3P-5

リアルタイム環境を持つマルチプロセッサUNIXシステム

臼井澄夫，斉藤彰男，高畑泰志（三菱電機株式会社）
高橋良岳（三菱電機東部コンピュータ株式会社）

1．はじめに

スーパーミニコンMelcom 70MXシリーズのOSであるOS／60UMXは，リアルタイム機能とUNIX機能を同時に提供するリアルタイムUNIXである。今回，双頭非対称型（マスタ・スレーブ型）マルチプロセッサを含むMX／5000シリーズのリリースに伴い，マルチプロセッサ機能の強化を行ったのでその概要を報告する。

2．設計方針

設計方針は以下のとおりである。

（1） リアルタイム処理とUNIX環境が並存する特色を生かし，スループットの向上と同時にリアルタイム・レスポンスの確保をはかる。

（2） 一般ユーザがプログラム及び操作の変更なしにマルチ・プロセッサ・システムに移行できるよう，両プロセッサ間の負荷の自動分散を行う。

（3） UNIXシステムではカーネルのプロセッサ負荷は全体の30～50%程度に達することがある。このため，ユーザ・プログラムだけでなくカーネル部分を両プロセッサで効率よく実行させるようにしてスループットの向上をはかる。

（4） プロセッサ負荷の自動分散の他に，ユーザ・プログラムでの明示的指定によりプログラムの実行プロセッサを選択できる，ユーザ・スケジュール機能も実現する。

3．実現方式

（1） OSの構造

OS60／UMXの中核は，全体の資源管理，物理入出力，割込み等のハードウェア依存部を制御するリアルタイム・スーパバイザ，及びこのスーパバイザの下で動作するUNIXカーネルより構成されている。

リアルタイム・タスクはスーパバイザ・コールを利用して種々のリアルタイム機能を実現することができる。

UNIXプロセスはUNIXカーネルのシステム・コールのサービスを受けることができるほか，リアルタイム・タスク同様にスーパバイザ・コールも利用することができる。

OS60／UMXは，このような構成をとることでリアルタイムOS機能とUNIXを完全に並存させ，かつUNIXプロセスからのリアルタイムOS機能の利用を可能としている。

（2） プロセッサ・スケジューリング

マルチプロセッサ・システムにおけるプロセッサ・スケジュールは基本的に次の方式で行った。

i）リアルタイム・スーパバイザは主としてマスタ・プロセッサで実行させ，主要な資源管理及び入出力等を行う。

ii）ユーザ・プログラム及びUNIXカーネルは両プロセッサに分散して実行させる。このとき，UNIXカーネル及びUNIXプロセスはCPUを連続して使うことが比較的多いので，スレーブ・プロセッサに優先して割付け，CPU稼動率及びリアルタイム・タスクのレスポンスの向上をはかった。

Multi-processor UNIX system with real-time environment

Sumio USUI[1], Akio SAITO[1], Yasushi TAKAHATA[1], Yoshitake TAKAHASHI[2]
[1]Mitsubishi Electric Co. [2]Mitsubishi Electric Computer Systems (Tokyo) Co.

ⅲ）リアルタイム・タスクに対する実行プライオリティ・スケジューリング，及びＴＳＳタスクに対するラウンド・ロビン・スケジュールは両プロセッサに対して適用する。だだし，ラウンド・ロビンにおけるタイム・クォンタムは，プロセッサ毎に調整を可能とした。

（３）ＵＮＩＸカーネルの実行制御

ＵＮＩＸカーネルはもともと単一プロセッサでの実行を前提として作られているため，そのままマルチプロセッサ化すると，各プロセッサで実行されるカーネル・コンテクスト又は割込みレベルの間でクリティカル・セクションに対する競合が発生して正しく動作しない。これを一般的に回避するためにはカーネル内のクリティカル・セクションをすべて洗い出したうえセマフォ等による煩雑な排他処理を必要とするが，今回は次の方式で競合の回避を実現した。

ⅰ）カーネル・コンテクスト間の競合

カーネル・コンテクストの実行を逐次化した。すなわち，カーネルは同時にはどちらか一方のプロセッサでのみ走るようにした。この制御はリアルタイム・スーパバイザのスケジューラで行うため，カーネル自身はクリティカル・セクションを意識する必要がなく，セマフォ処理によるコード変更が少ないため信頼性が高い。又セマフォ処理の誤りによるデッドロックの可能性もない。

ⅱ）割り込みレベルとカーネル・コンテクストの競合

割込みレベルではクリティカル・セクションとなる処理を行わず，別にデーモン・プロセスを作っておき，割込みレベルからは，このデーモン・プロセスに処理要求をメッセージとして渡すのみとした。デーモンは各処理要求をカーネル・コンテクストで実行する。

この結果，カーネルがクリティカル・セクションを実行している時にも割込みを制限する必要はなく，また割込み処理によってスレーブ・プロセッサでのカーネルの実行は阻害されることがない。

（４）ユーザ・スケジュール機能

ユーザ・プログラムから実行プロセッサを明示的に指定できる機能として，表１のようなスーパバイザ・コールを用意した。この他に，プログラムのリンク・エディット時に実行プロセッサを指定することも可能とした。

（表１）ユーザ・スケジュール用スーパバイザ・コール

クラス	名　　称	実行プロセッサ	有効期間
1	M: MASTER	マスタ・プロセッサのみ	次にクラス１のスーパバイザ・コールが出るまで
	M: SLAVE1	スレーブ・プロセッサのみ	
	M: SLAVE2	マスタ，スレーブ両プロセッサ	
2	M: DOMAST	マスタ・プロセッサのみ	当該ＣＰＵクォンタム
	M: SLAVE	スレーブ・プロセッサのみ	

これらの機能を用いることで，アプリケーション・システムごとのＣＰＵ負荷や実行順序の細かなチューニングが可能となる。

４．まとめ

リアルタイム機能とＵＮＩＸを同時にサポートするＯＳのマルチ・プロセッサ化の方式について報告した。本方式では，リアルタイムＵＮＩＸの構成を生かし，リアルタイム性を重視したマルチプロセッサ制御を実現した。