを使い、VM の実行環境、インタプリタ、JIT を使って動作する。これにより、VM 上で動作する AP やミドルウェアだけでなく、安全性を保ったシステムのポリシ、メカニズムの追加、削除が可能となる。

3.3 名前空間による動作制限

AP の実行時の動作をシステムが管理することができるようにタスクに対して名前空間を使った動作制限を可能とさせる。

タスク生成時にタスクごとに名前空間を規定する。これは Java の Package の機能を使って制限する。タスクが使用することが可能、不可能な Package を指定することによりタスクが実行時に生成することができるオブジェクトの使用可、不可を規定する。

実行時の動作の流れは次のとおりである。

1. 名前空間を指定してタスクを生成する
2. インタプリタがタスクを実行する
3. タスクがオブジェクトを生成しようとする
4. 指定されたオブジェクトが生成することが許可されているかをインタプリタが名前空間を調べて判断する
5. 許可されていればオブジェクトを生成し、許可されていなければ実行を止める

これによりシステムはタスクごとにオブジェクトの実行を制限することができ、ネットワークからダウンロードしたプログラムに対しては動作を制限するなどのことが可能となる。

また、すべてのタスクは標準でハードウェアアクセスのクラスの使用を不可にすることで、タスクによるハードウェアへのアクセスを禁止し、システムを保護する。

これを図２に示す。

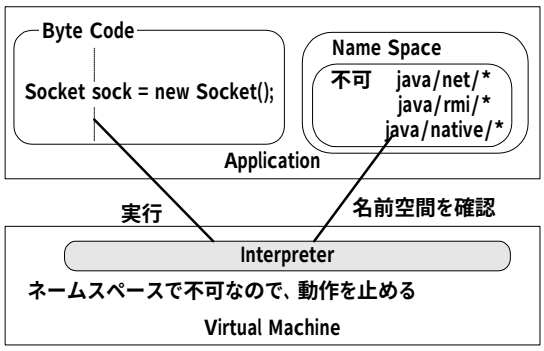

図 2　名前空間による動作

3．実現

本システムは PC のアーキテクチャを使用して実装している。コンパイラには Java をネイティブコードに変換できる GCJ[3]を使用し、Java のライブラリには libgcj を使用している。

Java 言語での実装として、Java 言語による JavaVM、システムのポリシ、メカニズム、ハードウェアアクセスを実装していく。

4．おわりに

仮想マシンを用いたオペレーティングシステムの構成について述べた。

仮想マシンと高水準な言語を使うことでハードウェアに頼らない保護が実現することができ、VM をシステムとして使用することにより、動的なプログラムの追加からのシステムの保護、タスクごとの動作の制限をシステムが与えることができることを述べた。

今後の予定として、システムの実装および実行速度評価をし、本システムの有効性を確認する。

参考文献

[1]Tom Saulpaugh：インサイド JavaOS, ピアソン・エデュケーション
[2]Michael Golm：The JX Operating System, 2002 USENIX
[3]GCJ：http://gcc.gnu.org/java/