計算機アーキテクチャ 64－2
（1987. 1. 27）

データフローコンピュータにおける デッドロックの発生

── 永久遅延問題 ──

曾和将容
群馬大学工学部

データフローコンピュータを、特に、共有の計算機として取り扱うとき実行の永久遅延を生じる可能性がある。本論文ではこのことを指摘するとともに、そのメカニズム、および解決方法についての原理的考察を行う。

DEAD LOCK PROBLEM IN DATAFLOW COMPUTERS

── INFINITE WAITING PROBLEM ──

Masahiro Sowa
Department of Computer Sicence, Gunma University
1-5-1 Tenjin-cho, Kiryu 376, Gunma, Japan

This paper points out that a infinite wating problem is occurred in dataflow computers, and also propose the fundamental method to prevent the problem.

1．まえがき

次世代コンピュータとして並列処理可能なコンピュータの研究が盛んにおこなわれている[1]。これらは、並列処理と言う性格上非決定的処理を行う場合が多い。我々もデータフローコンピュータの研究をおこなってきており[2]-[5]、データフローコンピュータ用のオペレーティングシステムの研究も行われてる[6]-[8]。データフローコンピュータ上で単独のプログラムを実行するときは大きな問題を生じないが、これをオペレーティングシステムを通して見たとき、データフローコンピュータは多くのユーザから使われる共有資源としての性格を持ってくる。データフローコンピュータが共有資源として用いられる時、そこでは、実行の永久遅延が発生することが明らかになっている[9]。本論文では、これに関する第一報として、永久遅延の発生原理とその原理的な阻止方法について報告する。

2．非決定的コンピュータにおける永久遅延の発生

2．1　データフローコンピュータにおけるアクタの命令の実行

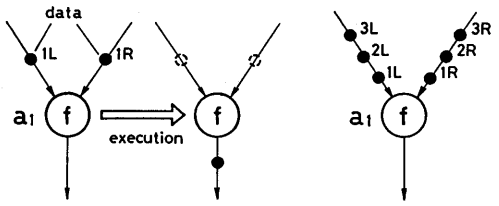
データフローコンピュータでは、図1（a）に示すように、アクタと呼ばれる命令に必要なデータ（この図では1L、1Rの2個）が到着したときアクタは実行可能となり、ここにプロセッサが割り当てられた時アクタが実行され、出力アークに結果のデータを出力してアクタの実行を終る。実行可能なアクタが複数個あるときは、このうちの複数個がアイドル中の複数のプロセサによって選らばれ実行される。実行可能アクタの選択順序は特に決められていない。すなわち、非決定的である。以後、説明の便宜のために、アクタを一回実行するのに必要な（1L、1R）のようなデータのことを、必要データと呼ぶ。

2．2　アクタでの発生

いま、同図（b）のように、複数個のデータが入力アークに到着しており、これ以上のデータ到着がその後ないと仮定すると、一時には、（1L、1R）または（2L、2R）、（3L、3R）のうちの1組に対してアクタの実行が行われ、3個の必要データに対して実行が終った時、アクタa_1に対する総ての実行を終了する。たとえば、（2L、2R）→（1L、1R）→（3L、3R）のデータ順にアクタの実行が行われ、到着した総てのデータに対してのアクタ実行を終る。しかし、もし、次から次へとデータ対が到着しており、アクタの入力アーク上に常に複数個の必要データがあるとすると、必要データに関するアクタの実行は、通常任意の必要データ（したがって、ランダムに取りだされた必要データ）に対して行われるので、たとえば、（2L、2R）に対するアクタ実行のまえに、後から到着した必要データに対するアクタの実行が次から次へと行われ、（2L、2R）に対するアクタ実行が永久に待たされることがおこりうる。たとえ永久遅延に致らなくとも、かなり長期間の遅延が発生する可能性がある。

マルチプログラミングを行わない普通の処理では、1つのプログラムの処理が終って結果を得てから次の処理に移ると言うことを繰り返すので、入力アークに必要データが永久に残ると言うようなことは起こらないが、マルチプログラミングやＴＳＳ処理におけるように共有資源を多くのプロセスで使う場合、共有資源使用要求であるデータが、次から次へと共有資源管理プログラムに到着するので、その中のアクタにおいて、上記のような現象がおき永久遅延の問題が発生する。また、ダイナミックアーキテクチャ方式のデータフローコンピュータにおいては、プロシジャの並列呼出時におけるデータの交りあいを避けるために、データの色付けが行われている。この場合にも、アクタの入力アーク上に複数個の色の違った必要データがおかれるので、これらの内の特定な必要データが永久に取り出されないという永久遅延の問題が起こりうる。

図2のマージアクタは、どちらかの入力アークにデータが到着したとき実行可能になり、プロセッサが割り当



（a）アクタの実行　　（b）

図1　　アクタでの永久遅延の発生

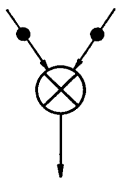


図2　　マージアクタでの永久遅延の発生

てられたとき実行される。入力アークの両方にデータが到着する場合は定義されていない。すなわち、両方のアクタに同時にデータが到着しないようにプログラムを書くか、もしくは、同時に到着した場合それを検出する機能をコンピュータに組み込まなければならない。もしこのようなことが考慮されていない場合、マージアクタにおいて永久遅延が起こりうる。たとえば、図において、両入力アークにデータが到着していて最初、左側の入力アークに対してアクタが実行されたとする。アクタ実行中に左入力アークに再びデータが到着し、最初のデータに対するアクタの実行が終了する時には、図に示す最初の状態に戻っていたとして、次のアクタの実行が再び左側の入力アークのデータに対して行われたとする。このように、左入力アークのデータに対してアクタの実行が繰り返される時、右側のデータに対するアクタの実行は永久に遅延される。

２．３　プログラム内での発生

必要データが到着している実行可能アクタが複数個あるとき、一つのプロセッサはこの中から一つのアクタを選んで実行する。この時、プロセッサの数より実行可能アクタの数が常に多ければ、実行されずに残る実行可アクタが常に発生し、もしそのうちの特定のアクタが常に選らばれないなら、そのアクタの実行は永久に遅延される。このような永久遅延は、個別のプログラムでは起こらない。というのは、データフロープログラムのアクタはデータの従属性にしたがってアークにより接続されている。したがって、もしあるアクタが長時間遅延されているとすると、それ以降のアクタが実行可能になることができないので、いつか実行可能アクタの数がプロセッサの数より小となり、条件１が阻止される状態となるからである。ところがマルチプログラミングのように複数個のプロセスが走っており、次から次へとプロセスが追加されている状態では、ある一つのプロセスで条件１が阻止されても、全体としては条件１が阻止されないので永久遅延が起こる可能性がある。たとえ永久遅延に致らなくとも、長時間遅延されることは十分ありうる。

以上の現象は、次のようにモデル化できる。いま、図３に示すように複数個（個数可変）の入口から必要データに相当するトークンが待合室に到着しており、一方、待合室から複数個のトークンが取り出されているモデルを考える。このとき待合室で、少なくとも１個のトークンに永久遅延が起こりうるのは、次の条件が成り立つときである。

定理１：永久遅延発生の必要条件は次のようである。

条件１

トークンが常に１個以上待合室に残っていること。

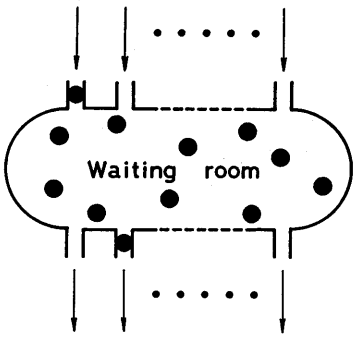


図３　永久遅延機構のモデル化

条件２

トークンの取り出し順序が、原則的にＦＩＦＯ（到着順処理）でないこと。

トークンが待合室に残っていないことは、永久遅延が起こっていないことを意味しているので、条件１は自明である。また、条件２のトークンの取り出し順序がＦＩＦＯであれば、到着したトークンは到着順に必ず取り出されるので、永久遅延が起こらないことは自明である。原則的にＦＩＦＯとは、部分的にはＦＩＦＯでなくとも、全体的に見た場合ＦＩＦＯである時も含むことを意味する。

３．アクタにおける永久遅延防止

永久遅延を防止するには、上記の必要条件の成立を阻止すればよい。

１）アクタの入力アークに、最大１個のデータしか到着しないようにプログラム全体の実行を制御する。

アクタの出力アークにデータがあるかないかを調べることは、セーフティチェック[3]と呼ばれる。このセーフティチェックを導入し、もし出力アーク上にデータがある場合には、そのデータが消費されるまでアクタの実行を延期すれば、入力アーク上にただ１つの必要データしか存在しないようにすることができる。この必要データに関してアクタが実行されると、入力アーク上の必要データがゼロとなり、条件１の成立が阻止される。通常は、このチェックは同じデータの色に対してなされるが、永久遅延をなくすためには、異なる色のデータに対しても行う必要がある。この方法は、条件１とともに条件２の成立をも阻止する。

２）必要データの個数がゼロになる時間を作る。

ある時間間隔で必要データの個数をゼロにする。この

ことは、必要データの遅延時間がその時間間隔内になることを表す。この方法もまた、条件１、条件２の成立を阻止している。

３）必要データの取り出しをＦＩＦＯ方式で行う。

ＦＩＦＯは到着順処理であるので、到着した必要データはある時間待った後必ず取り出される。この方法は条件２を阻止する。

４．プログラムにおける永久遅延防止

１）実行可能アクタ数がゼロになる瞬間を作る。

プログラムの並列度を調整、または、プロセスの生成数を調整し、実行可能アクタの数がプロセッサの数より小となる瞬間があるように変換する。ただし、並列度の調整は、データフロープログラムの場合常に可能とは限らないので、難しい場合もある。これは条件１成立の阻止である。

２）実行可能アクタの実行をＦＩＦＯに行う。

アクタは必要なデータが到着したとき実行可能となる。実行可能となったアクタをキューに格納しておけば、ＦＩＦＯ実行が出来る。ただし、データフローコンピュータは並列処理コンピュータであるので、同時に複数のアクタが実行可能になる。したがって、これらのアクタを同順位に格納取出しができる２次元キューが必要である。この方法は条件２の成立を阻止している。

５．むすび

データフローコンピュータにおける永久遅延の発生は深刻な問題である。たとえ永久遅延とまでいかなくとも長時間遅延は、リアルタイム処理やＯＳプログラムにとっては問題が大きい。

永久遅延を阻止するには、本論文の第３、４節の阻止法の両方を満足させなければならない。本論文では、問題発生の基本原理とその原理的解決方法のみに焦点をしぼって述べてきたので、これらの阻止方法のうちどれが一番実現性があり、どれが効率的であるかなど具体的な解決法については述べなかった。これに関しては次回に報告する予定である。永久遅延は、本論文に述べたアクタレベルだけではなく、ＯＳのプロセス管理においても発生することが報告されている[9]。本論文はデータフローコンピュータでの永久遅延のみについて述べたが、同様な永久遅延は、非決定的動作を行うコンピュータならば起こりうることも報告されている[9]。

参考文献

[1] Tiberghin, J.,"New computer architecture," Academic Press, 1984.

[2] 曽和；高レベルデータフローコンピュータシステム — GMMCS —，電子通信学会電子計算機研究会資料，ＥＣ７８－１１（１９７８－１１）．

[3] Sowa, M., and Murata, T.,"A data flow computer architecture with program and token memories," IEEE Transactions on Computers, Vol.C-31, No.9, pp.820-824, September 1982.

[4] Sowa, M., Ramos, F. D., and Murata, T., "Construction and structure of dataflow computer DFNDR-1 and subroutine implementation," Proc. of IEEE First International Conference on Computers and Applications, pp.490-497, June 1984.

[5] Sowa, M.,"A speeding-up method of serial processings in dataflow computer by means of a program counter," IEE The Computer Journal, Vol.29, No.4, 1986.

[6] 曽和，林；並列オペレーティングシステム（ＳＳＳ）における割込み処理の基礎的検討，電子通信学会電子計算機研究会報告，ＥＣ８５－４５（１９８５－７）．

[7] 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，電子通信学会データフローワークショップ論文集，６－４（１９８６－５）．

[8] 林，曽和；データフローオペレーティングシステムにおける資源管理の基礎的考察，情報処理学会オペレーティングシステム研究会資料，８６－ＯＳ－３３，８（１９８６－１１）．

[9] 曽和；非決定的コンピュータにおける永久遅延の発生とその防止に対する原理的考察，群大ＳＬレポート，ＳＬＬ８６００３９（１９８６－１０）．