# 仮想マシン機構を用いたオペレーティングシステムの実現

4U−01

石口 栄一† 高野 了成†† 早川 栄一‡ 高橋 延匡‡
†拓殖大学大学大学院工学研究科 ††東京農工大学工学部 ‡拓殖大学工学部

## 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．

また，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合わせて一つの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

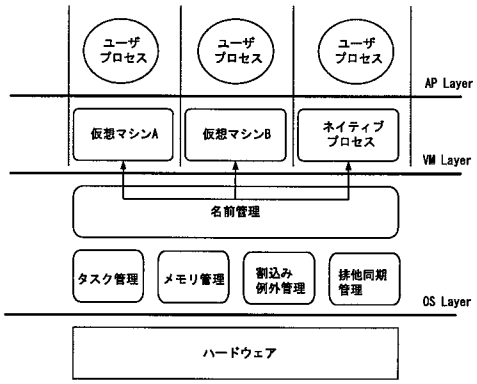


図 1: 全体構成図

## 2 設計方針

先に述べた要求より次のような設計方針を立てた．

(1) 仮想マシンインタフェースを提供する
アプリケーションの実行環境として仮想マシンを提供し，ネイティブコードの実行やスクリプトをインタプリトする．また，仮想マシン上で動作するアプリケーションはその一つの仮想マシン上だけで動作するのでは，相互に呼び出すことが可能である．
(2) 複数の実行環境を提供する
複数の実行環境を仮想マシンとして提供することにより，異なる言語で記述されたモジュール同士を組合わせて一つのシステムを構築する．

## 3 設計

### 3.1 全体構成

本システムの全体構成を図 1 に示す．このシステムはアプリケーション層，仮想マシン層，OS 層 (AP Layer, VM Layer, OS Layer) から構成されている．仮想マシンを用いてシステムを構築する場合，OS の役割としては，主に複数の仮想マシンから透過にみえる名前空間とモジュールのバインディング機構を用意すればいい．

### 3.2 OS 層

OS 層は，ハードウェアと仮想マシンとのインタフェースであり，ハードウェアを抽象化する．また，OS 層は 2 層から構成されており，下層をハードウェア抽象化層と呼ぶ．ハードウェア抽象化層を提供することによりハードウェアの差異を吸収することにより，OS 自体の移植性を向上させることができる．

また，上層でおいてはタスク管理やメモリ管理などの実際に仮想マシン層から使われる機能を提供する．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，システムコールとして上位に提供する．

具体的には次の機能を提供する．

(1) タスク管理機構
(2) メモリ管理機構
(3) 排他同期管理機構
(4) 割り込み管理機構
(5) 名前管理機構

### 3.3 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．

本システムで対象としている仮想マシンは，Mach における Unix エミュレーションのような OS パーソナリティや JavaVM や Ruby インタプリタのようなものも広義の仮想マシンと考えている．

### 3.4 タスク管理機構

OS 層では，プロセッサをタスクとして抽象化する．タスクは自身のスタックとコンテキストを持った実行実体である．タスク管理機構では，単純なスケジューラを提供し，複

---

Implement of Operating System using Virtual Machine Concept

† Eiichi ISHIGUCHI
†† Ryousei TAKANO
‡ Eiichi HAYAKAWA
‡ Nobumasa TAKAHASHI
Graduated school of Engineering, Takushoku University (†)
Tokyo University of Agriculture and Technology (††)
Department of Computer Science, Faculty of Engineering, Takushoku University (‡)

雑なスケジューリングポリシは仮想マシン上のポリシにより与えられる．つまり，この二つのスケジューラの協調によりスケジューリングポリシを提供する．

### 3.5 メモリ管理機構

メモリ管理機構では，動的に確保されるメモリ領域を管理する．具体的には，固定長メモリプールと可変長メモリプールの二つのメモリ領域を管理する．次に二つのメモリ領域について説明する．

(1) 固定長メモリプール
固定長のメモリブロックを提供するメモリ領域であり，固定長のメモリブロックを提供する．

(2) 可変長メモリプール
可変長メモリプールでは，要求されたサイズに応じたメモリブロックのための領域であり，一つのブロックに対して一つの管理領域を必要とする．

### 3.6 排他同期管理機構

OS 層では，タスク単位で実行を管理している．そのためタスク単位での排他・同期制御が必要になる．そのため，排他・同期のためにセマフォを提供する．

これは，バイナリセマフォあるいは計数型セマフォとして使用する．そのため一般的な排他・同期の問題を解決できる

### 3.7 割り込み管理機構

割り込みを扱う実装方式としては，raw handler をそのまま提供するか，Unix の signal のようにメッセージとして抽象化するかの二つが考えられる．raw handler を提供する方式では，性能がでるがセキュリティが問題になる．そこで，本システムでは，OS 層でのタスクで割り込みを扱う．

また，本システムは，アプリケーション層，仮想マシン層，OS 層により階層化されている．そのため，割り込みなどを上位の層に対して通知してやる必要がある．その仕組みをアップコールと呼ぶ．割り込みは，メッセージとして上位の層に対して通知される．具体的には，send と receive の二つのプリミティブを用いて割り込みを抽象化している．

### 3.8 名前管理機構

仮想マシン同士の相互呼出しを可能にするためには，OS 層において，呼出し関係を管理する必要がでてくる．そのため，OS 層では名前空間とそれを用いて名前を検索してバインドする機構が必要になってくる．そこで，OS 層において各仮想マシン上のモジュールにおける外部識別子を公開するためにシステムで一意の名前空間を提供する．本システムでは，データ構造としてシンボル (名前) と手続き (データ) と型をもった関数ポインタ表を用意し，名前検索のための名前空間を木構造として提供する．名前空間の木構造と名前管理機構のデータ構造について図 2 に示す．

## 4 実現

前述の設計に基づいて OS 層を実現した．OS 層では，移植性を考慮してハードウェア依存の処理を切り分けており，OS 層の内部を 2 層にして実現した．また，OS 層で提供する単純なスケジューラとしてプリエンプティブなラウンドロビンスケジューラを実装した．

開発環境としては，実機として SH3 を搭載した組込み用ボードである CAT68701 を用いて実現した．クロス開発環境としては Linux2.1.5+gcc2.95.3 を用いた．また，ブート環境としては sh-linux で使われている IPL(Initial Program Loader) である sh-ipl-g を用い，ertheboot を利用した．

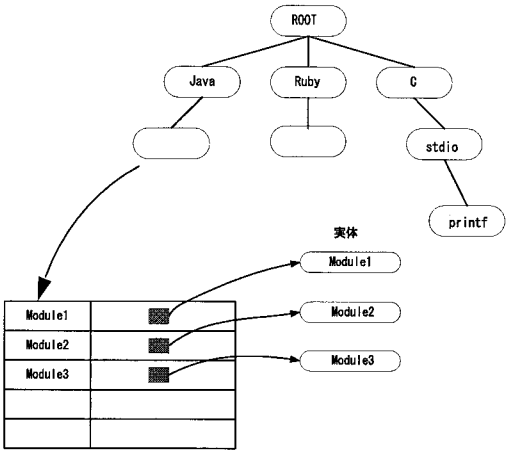


図 2: 名前管理機構

## 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使い分けることにより，アプリケーションや用途に応じて適合できるシステムである．

## 6 おわりに

本稿では，仮想マシンを用いたオペレーティングシステムの設計と実現について述べた．仮想マシン機構を用いることにより，複数の実行環境を提供でき，システムをアプリケーションの用途に応じて適合できる．

また，我々は，永続オブジェクトを共有するために仮想マシンインタフェースを提供している [3]．

今後の課題としては，複数の仮想マシンを用いて実用的なアプリケーションでの評価があげられる．

## 参考文献

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

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

[3] 高野了成，石口栄一，佐藤元信，早川栄一，高橋延匡，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).