オペレーティング・システム 36－2
（１９８７. ９. １１）

並列オペレーティングシステムＳＳＳ－ｄにおけるプロセス管理に関する基礎考察

曽和将容
名古屋工業大学電気情報工学科

奥平誠司　林宏也
群馬大学工学部

われわれは，データフローコンピュータを例にとって，sss-d（ space sharing system for dataflow computer)と呼ばれる並列オペテーティングシステムの研究を行ってきた．すなわち，１）割込みに関する研究，２）入出力機構に関する研究，３）資源管理に関する研究を行い，sss-dの基本ファイルシステムの構築を理論的に試みた．

本論文はこれらの研究の続きで，オペレーティングシステムの最小処理単位となるプロセスに注目し，sss-dにおけるプロセス管理とプロセス間通信に関しての基礎考察を行う．

A STUDY OF PROCESS MANAGEMENT OF OPERATING SYSTEM, SSS-d, FOR PARALLEL COMPUTER

Masahiro SOWA* and Seiji OKUDAIRA**, Hiroshi HAYASHI**

*Department of Electircal Engineering and Computer Science
Nagoya Institute of Technology, Gokiso,Nagoya 466,Japan.
**Department of Computer Science,Gunma University,
Tenjincyo,Kiryu 376,Japan.

We have been researched about an operating system of dataflow computers as an example of a parallel computer's one. This involves the interrupt mechanism, the I/O system, the resourcemanagement system and the file system. This paper descrives further reseach of the operating system, especialy, the process management system.

１．まえがき

高性能並列処理の実現をめざしてデータフローコンピュータやコントロールフローコンピュータなどの研究がなされている[１，２]．コンピュータを並列処理コンピュータとして完成させるためには，その上で走るシステムプログラムも並列化されなくてはならない．本論文は，並列オペレーティングシステムに関するものである．われわれは，データフローコンピュータを例にとって，sss-d (space sharing system for dataflow computer)と呼ばれる並列オペレーティングシステムの研究を行ってきた．すなわち，１）割込みをデータフローコンピュータでどのように扱うか[３]，２）割込みに関連の深い入出力機構に関する研究[４]，３）ファイルなどの資源をどのように処理するかという資源管理に関する研究[７]を行い，sss-dの基本ファイルシステムを構築を試みた[５，６]．

本論文はこれらの研究の続きで，オペレーティングシステムの最小処理単位となるプロセスに注目し，sss-dにおけるプロセス管理とプロセス間通信に関しての考察を行う[８]．

最初に，sss-dにおけるプロセスの定義を行い，さらに，このプロセスがシステム内で取り得る状態を示し，システム内でのプロセスの識別法，プロセスの優先度，プロセス間通信について考察する．

２．プロセスとその状態遷移

sss-dでは，オペレーティングシステムが，実行していると見なしたプログラムをプロセスと定義している．

ノイマンコンピュータでは，プロセスは基本的に３つの状態（実行可，実行中，待ち合わせ）を取ることが出来た．sss-dでも基本的には同じであるが，並列処理コンピュータの場合には，プロセッサの数が多いのでそれらの状態の中身は異なる．sss-dのプロセスの状態は以下のようである．

・実行可　：新たにプロセスが生成されるか，または，待ち合わせ状態からの復帰により，実行されるのを待っているプロセスの状態．

・実行中　：そのプロセスの実行を開始するのに必要な資源が割付けられ，実行を開始し，その実行を継続中の状態である．プロセスの実行は，多くのプロセッサで行われる故，プロセスが純粋に計算だけのプロセスとして実行されている場合と，計算とそれ以外の処理，たとえば，I/o処理とが同時に行われているている場合とがある．前者を**計算処理**，後者を**混合処理**と呼ぶことにする．

・待合せ　：入出力などの待合せのため，プロセスが完全に停止している状態である．

プロセスの状態遷移図を図１に示す．プロセスは生成されると実行可状態となり，プロセスに実行開始に必要な資源が割当てられると実行中状態へと遷移する．実行中状態では，純粋に計算のみを行う計算実行中と混合処理実行状態の間を移動し，入出力などの待合せにより完全にプロセスを停止せざるを得ないか，または，混合計算を行おうとする場合で，計算を続けるよりそのプロセスを停止させた方がよいと判断された場合には，プロセスは待合せ状態へと遷移する．待ち合わせ状態で，入出力処理など，

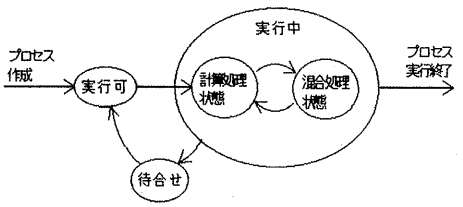

図１．ｓｓｓ－ｄにおけるプロセスの状態遷移

その待ち合わせの原因となっている動作が終了するとプロセスは再び実行可状態へと遷移する．プロセスの消滅は，実行中状態からのみ遷移する．ノイマン処理ではプロセッサが一つ故、実行中にあるプロセスはただ一つであったが，データフロー処理ではプロセッサが複数個であるので，多くのプロセスが実行中状態になること

が出来る．

３．プロセスの開始と消滅，停止制御

プロセスの実行開始は，プロセスにプロセス実行開始のための初期値データ（または，初期値トークン）を与えることによって出来る．ところがプロセスの終了に関しては少し工夫がいる．と言うのは，データフローコンピュータでは，プログラムの終了を確認する方法が確立されていない．それゆえ，プロセスの終了をＯＳが知るためには，プロセスの終了時に，それを知らせるための特殊トークンを出力するプログラムを，プロセスの最後に付加しなくてはならない．

プロセスの遷移において，実行中から待ち合わせへの遷移は，プロセスの停止という形を取る．ところがデータフローコンピュータでは，プログラムを停止することは大変難しい．というのは，データフローコンピュータは並列処理で非決定的処理を行うので，プログラム実行中，あることが起こったときプログラムを止めるとしても，実行のたびその時の状態が変わり，一意的に定まらない．また，データフローのプログラムは，データーによって駆動されるので，プロセスを停止させるには，そのプロセスに関係するデータを総て引き上げなければならない．このような制御は時間もかかるし，このような機能をデータフロープログラムによってデータフローコンピュータに組み込むことは難しい．残る方法は，全てのプロセッサに指令を出し，そのプロセスの実行を停止させることである．そのためには，データフローコンピュータにプロセッサ間通信と次に述べるプロセスの識別機構を用意しなければならない．

４．プロセスの識別機構

sss-dは，データフローコンピュータ用のＯＳであるので，並列プログラムを基本とする多重プログラミング方式をとっている．そのため，同時に複数のプロセスが実行され，しかもそのプロセスの一つ一つが多くのプロセッサによって実行される．このようなプロセスを実行中にしたり，待ち合わせ中にしたりと言うように制御するには，少なくともプロセスを識別できなくてはならない．プロセスを識別すると言うことは，どのアクタがどのプロセスに属しているかを実行中に動的に識別することである．ところがデータフローコンピュータは，実行中のプログラムの各命令であるアクタが，どのプログラムに属しているかを識別する機構を持っていない．このため，データフローコンピュータの基本構造を変えて，各プロセスを識別するための機構を準備しなければならない．データフロー処理においては，命令（アクタ）とデータ（トークン）が分離されており，プログラム実行の制御は全てトークンによってなされている．従って，各プロセスを識別し，これらを管理するためには，トークンに対してプロセスを識別するためデータを持たせなくてはならない．これはトークンにプロセス識別子（process ID）を付加することによって実現できるが［８］，すでにトークンが持っているトークンの色（トークンカラー）を利用しても実現できる．トークンカラーは，プロシジャ呼び出しやループなどで新しいインスタンスが作られるごとに新しい色につけ変えられるので，どの色がどのプロセスに属するかを記憶しておく表が必要となる．この色替えは，おのおののプロセッサによって新しいインスタンスが出来るごとに個別に行われるし，一方，この表の情報は全てのプロセッサによってアクセスされるので，表は共通メモリにつくられるか，表を作る情報が発生するごとに，それを各々のプロセッサに配り，各々のプロセッサが同じ表を持つようにしなければならない．

このように，共有する表を作るか，または，表のコピーを作り各プロセッサに配ることによる情報の共有方法を，以後情報共有手段と呼ぶことにする．

５．プロセスの優先順位処理

ノイマン処理におけるプロセスの優先順位処理とは，プロセスが実行可状態から実行中状態に遷移するとき，どのプロセスを先にするかと言う実行開始の順位であった．しかしながら，データフロー処理では複数のプロセスが実行中状態にあることが出来，また，実行中状態にある各プロセスは，複数個のプロセッサによって実行されるので，ノイマン処理の場合の実行開始順序による優先処理に加えて，プロセスに何個のプロセッサを割り当てるかと言う優先順位の決め方がある．すなわち，前者は，実行開始時間に関する優先順位処理であり，後者は，実行プロセッサの数，すなわち，実行速度による優先順位処理である．実行速度とは，単位時間に実行するプリミティブな命令の数として定義される．この数による優先順位を可能にするためには，プロセッサに、どのプロセスのアクタを優先的に実行するかを決めるための情報を与えなければならない．実行開始優先順位は，プロセス実行開始順位の表を作り，それに従って，プロセスを実行開始するのに必要な初期値データをマッチングメモリ，もしくは，トークンメモリに格納することによって行われる．実行速度優先順位は，実行開始時にプロセスにプロセッサを割り付けることによって行うことが出来る．プロセッサにどのプロセスを実行すべきかの情報を与えるには，トークンカラーと実行すべきプロセッサの関係を示す情報を，関連するプロセッサに知らせることによって可能である．これらは，情報共有手段によってなされる．

実行速度優先順位におけるプロセッサ割当において，１つのプロセスに多くのプロセッサを割り当て，実行速度をあげようとしても必ずしも期待どうりの結果が得られるとは限らない．ここで言うところの速度とは，アクタを並列に行うことによる単位時間当りの実行量であり，もし，プログラムに無限の並列度があればそれは可能であるが，プログラムの並列度が限られている場合には，そのプロセスに幾ら多くのプロセッサを割り付け，実行速度を増やそうとしても実行速度は増えない．もしこの時，一つのプロセッサを一つのプロセス専用に割り当てると，プロセスの並列度が小さく，それにもかかわらず多くのプロセッサを割り当てた場合には，多くのプロセッサが遊んでしまうことになる．このようなことを避けるためには，一つのプロセッサに多くのプロセスを割り当てることが必要になる．一つのプロセッサに複数個のプロセスを割り当てられると，プロセッサに，その内のどれを先に実行すべきかを指示しなくてはならない．すなわち，プロセスにプロセッサを割り付けるとき，プロセッサ内でどの程度の実行優先順位をもって割り付けるかを決定しなければならない．以上の事より，実行速度優先順位は，プロセスに割り当てるプロセッサの数と，そのプロセスに割り当てられたプロセッサ内での実行に対する強さによって決定される．この強さのことを，ここでは，実行強度と呼ぶ．

６．プロセス間通信

データフロー処理（色付きモデル）では，各プロセスは全て異なる色のトークンにより駆動されている．従って，プロセス間で通信を行う場合，送信側プロセス（データを送る側）のトークンの色を受信側プロセス（データを受け取る側）のトークンの色に付け換える必要がある．例えば，プロセスＡが赤色トークンにより，プロセスＢが緑色トークンにより駆動されている場合，プロセスＡからプロセスＢに送信するためには，トークンの色を赤から緑に，逆にプロセスＢからプロセスＡに送信するためには，トークンの色を緑から赤に付け換えねばならない．

データフローコンピュータでは，各プロセスは並列に処理されており，これらのプロセスを駆動しているトークンの色は動的に決る．すなわち，１つのプロセスがいくつかのプロシジャを含む場合，プロシジャコールなどによってインスタンスができるごとにトークンの色が変化する．このため１つのプロセスが複数色のトークンを持つことが一般的であり，その色も次から次へと変化する．それゆえ，送信側プロセスが，受信側プロセスの色を知ることは難しい．そのため，ここでは，プロセスの一番最初のインスタンスの間でプロセス間通信を行うことに

限定する．

プロセス間通信を実現するために，図２に示すｓｅｎｄアクタとｒｅｃｅｉｖｅアクタを導入する．ｓｅｎｄアクタはトークンを受信側プロセスに送るためのアクタであり，ｒｅｃｅｉｖｅアクタは送信側プロセスから送られたトークンを受け取るためのアクタである．ｓｅｎｄアクタは，入力に，受信側プロセス名，ｒｅｃｅｉｖｅアクタ名，そして通信文であるトークン

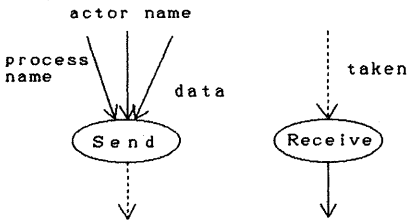

図２．センド，レシーブアクタ

を持つ．ｓｅｎｄアクタの出力は，受信側プロセスに送るトークンであり，出力アーク（動的アークであり破線で結ばれている）は，相手プロセスのｒｅｃｅｉｖｅアクタの入力に機能的につながっている．ｓｅｎｄアクタは，その入力アーク上に全てのトークンが到着すると発火し，入力のプロセス名をもとに，そのプロセスに付けられたカラーをプロセスーカラー表から求め，求めたカラーと入力のアクタ名，データとでアクタパケットを作り，これをRecieveアクタに送る．recieveアクタは，このパケットの到着により実行可能となり，プロセッサが割り当てられた時実行される．このようにして，プロセス間通信が行われる．

ＯＳプロセスとの通信は，あらかじめＯＳに定まったカラーを与えておけば，もっと簡単化することが出来る．すなわち，プロセスーカラー表のアクセスが必要でなくなるので，高速化される．

７．むすび

以上データフローコンピュータのオペレーティングシステムＳＳＳに関して，そのプロセス管理に関して基礎的考察を行った．このような研究は，ハードウエアが固まったノイマンコンピュータ上にオペレーティングシステムをのせる研究と違い，ハードウエア，ソフトウエア両面からのアプローチが必要で，簡単に決められないことも多く，残された問題も多い．今後とも努力を続けるつもりである．

文献

１．Sowa,M.,Murata,T.,"A data flow computer architecture with program and token memories,"Proceedings of 1980 IEEE Asilomaer Conference on Circuit and Systems,November,1980.

２．Arvind, and Brock,J.D.,"Resource manegers in functional programming, Journal of Parallel and Disributed Computer,1,1984.

３．林，”データフローコンピュータのＯＳに関する基礎研究”，昭和５９年度群馬大学工学部情報工学科卒業論文,SLL-840093,1985,3.

４．曽和，林，”並列オペレーティングシステム（SSS）における割り込み処理の基礎的検討”，信学研資，EC85-45，1985,10.

５．曽和，林，”並列オペレーティングシステム(SSS)の基礎的考察ーデータフローＯＳ用ファイルシステム”，３２回情報処理全国大会,3E-7，1986.

６．林，曽和，”データフローオペレーティングシステムＳＳＳにおけるファイルシステムの基礎的考察”，信学会データフローワークショップ，1986,5.

７．林，曽和，”データフロー用並列オペレーティングシステムにおける資源管理に関する考察”，情報処理学会，オペレーティングシステム研究会，33-8,1986,5.

８．林，曽和，”データフロー用並列オペレーティングシステムSSSにおけるプロセス管理に関する考察”,34回情処全大，3Q-4,1987,3.