仮想マシンを用いた再構成可能なオペレーティングシステム

4L−5

石口栄一†　早川栄一‡　高橋延匡‡
†拓殖大学大学院工学研究科　‡拓殖大学工学部情報工学科

1　はじめに

近年、計算機の高機能化、小型化が進みにコンピュータの利用形態がウェアラブルコンピューティングやユビキタスコンピューティングなどに代表されるように多様化してきている。これにともない、計算機の利用環境や用途に応じてシステムを柔軟に構築するためのフレームワークが求められてきている。さらに、アプリケーションが扱う問題ドメインによって、プログラミング言語の記述性が大きく異なり、異機種間でのポータビリティを確保するため、Java やスクリプト言語のようなインタプリタ言語が注目されている。以下、本稿では2章で要求分析を述べ、3章からは、最小セットのOSと拡張機能をもつ仮想マシンを用いて再構成を容易にできるシステムの構成について主にメモリ管理とスレッド管理について述べる。

2　要求分析

先に述べたような計算機環境では目的や用途に応じてアプリケーションが実行される。そのため、システム自体が様々な用途に対応できるような柔軟性が必要になってくる。また、このような小計算機環境では、メモリ管理ユニットなどを持たないものなどハードウェアにおける機能の違いなどがある。そのため異なるハードウェア上でシステムを構築するとシステム自体の保護に一貫性がなくなる。そこで、我々はシステム自体を用途別に柔軟にかつセキュアに構築するため仮想マシンを用いたアプローチをしている。このような環境では、アプリケーションの要求事項をあらかじめシステムに用意することは不可能であり、アプリケーションの利用目的や用途に合わせてシステムを再構成できる実行環境が求められている。システムに対する要求を次に示す。

（1）用途に適した言語で記述されたモジュールを組み合わせてシステムを構築したい

速度やリアルタイム処理、ハードウェアに依存している部分はC言語などで記述し、インタフェース部はJava言語で、テキスト処理などはPerlのように、システムの各モジュールを用途に特化した言語で記述し、システムを構築したい。

（2）システムを用途別に柔軟かつセキュアに構築したい

システムの用途に応じてシステム構成を容易に再構成でき、かつセキュアシステムを構築したい。

3　設計

3．1　設計方針

仮想マシンとOSを用いて一つのシステムを構築することを考慮しシステム全体として次のような設計方針をとった。

（1）仮想マシンを用いたOSサービスの提供

OSの拡張機能を仮想マシン上で実現することにより、用途に応じたシステムを構築できるようになる。また、仮想マシン上での構築により、プラットフォームに依存しなくなるため、機能のコンポーネントがされやすい。なお、VM/370のような実ハードウェアを多重化する仮想マシンの他にJavaVMやPerlインタプリタも広義で仮想マシンとして考えることができる。

（2）システム全体を3層に分けて構築する

OSが管理する資源には、用途によって異なる資源と共通の資源がある。共通の資源としては、プロセッサやメモリ、割込みなどがある。そこで、このような基本機能やハードウェア固有の機能とそれ以外の拡張機能と分離する。このため用途に応じたシステムを構築しやすくなる。なお、デバイスドライバは速度や応答性に応じてOS層と仮想マシン層に置かれる。

3．2　全体構成

本システムでは論理的に、OS層、仮想マシン層、アプリケーション層の3層により一つのシステムを構築する。図1に本システムの全体構成を示す。

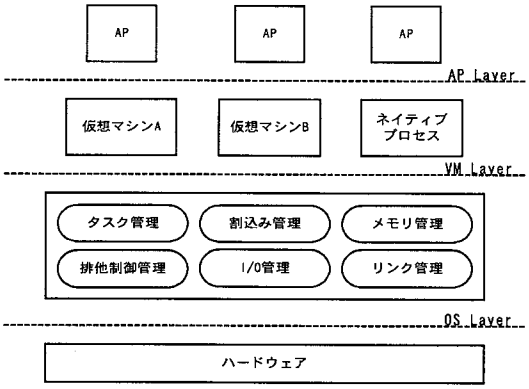

図1：システムモデル

3．3　OS層

OS層では、ハードウェアと仮想マシンとのインタフェースであり、ハードウェアの制御をし、仮想マシンの実行環境の基盤を提供する。つまり、プロセッサなどのハードウェアアーキテクチャを仮想マシン層に対して低レベルな抽象化をして、低レベルなプロセッサに依存した機能のプリミティブを提供する。ここでは、タスク管理やメモリ管理、排他制御管理、割込みコントローラによる割込みや例外ベクタの管理、I/Oレジスタへのアクセス、タイマ、キャッシュの制御や仮想マシンからネイティブコードのリンク処理やライブラリのロード処理などを提供する。そのため、OS層ではメカニズムだけを提供するだけであり、仮想マシン層

A reconfigurable operating system using virtual machines
Eiichi ISHIGUCHI†, Eiichi HAYAKAWA‡ and Nobumasa TAKAHASHI‡
† Graduate school of Engineering, Takushoku University
‡ Department of Computer Science, Faculty of Engineering, Takushoku University

に対してはサービスマネージャのような振舞いをする。また、管理ポリシなどは仮想マシン上で実行されるプログラムとして定義される。

3．4　仮想マシン層

仮想マシン層では、OS 層により抽象化し定義されたメカニズムに対してポリシを与える。さらにアプリケーションに対しての実行環境としても動作する。また、複数の仮想マシンを提供することにより、アプリケーションや現在の実行環境の特性に合った仮想マシン上で実行することにより最適化することを可能にする。OS 層で提供されたメカニズムに対して、仮想マシン層でのポリシを差し替えることによりアプリケーションに合わせた最適化をする。これは、従来の OS が提供する大きな粒度の構造化を用いた場合とことなり、アプリケーションの要求により柔軟に対処することが可能になる。

4　詳細設計

ここでは、仮想マシン層に JavaVM を用いたプロトタイプの構成についてついて述べる。

4．1　メモリ管理の設計

OS 層でのアドレッシングの基本単位は物理アドレスであり、仮想記憶のサポートは想定していない。メモリヒープ領域には OS 用ヒープと仮想マシン用ヒープ２種類あり、それぞれを利用するには OS 層が提供するメモリ管理の異なる API を用いる。次に OS 用メモリプールと仮想マシン用メモリプール利用 API についてそれぞれ示す。

（1）OS 用メモリプール

この OS 用メモリプールを利用する API はネイティブコードによる動的なメモリ割当てのためにある。これは JavaVM のランタイム環境と OS のネイティブコードで利用される。この API で確保されたメモリはガーベジコレクタにより制御はされない。また、この領域はポインタベースのヒープ領域として管理される。

（2）仮想マシン用メモリプール

この仮想マシン用メモリプールを利用する API が管理するメモリ領域は Java のクラスオブジェクトが保持される。また、このメモリ領域は、ガーベジコレクタによりガーベジのため監視される。そのためこの API により確保されたメモリ領域は Java プログラミングモデルによりアクセスされる。JavaVM 上からのメモリアクセスモデルは Java のクラスによりカプセル化され、そのクラスからネイティブメソッドを通じてメモリにアクセスされる。

4．2　スレッド管理の設計

スレッド管理では、スレッドの生成、削除、起動、終了、停止、スケジューリングなどのサービスを提供する。スレッドには JavaVM で実行されるものと低レベルのイベントを処理するもの、アプリケーションの一モジュールとして実装されるものがある。また、JavaVM 上のスレッドはネイティブスレッドと一対一にアタッチされる。図２に Java スレッドとネイティブスレッドの関係について示す。

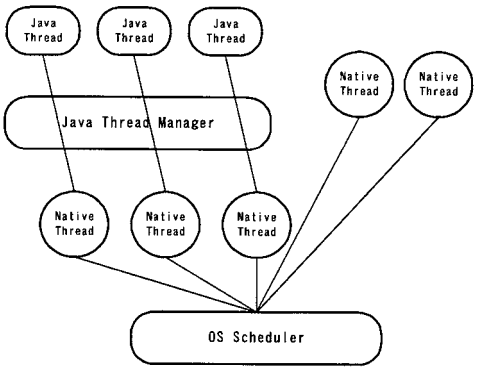

図２：Java スレッドとネイティブスレッドの関係

JavaVM からのスレッドの生成をすると OS のスレッド生成が呼ばれる。Java のスレッドは OS により Java スレッドという属性をもったタスクとしてタスク制御ブロック（TCB）のリストに格納される。このように各タスクはどのように何から生成されたかのという情報を TCB にもつ。

5　関連研究

仮想マシンコンセプトを用いてOS自体をモジュール化，コンポーネント化し、システムの拡張性、柔軟性を提供する研究としては Fluke [1]、MMLite[2]などが存在する。Fluke は再帰的仮想マシンコンセプトをOSに適用し、OS の拡張に，MMLite は各種言語の実行環境の提供に利用している。Fluke では、OS に対する拡張が仮想マシンとして提供され、仮想マシンを再帰的に定義することでアプリケーションに適したシステムに拡張することができる。MMLite は、コンポーネントのインタフェースとしてCOM (Component Object Module)を採用しており、様々な言語で記述された COM モジュールを実行するために仮想マシンを提供し、それらの仮想マシンを調停する役をする。これに対して本システムの目指すところは用途に応じて複数の実行環境を利用することにより再構成を容易にするシステムである。

6　おわりに

本稿では、仮想マシンコンセプトを用いてシステム自体をアプリケーションに応じて容易に再構成を可能にするシステムの構成について述べた。また、JavaVM を用いて仮想マシンのプロトタイプについてスレッド管理とメモリ管理について述べた。

今後はより多様な用途に応じて仮想マシンを用意しより柔軟にシステムを構築できるようにする。

参考文献

[1] Bryan Ford, et al, Microkernels Meet Recursive Virtual Machines, In Proc. of the 2nd OSDI, pp.137-151 (1996)

[2] Johannes Helander, et at, MMLite：A High Compon- rentized System Architecture, In Proc. of the 8th ACM SIGOPS European Workshop, pp.96-103 (1998)

[3] 高野了成, et al：永続オブジェクトを共有するための仮想マシンインタフェースの設計，マルチメディア，分散，協調とモバイルシンポジウム，pp.561-566 (2001)