3T-9

# ネットワーク仮想記憶システム：NET－VMS〔2〕プロセス間通信方式

陣崎　明　　八星　禮剛　　樋口　昌宏

（株）富士通研究所

## 1. はじめに

本稿ではNET-VMS ベースのプロセス間通信方式として，各プロセスの共有データに対するアクセス方法をあらかじめ宣言しておくことにより，実際のアクセス時にプロセス本体が競合，排他制御を全く考慮することなくプロセス間通信を実現する方式を示す。

## 2. プロセス間通信方式の問題点

一般にプロセス間通信では，一つのプロセスが行う共有メモリ上のプロセス間通信領域へのアクセス方法はライトアクセス，リードアクセス，排他アクセス，同期アクセスなどがあり，これらが行われる順序はプロセス実行時に動的に変化する。このためプロセス間の競合制御，排他制御の機能をセマフォやメッセージ通信といった形でプロセスの中にとりこむ必要があり，ユーザプログラムの構造を複雑なものとしている。この結果，プログラム開発において処理アルゴリズムの素直な記述が阻害され，プログラムのデバグが困難になるという問題点が生じている。

ところで，プログラムの個々の手続き呼出し（サブルーチンコール）をそれぞれ一つのプロセスとみなす程度に，プロセスを小さな単位で扱えるならばプロセス間通信の形態も単純になり，プログラムの記述やデバグを容易にすることが期待できる。もちろん，こうしたプロセス間通信を実現するためにはプロセスの生成，移動を高速に行えるプロセッサ結合方式が前提となる。

## 3. 宣言的プロセス間通信

プロセスを手続き呼出しにまで分解して考えると，共有メモリ上のプロセス間通信領域に対するプロセスのアクセス方法は，各プロセスと領域毎に定まり，動的に変化しないものとなる。図1に示すマージソートプログラムの例でいえばこのプログラムを実行するプロセス（$P_0$とする）と(1)，(2)で再帰的に呼ばれるプロセス$P_1$と$P_2$の間には図2のようなデータの依存関係があり，$O_1$に対するアクセスの方法は　$P_0$はリードアクセス，$P_1$は排他的ライトアクセス，と定まる。

```
procedure  msort(var I,O : array) ;
  var  I1,O1,I2,O2 : array ;
  begin
     if small(I) then          /*さらに分割するかの判定
        sort(I,O)              /*短配列のソーティング
     else  begin
        divide(I,I1,I2) ;      /*I,をI1,I2 に分割
        alloc(O1,size(I1)) ;   /*配列O1のための領域を確保
        alloc(O2,size(I2)) ;   /*配列O2のための領域を確保
(1)     msort(I1,O1) ;         /*I1のソーティング
(2)     msort(I2,O2) ;         /*I2のソーティング
        merge(O1,O2,O) ;       /*O1,O2 をO にマージ
        free(O1) ; free(O2)    /*O1,O2 の領域を解放
        end
  end
```

図1　マージソートプログラム

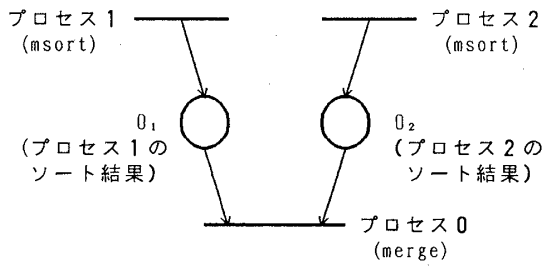


図2　マージソートにおけるデータの依存関係

このようなプロセス間通信領域へのプロセスのアクセス方法が，あらかじめプロセス毎に陽に宣言されていれば，プロセス間のアクセス競合を宣言に基づいて解決する調停者(supervisor)を置くことにより，実行時にはプロセス本体がアクセス競合や排他制御を考慮しないプロセス間通信（宣言的プロセス間通信）を実現することができる。宣言的プロセス間通信とは次のような

---

Networked Virtual Memory System:NET-VMS (2)Inter Process Communication

Akira Jinzaki, Reigo Yatsuboshi, Masahiro Higuchi

Fujitsu Laboratories LTD.

ものである。
①プロセスの実行に先立って，そのプロセスの宣言を調停者に通知する。
②プロセス実行時におけるアクセス制御は各プロセスのアクセス方法に対する宣言内容に応じて調停者が行う。
③プロセス実行の終了時にはアクセスの必要性が消滅するので，そのことを調停者に通知する。

マージソートにおける$O_1$に対するアクセスでいえば$P_1$がソーティングを終了するまでは$P_1$が排他的ライトアクセスを宣言し，$P_0$がリードアクセスを宣言している。そして，両プロセスはアクセス制御を意識せず$O_1$にアクセスしようとする。このとき調停者は次のようなアクセス制御をする。
①$P_1$のアクセス以前は，論理的に存在しない$O_1$への$P_0$のアクセスを拒否する。$P_0$はアクセス要求をだした状態で停止する。
②$P_1$のアクセスが開始されると，$P_1$の排他的ライトアクセスを満たすため$P_1$のみのアクセス要求を許し$P_0$のアクセス要求を拒否する。
③$P_1$の実行終了後，$P_1$ の宣言が無効になるのでアクセス要求を出しているのは$P_0$のみになるので調停者は$P_0$のアクセスを許す。

## 4. NET-VMS 上での実現

NET-VMS 上での宣言的プロセス間通信の実現方法について述べる。NET-VMS は各ＰＥにおいてページ単位に設けられたアクセスキーを用いて，宣言的プロセス間通信における調停者の機能を実現することができる。プロセスの実行に先立つ宣言の処理はプロセスが実ＰＥに割当てられた時点でＯＳがそのＰＥ内のアクセスキーを宣言内容に応じて設定することにより行い，プロセス実行終了時の宣言の解除はやはりＯＳがアクセスキーの解除により行う。（図3）宣言，解除の処理はともにＰＥ内のローカルなアクセスキーの設定として行うのでこの操作に関して競合の心配はない。

また，各プロセスの宣言内容の記述の生成はコンパイル時に行えばよい。

NET-VMS ではプロセッサ16台を200Mbps のリングネットワークで接続した場合，プロセッサ１台当たり2000ページ/ 秒（256バイト/ ページ）を１ページ当たり平均50μs で移動できる(1)。仮に，共有メモリ上のプロセスプールを介してプロセスの移動を行うとすれば，①プロセス生成側でのプロセスプールへのプロセス記述体(2) の書込み，②プロセス実行側でのプロセスプールからの読みだし，の２回で行えるので，プロセスの平均移動時間は 100μs 程度になる。これは1MIPS のプロセッサで100 命令程度のオーバヘッドであり，サブルーチンの実行命令数が数百命令以上であればプロセス化し，並列実行可能となる。

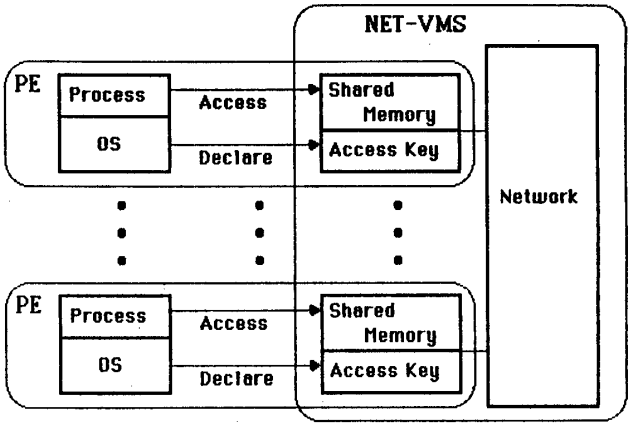


図３．NET-VMS 上での宣言的プロセス間通信

## 5. おわりに

並列分散処理ソフトウェアの開発においては，本来アルゴリズムを実現するための手段でしかないプロセス間通信が実質的な問題となる。これに対し，NET-VMS をベースにした宣言的プロセス間通信方式は，プロセス間通信をアプリケーションレベルからコンパイラ，ＯＳ，ハードウェアへオフロードする。

我々は，この方式により並列アルゴリズムを簡明に記述し，デバグを容易にするだけでなく，並列・分散処理そのものを高速化することをねらっている。

〔参考文献〕

(1)八星他：ネットワーク仮想記憶システム：NET-VMS 〔１〕システムアーキテクチャ，本大会予稿3T-8
(2)P. J. Denning:Fault Tolerant Operating System, ACM Computing Survey vol. 8 no. 4 (Dec 1976)