次世代アーキテクチャ向け オペレーティングシステム・マイクロカーネルの開発

6B－6

福本 淳　吉田 英樹　申 承昊　岡本 利夫

（株）東芝　研究開発センター

1 Cubix

我々は次世代マイクロプロセッサアーキテクチャ向けオペレーティングシステム Cubix（CUBe of 2 byte unIX）[1, 2, 3, 4] の開発をおこなっている。Cubix は以下のような特徴をもつ。

1. Cubix は単一仮想記憶空間を用いる。Unix [1] などではプロセスごとに仮想記憶空間が独立してる多重仮想記憶空間であったが、Cubix ではすべてのプログラムやデータが共通の単一仮想記憶空間中に存在する（図1）。アプリケーション間の通信はメモリポインタの受け渡しとサブルーチン呼び出しと等価になる。

   仮想空間は複数のメモリセクションにより構成される。各メモリセクション毎に「スレッドのID」と「現在スレッドが実行しているプログラムの存在するメモリセクションのID」をキーとして readable/writable/executable の各属性を指定することにより、メモリセクションに対するアクセス保護を設定する。ここでスレッドIDがキーとして加わっているため、共通の単一仮想記憶空間であってもスレッドごとに参照可能メモリセクションが異なり、スレッド間のアクセス保護が可能となる。

2. Cubix は一元化仮想記憶を用いる。ディスク上のデータとメモリ中のデータの区別なくアクセスできるように、すべて仮想記憶空間中に存在する。

   ディスクに格納されているファイルは、Cubix ではメモリセクションに対するバッキングストアとなる。物理メモリは実体に対するキャッシュとして扱う。この点で Cubix のメモリセクションは Mach におけるメモリオブジェクトと類似の概念である。

3. マイクロカーネルアーキテクチャにより、さまざまな構成で柔軟に用いることができるようにする。

   全体構成概念図を図2に示す。Unix サーバなどの OS サーバにより他の OS との互換性を提供する。

Development of an Operating System Microkernel for Next-Generation Architecture

FUKUMOTO Atsushi, YOSHIDA Hideki, SHIN Sung Ho, OKAMOTO Toshio

Toshiba Corporation, R&D Center

本報告ではマイクロカーネルのプロトタイプの作成について述べる。

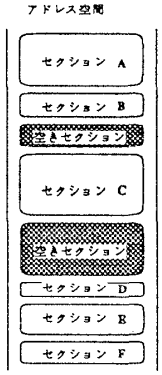

図 1: 仮想空間構成概念図

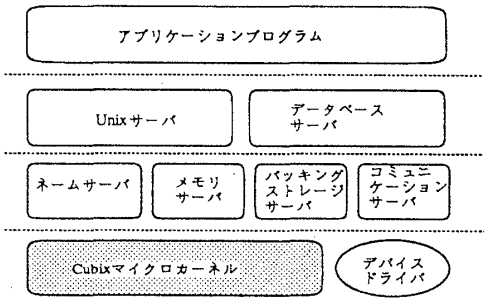

図 2: Cubix 構成概念図

2 プロトタイプの位置付け

Cubix ではメモリセクション単位のアクセス保護実現のためのメモリ保護機構をプロセッサが備えていることを想定しているが、このためには実際の使用環境でどのような問題が

[1] Unix は Bell 研究所が開発し、USL がライセンスするオペレーティングシステムです。

生じるかを評価することが必要であり、この評価をおこなうために、本マイクロカーネル・プロトタイプを開発している。

Cubixは64ビット程度以上の広いアドレス空間を持つプロセッサを前提としているが、今回開発しているマイクロカーネル・プロトタイプはインテル80386マイクロプロセッサをターゲットとしている。i386選定の理由は以下のとおり。

- i386ではセグメントとオフセットをあわせることにより45ビットの仮想アドレス空間を用いることができる。
- Cubixのメモリセクションをi386のセグメントに対応させることで、Cubixのメモリ保護機能の一部をセグメント単位のメモリ保護機能を用いて実現できる。

3 マイクロカーネルの構成

Cubixマイクロカーネルは、スレッド管理部・仮想空間管理部・例外ハンドラ・デバイスドライバから構成される。スケジューラやページャ、ファイルシステムなどは上位のOSサーバが提供する。

マイクロカーネルへのシステムコールには以下のようなものがある。

- スレッド管理（生成・破棄・状態の読みだし・状態の変更・ID読みだし・統計情報・その他）。
- メモリ管理（メモリセクション生成・破棄・属性の読みだし・大きさ変更・ID読みだし・アクセス権追加・アクセス権削除・統計情報・ページのマップ・アンマップ）。
- スケジューラインタフェース（スケジューラ設定・スレッド実行停止・スレッド実行再開・スレッドディスパッチ・スケジューライベント取得）。
- 例外処理インタフェース（例外ハンドラ登録・例外ハンドラ情報読みだし・被割込スレッド情報読みだし）。

4 i386での実装

i386での実装に際して主に問題なのは仮想アドレス空間の取り扱いである。メモリセクションをi386のセグメントに一対一対応させて割り当てた。ローカルデスクリプタテーブルすべてをメモリセクションに用い、グローバルデスクリプタテーブルはシステムコールゲート等に用いている。したがってメモリセクションはシステム中に8K個まで存在できる。またi386のリニアアドレス空間はメモリセクション生成時に動的に割り当てており、リニアアドレスは32ビットなので、現在のところメモリセクションは合計4ギガバイトまでしか割り当てられない。

前述のようにCubixのアクセス保護は「対象メモリセクションID」「スレッドID」「現在スレッドが実行しているセクションID」の3つをキーとしている。したがって、スレッドの切替をおこなう時と、実行しているメモリセクションの変更（セクション間ジャンプ）をおこなう時に、セグメントの保護属性を更新する必要がある。本来CubixではプロセッサのMMUによりメモリアクセス保護をおこない、セクション間ジャンプをおこなう際にはMMUへ新しいプログラムのメモリセクションIDの情報が自動的に伝わることにより保護属性の更新をおこなうことを想定している[4, 3]。i386上のプロトタイプでは上記のMMUの代りにセクション間ジャンプのシステムコールを用意し、マイクロカーネル内でソフトウェアにより保護情報を更新している。この際なるべく高速に更新できるようにマイクロカーネル内では「スレッドID」と「実行中セクションID」の2つをキーとしてアクセスが許可されるメモリセクションを検索するためのリストを保持している。Unix流のプログラムを実行する場合に（最低限）必要となるのはText・Data・Stackの3つのメモリセクションなので、保護情報切替の際は、切替え前のスレッドが使っていた（すくなくとも）3つのセグメントを無効状態にし、新しいスレッドのためのセグメント3つを許可状態にすることになる。

5 おわりに

現在までにマイクロカーネルプロトタイプとその上で動くBSD Unixサーバを作成した。BSDサーバはほぼオリジナルのままのモノリシックな構成のまま移植されており、本来ならばマイクロカーネルの機能を使うべきところがUnixサーバに組み込まれている部分がある。今後はOS機能を複数のサーバに分割することにより図2に示したような構成にし、評価を進める。

参考文献

[1] 申 承昊 他、「大規模アドレス空間を利用するOSの構想」、情報処理学会第44回大会、1G-5、1992/3

[2] Okamoto et al., "A Micro Kernel Architecture for Next Generation Processors", pp. 83–94, Microkernels and Other Kernel Architectures Symposium, USENIX, 1992/4

[3] 瀬川 英生 他、「単一仮想記憶域を提供する64ビット・アドレス指向OS」、情報処理学会 研究報告 92-OS-55、1992/6

[4] 瀬川 英生 他、「単一仮想記憶域を提供する64ビット・アドレス指向OS」、情報処理学会 第4回コンピュータシンポジウム、情処シンポジウム論文集 Vol. 92、No. 7、1992/10