並列プログラミング言語SERVEの設計

2Q-1

廣谷良彰　福田晃　村上和彰　富田眞治
(九州大学大学院総合理工学研究科)

1. まえがき

我々は,多様な並列処理に柔軟に対処できる「可変構造型並列計算機」の開発を進めている[1]. 一方,ハードウェア開発と並行して,その上の並列オペレーティング・システム[2] および並列プログラミング言語SERVEの開発も行っている. 本稿では,SERVEの概要を述べた後,その処理系についても触れる.

2. 並列プログラミング言語SERVE

2.1 プロセス間通信

並列プログラミング言語のプロセス間通信は,一般に共有変数とメッセージによる方法に大別される. 共有変数による方法は,効率的な通信が可能である反面,プロセス間の相互作用をわかりにくくし,また,セマフォなどにより排他制御を行わなければならない. これは,プログラムが複雑になると予期しない誤りを犯す可能性が高くなる. 一方,メッセージによる方法は情報交換と同期操作を同時に行うことができ,見通しの良いプログラムを記述することが可能である. SERVEの設計方針は,記述性の良い並列プログラミング言語を提供することにある. 従って,SERVEのプロセス間通信は,メッセージ通信によって行うことにした.

2.2 プログラム構造

SERVEのプログラムは図1に示すように,1つ以上のモジュールから構成され,更にそのモジュールは0以上の並列プロセスから構成される.

規模の大きなプログラムの作成には,機能単位に分割してコーディング,コンパイル,テストできることが必要である. モジュールはその単位になるものである. 更に汎用的なモジュールは,他のプログラムを構築する時に再利用される. モジュール内にプロセスが存在する必要はなく,関数のみを定義しておくことも可能である. モジュールの結合は,関数やメッセージのためのポートおよびエントリを輸出入することで行う.

2.3 プロセス生成

SERVEのプロセス生成は,静的あるいは動的に行うことができる. 動的な生成により,並列度が未知の処理にも対処できプログラミングが柔軟になる. また,並列処理では,機能が同一つの多数の均質プロセスにより処理を行うことがよくある. この処理を記述するために,SERVEではインデックス付きのプロセス配列の機能を提供している.

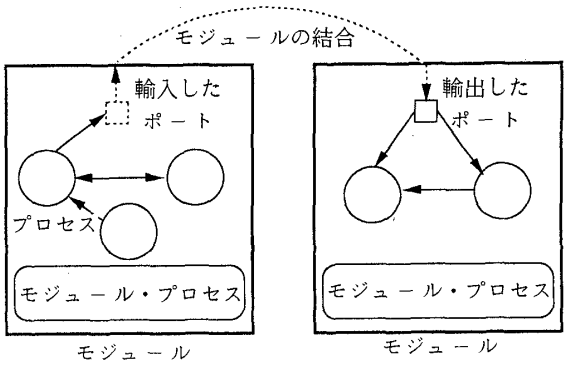

図1 SERVEプログラムの構造

2.4 メッセージ通信

(1) 通信プリミティブ

SERVEは通信プリミティブとして,非同期式と同期式のメッセージ通信を持つ. 同期式通信としては,ランデブー方式を用いている. 同期式通信は,クライアントーサーバ関係をプログラムするために必要な通信形態である. しかし,同期式通信は双方向通信であり,これのみを用いて記述されたプログラムは効率的な実行が望めない. また,不必要に同期操作が行われるため,デッドロックの可能性が高くなる. 例えば,プロセス配列の個々のプロセス間で通信を行う場合,コードは同じであるため同期式通信を用いることは好ましくない. メッセージを送信するだけで返答メッセージを必要としない時のために,非同期式通信もサポートしている. 送受信のシンタックスは以下の通りである.

(a) 非同期式

send ＜ポート名＞＜式リスト＞ [**to**＜送り先＞]
receive ＜ポート名＞＜変数リスト＞
[**from**＜送り元＞]

(b) 同期式

call ＜エントリ名＞ [＜引数リスト＞]
[**to** ＜送り先＞] **wait** [＜変数リスト＞]
accept ＜エントリ名＞ [＜変数リスト＞]
[**from**＜送り元＞] [**do**＜実行文の並び＞]
reply [＜式リスト＞]

Design of the Parallel Programming Language SERVE
Yoshiaki HIROTANI, Akira FUKUDA, Kazuaki MURAKAMI and Shinji TOMITA
Kyushu University

(2) 対多通信

同期式および非同期式のためのメッセージの受口を，それぞれポートおよびエントリと呼ぶ．エントリおよびポートの宣言は，プロセス定義の内部および外部で行える．プロセス内部で宣言された場合には，そのプロセスだけが受信可能である．これに対し，外部でグローバルに宣言された場合は各プロセスに共有され，これにより複数のプロセスが同一のエントリあるいはポートに対して受信できる．すなわち，対多通信が可能となる．従って，SERVEのメッセージ通信は上記のように，送り先および送り元の指定に関して対称形をなす．送り元および送り先の指定は，プロセス名あるいはプロセス変数による．プロセス変数は，プロセスを識別するための変数であり，プロセス名およびANY（任意のプロセスを対象とすることを意味する）などを代入できる．これにより，通信相手を状況に応じて切り替えることが可能となる．

2.5 非同期文（async／endasync）

あるプロセスの同期通信による異なるプロセスへのエントリ呼出し（**call**）は，並列に行うことが可能である．しかし，通常の同期式通信では，呼出しの度に呼出し側は返答が返されるまでブロック状態となり，各呼出しは逐次的に行われることになる．非同期文は，同期式通信と共に用い，その返答待ちを**endasync**で行う．これにより，図2に示すような複数のエントリに対する並列呼出しが可能となり，プログラムの効率を高めることができる．

2.6 初期化変数（initvar）

この変数は，図1のモジュール・プロセスにより初期化され，その他の通常のプロセスにはread操作のみが許されるモジュール内でグローバルな変数である．モジュール・プロセスは，モジュール内で最初に生成されるプロセスであり，プログラム上ではこの共有変数の初期化のみを行なう．プロセス名の定義された静的プロセスは，モジュール・プロセスの終了直前に生成される．この初期化変数は，各プロセスに対して”掲示板”の役割を果たし，これによりwrite操作が必要ない大きなデータを取り扱うことが容易になる．

2.7 プログラムの終了

並列プログラムの終了は，逐次プログラムのそれに比べ重要な概念である．SERVEプログラムが終了するのは，全てのプロセスが終了するか，または，生存している全てのプロセスが，非決定性を表現する**select**文の終了分岐で待っているときである．SERVEのプロセスは，終了に関して親子関係はなく，あるプロセスを生成したプロセスは生成したプロセスの終了を待たずに終了する．このように，Adaの複雑な終了条件を簡単化し，プログラマーに理解し易いものとした．

3. SERVEの処理系

3.1 SERVEの実行方式

各SERVEプログラムは，1つのアドレス空間を持つものとする．また，プログラム中の各プロセスは，それ自身のスタックとレジスタを持ちアドレス空間を共有するいわゆるスレッドとして動作させる．この方法は，SERVEの各プロセスに個別のアドレス空間を与えて実行する方法に比べ，コンテキスト・スイッチやSERVEプロセスの生成に伴うオーバヘッドを軽減できる．

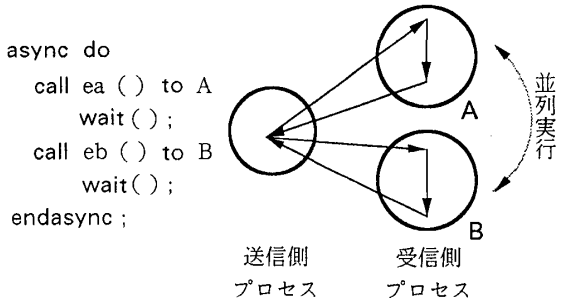

図2 非同期文による並列呼出し

3.2 疑似並列用SERVE処理系の概要

現在，「可変構造型並列計算機」およびその上の並列OSは開発中であり，また，早急にSERVEの記述性を確かめる必要から，実験的な処理系をSun-4上に試作し，疑似並列に動作させることにした．処理系は，C言語を中間言語とするトランスレータであり，既存のCコンパイラを流用する．また，モジュール単位で分割コンパイルを行う．SERVEの各プロセスは，3.1の実行方式に基づき，Sun-4のLightweight Process（以下LWP）として動作させる．Sun－4のLWPライブラリを直接用いて実現できないメッセージ通信などのSERVEの機能は，実行時ルーチンとして用意しコンパイル段階でリンクする．

我々が開発中の並列OSは，タスクとスレッドの考え方に基づいており，C言語レベルでスレッドの操作が行える．従って，可変構造型並列計算機上で実並列処理を行なう場合も，SERVEの処理系はC言語へのトランスレータとして構成できる．

4. おわりに

並列プログラミング言語SERVEと処理系の概要について述べた．現在，Sun－4上において，疑似並列用SERVE処理系の開発を進めている．

参考文献

1) K.Murakami et al.: The Kyushu University Reconfigurable Parallel Processor － Design of Memory and Inetrcommunication Architectures －, Proc. 1989 Int. Conf. on Supercomputing, pp. 351－360 (1989).

2) 福田ほか：可変構造型並列計算機の並列／分散オペレーティング・システム，情報処理学会研究報告，89－OS－43 (1989).