4P-5

種々のハードウェアに適用可能な密結合マルチプロセッサ用OS MUSTARD

広屋 修一 *、渡辺 明子 **、川口 浩美 **、宮地 利雄 *
* 日本電気（株）、** 日本電気技術情報システム開発（株）

1. はじめに

シングルプロセッサ構成の組込み型システムを密結合型マルチプロセッサ(TCMP : Tightly Coupled MultiProcessor)用に拡張するためには、スヌープ・キャッシュなどによるメモリ・インタフェース部の改造が必須となるが、これはプロセッサ・ボード内だけの小規模なハードウェア改造であり、ソフトウェアに対する変更も少ない。これに対して、バス構造、I/Oモデル、割込みモデルなどの変更は、ソフトウェアを含めたシステム全体への影響が大きく、組込み型システムにおいてスループットを短期間で向上させるためには、これらの変更なしにTCMP型に移行できることが重要となる。本稿では、理想的なハードウェアを持つTCMP型システムと、従来のシングルプロセッサ用ハードウェアをベースにしたTCMP型システムの両方をサポートするようなOS構造について述べる。

2. MUSTARDのねらい

従来、組込み型システムにおいては、リアルタイム性能が強く望まれていたが、近年、これに加えてシステムのスループット向上とフォールト・トレラント機能が望まれている。そこで、我々は交換機システムへの組込みなどで実績を持つ V60/V70 [1] 用オペレーティング・システム RX616 [2] をベースに、密結合型マルチプロセッサ上で動作可能な **MUSTARD** [3][4] を開発した。

MUSTARDでは、どのプロセッサでもカーネル／タスクが走行可能であり、タスクは優先度に従いダイナミックに負荷分散される。カーネルは、割込みハンドラ起動部とリスタート用システム・コール処理部以外は、ほぼ全体がクリティカル・セクションで構成されるが、 RX616 のような組込み型OSではカーネル走行比率が低く、 4〜8台程度のプロセッサ構成であれば、プロセッサ数に比例した速度向上が期待できる。

ミニスーパーコンピュータやワークステーションなどで TCMP型システムを構築する場合には、特化したハードウェア構成に対して、そのハードウェアの特徴をぎりぎりまで活かすOS構造が望まれるが、組み込み型システムにおいては、既存のハードウェア・モデルを含む種々のハードウェア・モデルに適用可能なポータブルなOS構造が望まれる。

3. 種々のハードウェアに適用するため機構

3.1 プロセッサ間通信

MUSTARDでは、プロセッサ間通信をプロセッサ間割込みか、各プロセッサごとのクロック割込み（分散クロック）を用いて実現する。 **MUSTARD**では、プロセッサ間通信要求が発生した場合に、共有メモリ上にある対象プロセッサ用通信エリアにその要求を書き込み、プロセッサ間割込みが実装されている場合には、対象プロセッサに割込みをかける。対象プロセッサでは、プロセッサ間割り込みかクロック割込みが発生した時に、この要求をセンスする(図1)。このようにすれば、クロック割込みをクロック周期の遅延をともなったプロセッサ間割込みとして統一的にとらえることができる。 **MUSTARD** では、さらにシステム・コールの出入口でもプロセッサ間通信要求のセンスを行うので、一般的にはシステム・コールの発行レートでプロセッサ間通信が完了する。

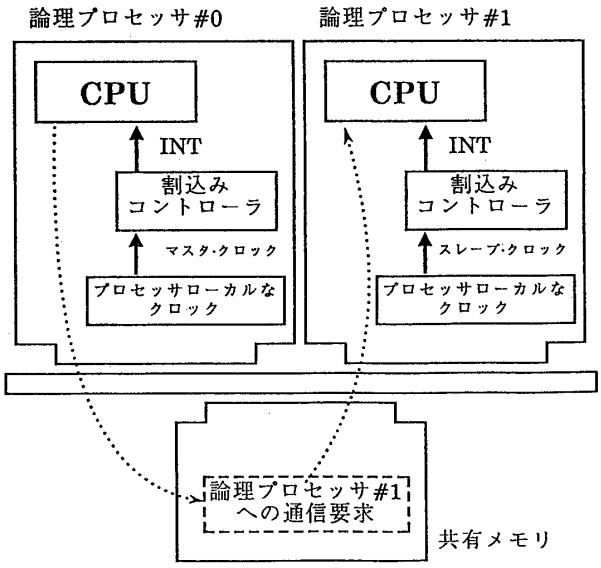

図1. 分散クロックを用いたプロセッサ間通信

Mechanism for Portability of the Multiprocessor Operating System MUSTARD
Shuuichi HIROYA*, Akiko WATANABE**, Hiromi KAWAGUCHI**, Toshio MIYACHI*
*NEC Corporation, **NEC Scientific Information System Development Ltd.

3.2 分散クロック制御

分散クロックを用いた場合、従来クロック処理の中で行っていたタイム・スライス処理、システム時計管理、周期起動などの処理を行うプロセッサを決定する必要がある。 **MUSTARD** ではクロック・マスタ／スレーブという概念を持ち込み、クロック・マスタにアサインされたプロセッサのみが従来のクロック処理を行い、クロック・マスタ及びクロック・スレーブの両方でプロセッサ間通信のセンスを行う。

3.3 遠隔I/O制御

MUSTARDでは、どのプロセッサからもアクセス可能なシステム・グローバルなI/Oと特定プロセッサからのみアクセス可能なプロセッサ・ローカルなI/O の2つのI/Oモデルをサポートする。プロセッサ・ローカルな I/O装置を制御する方法として、特定プロセッサでのみ走行可能なタスクを実現する方法が考えられるが、この方法ではタスク・ディスパッチ時間の増加が避けられない。そこで **MUSTARD** では、プロセッサ固有のI/Oハンドラをプロセッサ間通信を用いて起動するリモート・ハンドラというメカニズムを導入した。 def_rhd()システム・コールはI/Oハンドラを特定のプロセッサにはりつけることを行い、 rhd_cll()システム・コールはこのI/Oハンドラを起動することを行う(図2)。

3.4 プロセッサID

プロセッサIDには、物理プロセッサIDと論理プロセッサIDの2種類が存在する。物理プロセッサIDは、プロセッサ・ボードに割り振られた任意の番号であり、論理プロセッサIDは、 OS が利用する 0 から始まる連続した番号である。論理プロセッサIDを得る場合に、毎回物理プロセッサIDをキーにテーブル参照を行うのは非効率であり、プロセッサ・ローカルな論理プロセッサIDキャッシュを参照することが望まれる。 **MUSTARD** では論理プロセッサIDキャッシュ有／無の2つのモデルをサポートしている。

表1. ユーザ・カスタマイズ・ルーチン

ルーチン名	内容
ProcUserPhysId	物理プロセッサ番号の取得
ProcUserId	論理プロセッサ番号の取得
ProcUserIdSet	論理プロセッサ番号のハードへの設定
ProcUserInt	指定プロセッサへの割込
ProcUserMnt	指定プロセッサの動作凍結解除
ProcUserUmnt	指定プロセッサの動作凍結
ExcUserIPIMode	プロセッサ間割込みの有無の指定
ClkUserMode	単一クロック／分散クロックの指定

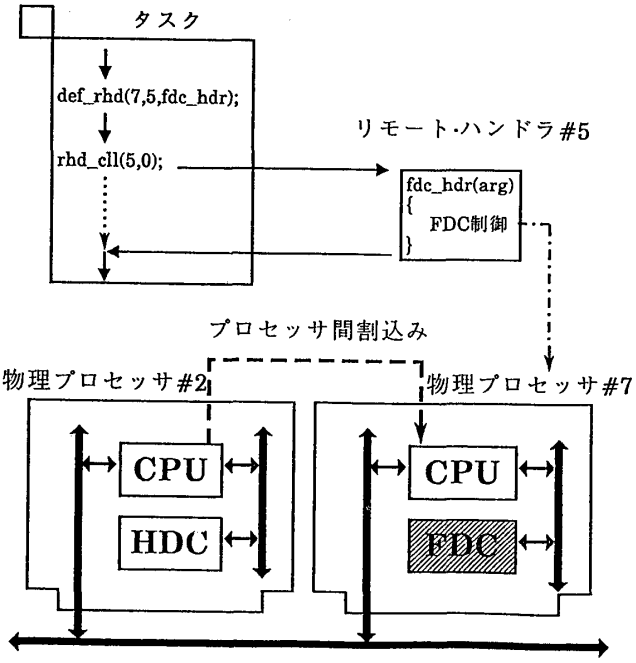

図2. リモートハンドラの動作

4. ユーザ・カスタマイズ・ルーチン

MUSTARDでは、表1のようなユーザ・カスタマイズ・ルーチンにより、ハードウェア依存部の分離を行っており、種々のTCMP型ハードウェアへの移植が容易となっている。

5. おわりに

MUSTARD は、 32ビットマイクロプロセッサ V70 とバス・スヌープ・プロトコルを持つワンチップ・キャッシュ・コントローラμPD71641 [5] を用いたシステムで開発を行っている。今後、 **MUSTARD** を種々のシステムに適用し、実際のアプリケーションによる性能評価を行う予定である。また現在、カーネル内をさらに並行動作させるための改造と、 RX-UX832 [6] をベースにした密結合マルチプロセッサ用リアルタイムUNIX の開発を行っている。

参考文献

[1]河本他, "V60/V70マイクロプロセッサと高信頼化システム", コンピュータ・システムシンポジウム, 1987.
[2]古城他, "V60リアルタイムOSの設計", 情処第33回全国大会1C-4, 1986.
[3]渡辺他, "密結合マルチプロセッサ用OS MUSTARDにおけるプロセッサ間通信方式", 情処第39回全国大会, 1989.
[4]川口他, "密結合マルチプロセッサ用OS MUSTARDの耐故障性の強化と障害処理機構", 情処第39回全国大会, 1989.
[5]"μPD71641ユーザーズ・マニュアル", 日本電気, 1988.
[6]下島他, "V60/V70リアルタイムUNIX -概要-", 情処第35回全国大会3D-3, 1987.