3H－6

２次元アドレスを管理するマイクロカーネルの実現と評価

森永智之，早川栄一，並木美太郎，高橋延匡
（東京農工大学　工学部）

１．はじめに

計算機の用途が多様化するにしたがって，OS には多様な資源管理を行うための資源管理の拡張性が要求されている．我々が研究を行っている OS/omicron V4[1]（以下V4）では，OS をマイクロカーネルとサーバに分離し，資源管理の拡張性を確保する．また，ワンレベルストアを採用して資源管理を容易にする．

しかし，既存のマイクロカーネルではメッセージ通信によるオーバヘッド，ワンレベルストアを実現する際の保護，システムの信頼性や拡張性と速度とのトレードオフに応じて保護を設定できない，という問題がある．したがって，マイクロカーネルには高速な拡張手法，ワンレベルストアを容易かつ安全に実現できる基盤，そして柔軟な保護機構の提供が求められている．

これに対して我々は，拡張手法としてダイナミックリンクを採用し，関数コールでシステムを構築することで高速化を図る．また，保護とワンレベルストアを容易に実現するため，２次元アドレスを採用する．なお，ターゲットプロセッサは，セグメンテーションとページングを備えた 80x86 や PA-RISC を想定する．

我々はマイクロカーネルのプロトタイプを行い，その結果から，ダイナミックリンクのための実行環境，２次元アドレス，柔軟な保護機構を提供するマイクロカーネル[2]の設計を行い実現した．

２．プロトタイプの問題点

プロトタイプ[3]では，プロセッサを複数個に仮想化して提供する仮想マシンと，多重２次元アドレス空間を提供した．しかし，次のような問題点が発生した．

(1) 多重２次元アドレス空間の問題

セグメント id と実体との対応が一意でなくなり，メモリ管理の複雑化，保護の問題が発生した．

(2) 仮想マシンインタフェースの問題

サーバに提供されるインタフェースが低レベルなため，サーバを実現することが困難であった．

(3) カーネル呼出しのオーバヘッドの問題

カーネル呼出しをトラップで行ったため，コンテキストスイッチのオーバヘッドが存在した．

上の問題点を解決するためマイクロカーネルの設計を行った．

３．マイクロカーネルの設計方針

マイクロカーネルの設計方針は次のとおりである．

(1) 単一２次元アドレス空間を提供する

プロトタイプでは多重２次元アドレス空間を提供したため，セグメントの管理と保護の問題が発生した．このことから，単一２次元アドレス空間を提供する．

(2) 柔軟な保護機構を提供する

単一２次元アドレス空間の保護機構を提供する．さらに，個々のサーバ，サーバ全体というように，保護の対象を自由に設定できるインタフェースにする．

(3) タスクを管理する

仮想マシンのインタフェースは低レベルであったため，マイクロカーネルでタスクを管理し，サーバの実現を容易にする．

(4) マイクロカーネル自体の拡張性を確保する

OS 構成法の研究基盤を提供するため，スケジューリングポリシーやコンテキストを変更可能にする．

(5) マイクロカーネル呼出しにゲートコールを用いる

ゲートコールを用いて，マイクロカーネル呼出し時のコンテキストスイッチをなくし，高速化を図る

セグメントとタスク以外の，ダイナミックリンカ，ファイルシステム，デバイスドライバなどはマイクロカーネルの外に出すことにした．これはワンレベルストアを実現するためと，拡張性を確保するためである．

４．マイクロカーネルの設計

マイクロカーネルは，次の特徴を持つ．全体構成を図１に示す．

(1) 単一２次元アドレス空間と柔軟な保護機構の提供

マイクロカーネルでは単一２次元アドレス空間を提供する．これによって，セグメント id と実体の対応を一意にし，セグメント管理を容易にする．また，これによる保護の問題を解決するため，保護空間と呼ぶ

Implementation and evaluation of a micro kernel with a 2-dimensional addressing space
Tomoyuki MORINAGA, Eiichi HAYAKAWA, Mitarou NAMIKI and Nobumasa TAKAHASI
Tokyo University of Agriculture and Technology

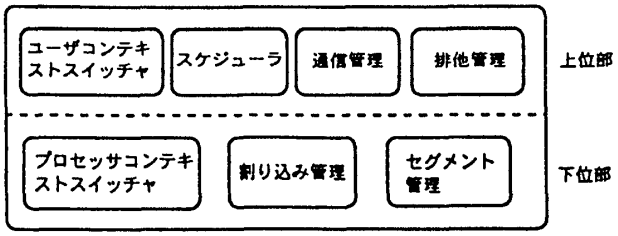

図 1 マイクロカーネルの全体構成

機構を提供することにした．この保護空間は単一な2次元アドレス空間であり，アクセス可能なセグメントの集合を設定できる．この保護空間の割当てのポリシーはマイクロカーネルで規定しない．これにより個々のサーバに対して保護空間を割り当てたり，複数のサーバに対して一つの保護空間を割り当てることが可能になり，トレードオフに応じて保護を設定できる．

(2) ダイナミックリンクを行うための実行環境の提供

我々が設計を行った2次元アドレス，ダイナミックリンクに対応した実行環境は複数のセグメントを持つ．マイクロカーネルではこの複数セグメント上で動作するタスクを管理し，サーバの実現を容易にする．また，割込みをタスク間通信として仮想化することで，ドライバの記述を容易にする．ただし，高速性を要求される割込み処理については，割込みハンドラのインタフェースを提供することで対応する．

(3) マイクロカーネル自体の拡張性の確保

マイクロカーネルを上位部と下位部に分け，上位部でタスクとしての仮想化を，下位部でプロセッサの管理を行い，上位部をブート時にリンクすることで拡張性を確保することにした．このリンクは通常の関数コールにすることで，層分けのオーバヘッドを少なくする．下位部では，プロセッサを管理する．これをプロセッサコンテキストと呼び，下位部では，プロセッサコンテキストの切替え，変更，セグメントの生成，消去といった機能を提供する．上位部ではタスク間通信や割込みの仮想化，スケジューリングなどを行う．

下位部の SVC の一部を表 1 に示す．

表 1 マイクロカーネルの下位部 SVC

```
プロセッサコンテキスト型  *プロセッサコンテキストを確保する(void)
int          プロセッサコンテキストを解放する(
                        プロセッサコンテキスト型 *)
void         プロセッサコンテキストを切替える(
                        プロセッサコンテキスト型 *)
int          セグメントを生成する(セグメントid, セグメント属性)
int          セグメントを消去する(セグメントid)
保護空間id   保護空間を生成する(void)
int          保護空間を消去する(保護空間id)
```

5．マイクロカーネルの性能測定

マイクロカーネルの下位部を実現し，次の点について測定を行った（486DXII 40MHz 上）．

・ゲートコールとコンテキストスイッチの時間

カーネル呼出しをゲートコールで行った場合と下位部のコンテキストスイッチの時間

・割込み通知の時間

割込みハンドラの場合と，上位部で割込みを仮想化する場合に下位部で経過する時間

この測定結果を表 2 に示す．

表 2 下位部 SVC の測定結果（単位：μsec）

プロセッサコンテキストの切替え	24.5
ゲートコール	3.3
割込み(上位部への通知)	13.2
割込み(ハンドラ)	5.4

6．考察

測定結果から，ゲートコールによるマイクロカーネルの呼出しはトラップよりも 8 倍程度高速であり有効である．また，割込みについては，ハンドラは 5μs であり，実用的であることがわかる．また，上位部で仮想化する場合でも 13 + 24 = 37 μs であり，上位部で割込みの仮想化オーバヘッドを小さくすることでドライバやダイナミックリンカを容易に実現することが可能であると考える．

7．おわりに

本報告では，2次元アドレスを管理するマイクロカーネルの実現と評価について述べた．今後はマイクロカーネル上位部の実現，システムの実現を行っていく．

参考文献

[1] Hayakawa, et.al: “Basic Design of SHOSHI Operating system that Supports Handwriting Interfaces”，情報処理学会論文誌，Vol.35，No.12

[2] 森永，他: “OS/omicron V4 のためのマイクロカーネルの設計”，情報処理学会コンピュータシステムシンポジウム論文集，pp.15-22

[3] 森永，他: “2次元アドレスを管理するマイクロカーネルのプロトタイプと評価”，情報処理学会第 48 回全国大会