4N-9

OS／omicron におけるタスク間通信の一方式

森康弘 ， 並木美太郎 ， 高橋延匡
(東京農工大学)

1．はじめに

OS／omicronはマルチプロセッサによる並列処理を目標としている。並列処理において、タスク間の通信・同期は大きなオーバーヘッドとなる。特にOS／omicronは、ユーザが自由にタスクスケジューリングを行うことができる反面、プログラマの負担は大きくなっているので一層、通信・同期のオーバーヘッドの改善が望まれる。

本稿ではOS／omicronにおいて効率が良く、プログラミングしやすいタスク間通信方式について述べる。

2．OS／omicronタスクモデル

OS／omicronにおけるタスクの形態には2種類ある(図1)。一つは、互いに環境を共有せず、OSを介して通信する互いに疎な関係であるタスクである。もう一つはタスクフォースといい、手続き領域と静的データ領域を共有しながら、並列に実行されるタスクの集まりである。前者は小容量のメッセージの送信に有利であり、後者は大容量データの参照などに有利である。

現在の版は単一プロセッサで動いているが、次版では共有メモリ型のマルチプロセッサを、最終的にはマトリクススイッチ型のマルチプロセッサを実現する。タスクフォースはそれらのマルチプロセッサを考慮したモデルである。タスクフォースの内側と、外側でのプログラミングの効率の差異をなくすために、タスクフォースの内外における統一した通信のインターフェースも必要である。そのため、統一した通信のインターフェースとしては現在のOS／omicronのタスクフォース外通信として実現されている、メッセージ通信方式を用意した。

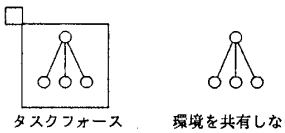

図1．OS／omicronにおけるタスクの形態

3．タスクフォース内通信方式

タスクフォース内の通信方式として、ユーザタスクが知りうる空間に対するデータの読み書きによって通信する、“メモリマップトメッセージ”(memory mapped message) 方式を示す。この方式ではシステムが管理しなければならない情報は、メッセージのポインタと受信側の識別子だけである。

システムが用意する基本命令としては、表1に示す通りである。1)__rqst__mssg__area () はセグメント化された、mssg__sizeで示されるだけのワード数のメッセージ領域をタスクフォース内の静的データ領域に確保する。2)__free__mssg__area () はmssg__pntrで示されるメッセージ領域を開放する。3)__rqst__and__rcve () は送信側タスクから、メッセージのポインタを受信し、メッセージへのアクセス権を獲得する。4)__free__and__send () は受信側タスクへ、メッセージのポインタを送信し、タスクフォース内メッセージ領域のアクセス権を放棄する。

通常のタスク間通信では、3)__rqst__and__rcve () と4)__free__and__send () で同期をとって、メッセージ領域のアクセス権を譲渡することにより、タスク間の通信を実現する。このようにした理由は、完全に同期をとることによってデッドロックをさけるためと、セグメント化してポインタで送ることによるパフォーマンスの向上を考えたからである。

並列処理に欠くことのできないパイプライン処理もこの方式で行うと、メッセージ領域の切り替えにより、互いに競合することも少なく、効率良く動くと考えられる。

“メモリマップトメッセージ”方式は、ユーザプログラムの負担を多少重くすることによって、システムにかかる負荷を軽減している。この方式はタスクフォース内に限らず、共有メモリ型のアーキテクチャ全般に適用できるものである。

送信タスク	受信タスク

pntr1 = _rqst_mssg_area(mssg_size);
/* pntr1からpntr1 + mssg_size - 1で示される領域にメッセージ(データ)を書き込む */
_free_and_send(pntr1,receiver); →
	pntr2 = _rqst_and_rcve();
	/* pntr2からpntr2 + mssg_size - 1で示される領域のメッセージ(データ)を処理する。 */
	_free_mssg_area(pntr1);

図2．メモリマップトメッセージ型のタスク間通信

A Method for Intertask Communication provided by OS／omicron
Yasuhiro MORI , Mitarou NAMIKI and Nobumasa TAKAHASHI
Tokyo University of Agriculture and Technology

表１．タスク間通信用基本命令の仕様

	基本命令	引数	リターン値
1)	_rqst_mssg_area() /* メッセージ領域を確保する。 */	WORD mssg_size /* メッセージ領域の大きさ */	CHAR *pntr /* メッセージ領域へのポインタ */
2)	_free_mssg_area() /* メッセージ領域を解放する。 */	CHAR *pntr /* メッセージ領域へのポインタ */	WORD torf /* 成功 TRUE 失敗 FALSE */ */
3)	_rqst_and_rcve() /* メッセージ領域へのポインタを受け取る */	なし	CHAR *pntr /* メッセージ領域へのポインタ */
4)	_free_and_rcve() /* メッセージ領域を解放し、メッセージ領域へのポインタを送信する。	CHAR *pntr /* メッセージ領域へのポインタ */ WORD receiver_id /* 受信者の識別子 */	WORD torf /* 成功 TRUE 失敗 FALSE */

```
typedef struct mssg_dcpr  {
    CHAR        *mssg_pntr;
                    /* タスクフォース内メッセージ領域へのポインタ */
    struct  mssg_dcpr  *next_mssg_dcpr;
                    /* 次メッセージデスクリプタへのポインタ */
    }  MSSG_DCPR;    /* メッセージデスクリプタ */
```

図３．メッセージ記述子

４．タスクフォース外通信との比較・考察

タスクフォースの内と外では、タスク間通信の性格が異なってくる。タスクフォース外はOSがタスク間の通信を管理し、タスクフォース内ではユーザに通信を任せることも可能である。保守性の問題を考えると、タスクフォース外の通信が、ユーザタスクが他のユーザタスクのメモリ空間を覗けないため、別のタスクを破壊することは少ない。しかし、メモリを共有する形であるタスクフォース内の通信は、ユーザタスクが静的データ領域を自由にアクセスできるので、排他制御を正確に行わないと、別のタスクが誤動作をしたり、破壊されたりするということが考えられる。“メモリマップトメッセージ”方式はこれを、“アクセス権を同時に２つのタスクが持つことはない”という仕様によって吸収している。しかし、前節で述べた“メモリマップトメッセージ”方式はあくまで基本命令であって、危険なことが起こる可能性はある。例えば、送信側のタスクがポインタを保存しておくことでタスクフォース内のメッセージ領域をアクセスすることもできる。また、タスクフォース外の通信方式との統一性も考える必要がある。

タスクフォースの外の通信方式は、OS／omicronでは従来のメッセージ通信方式である。タスクフォース内の通信方式を抽象化し、メッセージ通信方式と同様のインターフェースを提供することによってタスクフォース内外の通信方式を統一した（図3）。タスク間で通信するとき、通信相手が自タスクと同一のタスクフォースに存在するかどうかを調べ、存在するならタスクフォース内通信としてタスクフォース内メッセージ領域にメッセージをコピーし、アクセス権を譲渡することで相手にメッセージを送る。相手がタスクフォース外であるなら、OSの管理下でメッセージ通信としてメッセージを相手に送るのである。タスクフォース外の通信とタスクフォース外の通信を処理の違いを表２に示す。

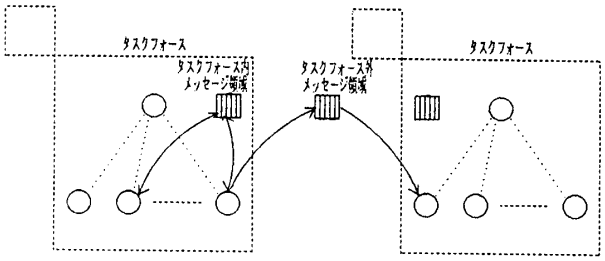

図４．タスクフォース内外の統一した通信イメージ
◯ ：タスク

表２．タスクフォース内通信とタスクフォース外通信の差

通信方法	仕様
タスクフォース内通信	・タスクフォース内でメッセージ領域を共有し、そのアクセス権を譲渡しあうことによって、通信を行う。
タスクフォース外通信	・ＯＳがメッセージ領域を持っていてタスクフォース外タスク同志の通信の仲介を行う。

５．パイプライン処理

“メモリマップトメッセージ”方式は共有メモリ上におけるパイプライン処理に適している。パイプライン処理はマトリクススイッチ型のマルチプロセッサが最も得意とする処理である。メモリを切り替えて動作することから、“メモリマップトメッセージ”方式を使って共有メモリ型のマルチプロセッサ上でシミュレートすることも可能である。

６．終わりに

OS／omicronにおけるタスク間通信の一方式を示した。今後、OS／omicronがマルチプロセッサ化された時に役立てて頂きたい。

７．参考文献

武部桂史 “並列処理に関する研究”修士論文，東京農工大学（1982）