6P-2

分散ＯＳ　ＣＯＳＭＯＳ２のカーネル基盤

計 宇生　李 在己　相田 仁　斉藤忠夫

東京大学　工学部

１．はじめに

現在、筆者らの研究室では、ＣＯＳＭＯＳ２と呼ばれる分散型オペレーティングシステムの構築を行っている。ＣＯＳＭＯＳ２では、ＯＳの各種の機能をモジュールに分割し、それらを高速なＬＡＮで結ばれたネットワーク上の各ノードに配置することによって、機能分散された不均質なネットワーク資源の共有と、負荷の分散をネットワーク透明で、均質に実現する。ユーザレベルにおいてはＵＮＩＸと互換性を持つ。

本稿では、ＣＯＳＭＯＳ２のカーネル基盤について述べる。それは各種の機能を外部に提供するＯＳ機能モジュール[1]をカーネル内部でサポートする部分に当たる。

２．ＣＯＳＭＯＳ２システムの概要

図１にＣＯＳＭＯＳ２システムのノードの一構成を示す。分割された機能には、プロセス管理機能、ファイル管理機能、ネーミング機能、Ｉ／Ｏ機能等がある。システム内で唯一に存在できるモジュールをマネージャと呼んで、プロセスマネージャ、ファイルマネージャ（ネーミングサーバ）がある。それ以外のサービスはサーバで提供され、プロセスサーバ、ファイルサーバ、プリントサーバ等がある。物理デバイスの制御はデバイスハンドラで行なう。

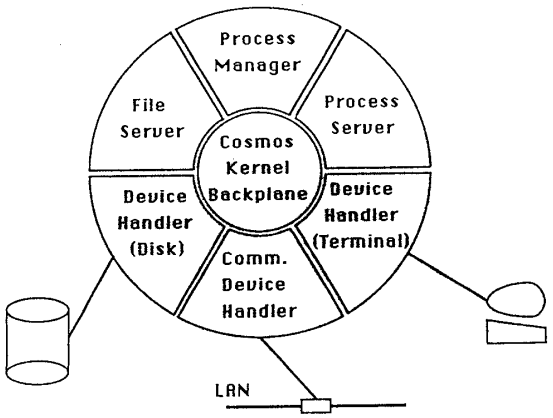

図１　ＣＯＳＭＯＳ２のノード構成

さらに、ＣＯＳＭＯＳ２ではメッセージオリエンテドな通信をモジュール間でサポートしている。それによって、自ノード内とノード間の区別なく、モジュール間で自由な情報交換ができるようになる[2]。

モジュール間のスケジューリング機能、モジュール間におけるメッセージ通信機能、そしてノードの初期化等をする部分を合わせてＣＯＳＭＯＳカーネルバックプレーン（ＣＫＢ）を構成する。ＣＯＳＭＯＳ２の各ノードにはＣＫＢと通信デバイスハンドラが一つずつ置かれる。それら以外のモジュールはすべて、各ノードに自由に組み合わせて配置することができ、システム構成に大きな柔軟性をもたらしている。

３　モジュールのサポート

ＣＯＳＭＯＳ２では、ノード内のモジュールをサポートするための機能をできるだけ簡略化し、モジュール間の切り替えの負荷が軽く済むように配慮している。そうすることによって、モジュールの拡張性を生かしながら、モジュール分割によるオーバヘッドを少なくしている。

モジュールの実装は次のような方針を取っている。

・デバイスハンドラ以外のモジュールは、すべて自らのコンテキストを持ち、モジュール・コンテキスト・スイッチ（msleep／newrun）によってモジュール切り替えが行われる。

・デバイスハンドラはコンテキストを持たず、他のモジュールから直接呼び出される。デバイスからの動作完了の知らせは、割り込み処理の中でメッセージとして作成し、メッセージインタフェースを通して、呼ばれたモジュールに戻る。

・モジュール間のスケジューリングは、固定優先度で行なう。モジュールの優先度は高い方から、（デバイスハンドラ、）マネージャ、プロセスサーバ以外のサーバ、プロセスサーバの順となっている。これを低レベルスケジューリングと称して、後述のモジュール内のスケジューリングと区別している。

Kernel Foundation for Distributed Operating System COSMOS2

Yusheng JI, Jaekee LEE, Hitoshi AIDA, Tadao SAITO

University of Tokyo

4　モジュール内におけるプロセス制御

ＣＯＳＭＯＳ２では、各モジュールの内部におけるプロセス制御のために、以下のようないくつかのプリミティブを提供する。それらを異なる種類のモジュールの間で使い分けることによって、性能の向上と、各モジュールが共同で行われる各種の処理を円滑にサポートすることができる。

4．1　プロセスサーバ

ＣＯＳＭＯＳ２ではすべてのユーザプロセスがプロセスサーバ上で実行される。したがって、プロセスサーバ上では、通常のＵＮＩＸと同じような動的なプロセススケジューリング機構が要求される。さらに、ＣＯＳＭＯＳ２のプロセスサーバ上では、他のモジュールからのメッセージを受け取り、メッセージの内容に対応する処理を行なう機能が必要である。

ＣＯＳＭＯＳ２では、上述のプロセス実行のスケジューリングを行なうスケジューラと、受信メッセージの監視を行なうメッセージ監視・処理プロセスを合わせてプロセスサーバ上のカーネルプロセスとする。

すなわち、プロセスサーバ上では、プロセスの自らの実行権放棄か、タイマ割り込みによって実行を中断された時、カーネルプロセスはまずメッセージキューに届いた自モジュール宛てのメッセージがあるかどうかを調べ、あれば、まずメッセージに対する処理を行ない、その後、他のプロセスの中で高い優先度のプロセスを捜して実行に移る。

4．2　他のサーバ

プロセスサーバ以外のサーバモジュールでは、分割されたＯＳ機能の一部を実行するので、いわば、ほとんど同一種類のサービスを実行している。したがって、ユーザプロセスに対するようなスケジューリングをする必要がなく、同じ優先度で実行させてよい。

一方、プロセスサーバ以外のサーバ類は通常、複数のモジュールからサービスを要求される。この時、十分なスループットを保証するため、それらと同時に通信できることが必要である。また、サーバからデバイスハンドラを呼び出す時など、処理の一時中断と再開機能も要求される。

以上のことから、プロセス以外のサーバでは処理の中断と再開が可能なライトウェイトプロセスを採用する。このとき、サーバ上のメッセージキューの監視をするプロセス（これを窓口プロセスと呼ぶ）とサーバ上の各受信したメッセージに対する処理をするプロセス（これをサーバプロセスと呼ぶ）は、Ｍａｃｈ[3]システムにおけるtaskとthreadの関係に似ている。すなわち、それらは共通のメモリ、資源を持ち、各サーバプロセスにはそれぞれスタックポインタとレジスタイメージを持っている。それによって処理の一時中断と再開ができるようになり、かつそのためのオーバヘッドが少ない。ただし、サーバプロセスが実行を停止するのは処理が終了（exit）した時か、他のモジュールへの通信が発生（sleep）した時のみである。

4．3　マネージャ

他のコンテキストを持つモジュールとして、マネージャ類がある。ここでは、モジュールＭ１、Ｍ２、Ｍ３の間で発生するトランザクションで、Ｍ１－＞Ｍ２－＞Ｍ３－＞Ｍ２－＞Ｍ１のようなルートを取る場合、これをモジュールＭ２を介してネストしたという。

ネストが通常起こらないマネージャでは、処理の一時中断／再開は必要でない。この時、基本的に、単一プロセスで処理することが可能である。モジュールによって、処理の一時中断が必要になった時でも、再開する場所が決まっているので、単なるテーブルによる中断前の状態の保持でまかなうことができる。

5．おわりに

以上で、機能別に分割されたモジュールをＬＡＮ上に物理的に分散配置することによって構成される分散オペレーティングシステムＣＯＳＭＯＳ２のカーネル基盤について述べた。ＣＯＳＭＯＳ２では、上述のような、モジュール間とモジュール内の２段スケジューリング、及びモジュールの処理形態に見合った３種類のプロセス制御機構を提供することによって、より効率的に、上位のモジュールをサポートしている。

＜参考文献＞

［１］計、李、鈴木、相田、斉藤：「分散ＯＳ　ＣＯＳＭＯＳ２の実装（１）－ＯＳ機能モジュール」　情報処理学会第36回全国大会（4D-1）１９８８年３月

［２］鈴木、李、計、相田、斉藤：「分散ＯＳ　ＣＯＳＭＯＳ２の実装（２）－通信サブシステム」　情報処理学会第36回全国大会（4D-2）１９８８年３月

［３］M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian and M. Younge, "Mach: A New Kernel Foundation for UNIX Development", Proceeding of USENIX 1986 Summer Conference, pp. 93-112(1986).