計算機アーキテクチャ 59−6
オペレーティング・システム 29−6
(1985・10・24)

並列処理コンピュータにキャッシュメモリとパイプライン処理を導入した場合の性能評価

Performance Evaluation of Parallel Computer with Cache Memory and Pipeline Processing

曽和　将容，　　渡部　良浩
Masahiro Sowa，　Yoshihiro Watanabe

群馬大学　工学部
Department of Computer Science, Gunma University

1．まえがき

コンピュータの応用範囲が拡がるにつれて、コンピュータには、より高速性、より高性能性が要求されるようになっている。そのために、並列処理を主体とした非ノイマン型と呼ばれるコンピュータ[1]-[8]の研究が精力的に続けられている。これら非ノイマン型コンピュータのあるものは試作[2],[4],[8]され、また、あるものはシミュレーションによって、ある特定のプログラムを実行した場合の性能測定が行なわれている。しかしながら、コンピュータの性能はプログラムの性質によって大きく変るので、必ずしも、コンピュータそのものの基本性能、すなわち、ハードウェア上の性能を反映しているとはいいがたい面もあった。この基本性能を知るため、プログラムを理想化し、コンピュータの構造を簡単化して、データフローコンピュータ（ＤＦＣ）、パラレルコントロールフローコンピュータ（ＰＣＦＣ）についてノイマンコンピュータ（ＮＣ）と比較しようとする試みがなされている[9]。本論文では、これらのコンピュータにキャッシュメモリやパイプライン処理を導入し、これらの導入によってどの程度の性能改善が得られるかを求める。

2．プログラムのモデル化

プログラムの実行時間に対する影響を避け、極力ハードウェアに基づいた性能を知るために、プログラムを理想的にモデル化する。図１にモデル化したｐ並列プログラムを示す。○印は処理を表わし、ここではアクタと呼ぶことにする。矢印は命令の実行順序を表わし、ここでは矢印のことをアークと呼ぶことにする。このプログラムでは、２本の入出力アークを持つアクタが、横にｐ個、縦に無限に並んでおり、各アクタの処理は、そのアクタに入力されているアークに連なるアクタがすべて実行された後に、実行が開始される。プログラムの並列度ｐは、全てのプロセッサが同時に動作できるように、十分大きくとられていると仮定する。各アクタの実行時間はすべて等しいとし、もし、ある行のアクタすべてに同時にプロセッサが割り付けられた場合には、その行のアクタは同時に実行が開始され、同時に実行が終了するものとする。このように仮定すると、プログラムの実行は、各行ごとに行なわれると考えることができるので、解析が容易になる。今後の便利のために各行をステージと呼ぶことにする。実際にはアクタの処理時間はすべて一定ではないことが多いが、アクタ内の処理を基本命令に限定し、プロセッサがシステムクロックに同期して働いていると考えると、この仮定は、それほど現実から離れた仮定でもない。

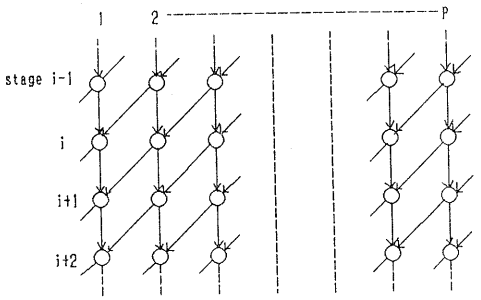


図１　　プログラムのモデル化

3．ノイマンコンピュータＮＣのモデル化とプログラムの実行時間

ＮＣは、図２に示すようにモデル化できる。中央処理装置ＣＰＵには、ＰＣとＰＵが含まれているがここでは部合上分離する。ＰＵにはｒ個のアキュムレータに相当するレジスタが含まれており、また、ＰＣとＰＵ間は高速バスで、ＰＵとM間はメモリのアクセスタイムに相当する速度のバスにより接続されているとする。コンピュータは３２ビットを基本語としたコンピュータであり、命令とデータは図３に示す形式でメモリに格納されるものとする。この図でinstは命令部、opr 1 、opr 2 はオペランドを表わし、opr 1 は主にレジスタを示し、opr 2 は主にメモリ参照番地を示すために用いられるとする。インデックスレジスタやベースレジスタなどは、ｒ個のレジスタとは別に用意されているものとする。また、ＮＣでは、ロード・ストア命令（ＬＳＩ）、加減算命令など（ＡＳＩ）、レジスター レジスタ命令（ＲＲＩ）は、図４（ａ）、（ｂ）、（ｃ）に示されるアルゴリズムで行われる。このモデルによる１ステージの実行時間t_nは、文献９により次のように求められている。

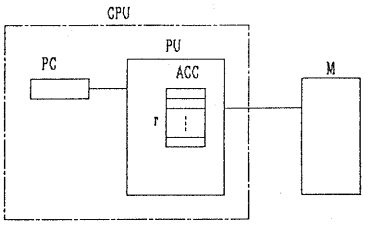


図２　　ノイマンコンピュータのモデル化

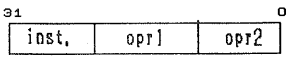


図３　　ノイマンコンピュータの命令形式

$p < r$のとき

$$t_n = p(3+m+c)u \qquad (3.1)$$

$p = r$のとき

$$t_n = \{p(3+m+c)+2m-1.5\}u \qquad (3.2)$$

$p > r$のとき

$$t_n = \{p(6m+c+9)-r(5m+6)-(m+3)\}u \qquad (3.3)$$

ここで、ｕ、ｍ×ｕ、ｃ×ｕは、それぞれレジスタ操作、メモリ操作、計算にかかる時間で、ｐはプログラムの並列度である。

4．データフローコンピュータ　ＤＦＣ

4．1　モデル化

ＤＦＣ[6),(10)]では、データの到着により命令が実行可能になるので、データ格納部が実行可命令指示器となる。これをトークンメモリＴＭと呼ぶことにすると、ＤＦＣは図５のようにモデル化される。図でPU_1～PU_nはプロセッサ群を表している。Ｍは、メモリを表す。ＴＭとＰＵ、ＰＵとＭ間は、スイッチングネットワークＳ

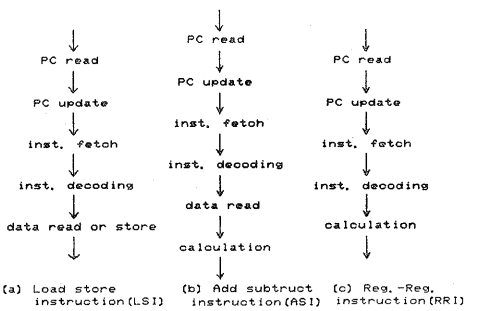


図４　　ＮＣの実行シーケンス

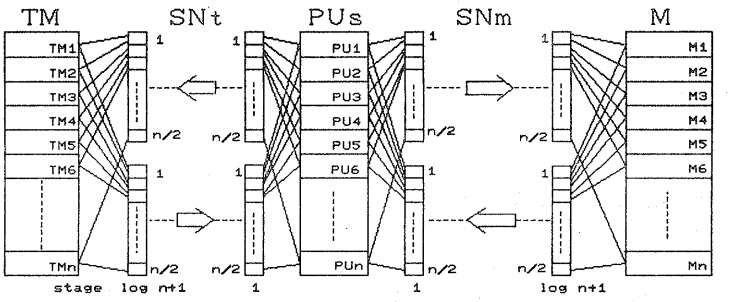


図５　　ＤＦＣ，ＰＣＦＣのモデル化

N_t 、SN_m で結ばれている。メモリMは、M_1 ～M_n のｎ個のブロックに分割されている。ＤＦＣのアクタ形式は図６（ａ）のようにアクタパケットＡＰと呼ばれる形式でモデル化される。また、データはトークンと呼ばれ、トークンは、同じアクタに入力されるものが集められ同図（ｂ）のような形式でモデル化される。このような、トークンの集合は、トークンパケットＴＰと呼ばれる。ＡＰにおいて、ｆはファンクション（命令）を示し、const は定数トークンを、des 1 、des 2 はアクタ実行終了後、結果トークンを送り出すアクタ名（destination node ）、すなわち、出力アークを表す。トークンパケットＴＰは、ＴＰ内のトークンが入力されるべきアクタを示すアクタポインタＡＰＲ、トークンの色（color ）、トークンの状態を表すステータスデータS_d 、トークン（token 1 、token 2 ）よりなっている。あるアクタの処理に必要なトークンがそろったＴＰは、そのＴＰが入力されるべきアクタが実行可能であることを表わしている。このようなＴＰは完全トークンパケットＣＴＰ（complete TP ）と呼ばれる。ＴＰは、トークンメモリＴＭに格納される。メモリの各ブロックには、複数のアクタパケット（ＡＰ）が格納され、１つのアクタパケットは、２つ以上のブロックにまたがって格納されることはないものとする。また、スイッチングネットワーク内で、パケットの衝突を少なくし、実行スピードを上げるために、同時に発火可能なアクタを、それぞれ別々のブロックに格納する。トークンメモリＴＭも、メモリと同様にｎ個のブロックよりなっており、各ブロックは、それぞれ連想メモリとなっている。ネットワークには、Multipath Multistage Interconnection Networks (MMINs)[12] が用いられている。このネットワークは、セルと呼ばれる２×２のスイッチングネットワークから成っており、例えば、プロセッサの台数がｎの場合、縦にｎ／２個、横に$\log_2 n+1$個セルを並べた構成になっている。このセルの構造は、図７に示されている。以後、ＭＭＩＮｓの各列をステージと呼ぶことにする。このネットワークでは、入出力間に複数のパスを作ることができるので、ネットワーク中での衝突を減少させることができる。メモリMに直接接続されているセルには、メモリのリード、ライト機能が組み込まれているものとする。

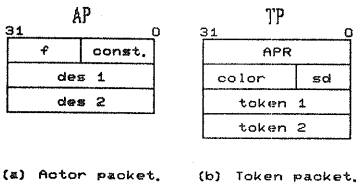


図６　　ＤＦＣのアクタとトークンの形式

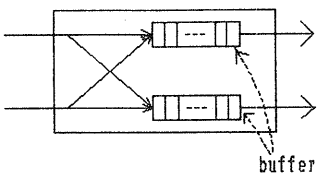


図７　　セルの構造

４．２　プログラムの実行と実行時間

ＤＦＣにおけるＰＵの動作は、図８のようになる。ＤＦＣの実行シーケンスは、トークンメモリ内の完全トークンパケットＣＴＰを見つけ出し、SN_t を通してＰＵに転送することから始まる（ＣＴＰフェッチサイクル）。ＣＴＰを受け取ったＰＵは、ＣＴＰのアクタポインタＡＰＲをもとにメモリMにアクタパケット（ＡＰ）を要求する。その要求は、SN_m を通してメモリに達し、それによってメモリからアクタパケットが１ワードづつ送り出される。送り出されたアクタは、SN_m を通してＰＵに送られる（ＡＰフェッチサイクル）。アクタがＰＵに到達すると、命令の実行が開始され(Calculation サイクル）、その結果は、des 1 、des 2 で表されるアクタに送られる。実際には、送られる結果であるデータは、行き先アーク名と色を接続したトークンパケットＴＰとして、ＴＭに送られる。ＴＭに送られたＴＰのうち、同じアーク名とカラーを持つＴＰは、ＴＭ内の連想機能により、一語にまとめて格納される(ＴＰライトサイクル)。以上の各サイクルは、さらに図９に示すように細かく分解することができる。この図で直線の下に記されたｕ、ｂｕ等は、時間を表わしている。いま、図８の各サイクルの実行時間を、それぞれＤＣＴＰＦＴ、ＤＡＰＦＴ、

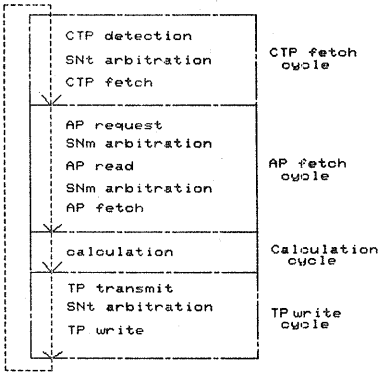


図８　　ＤＦＣの実行シーケンス

ＤＣＴ、ＤＴＰＷＴとすると、それぞれの時間は、図９より

DCTPFT＝6u+bu
DAPFT＝2u+3mu+2bu
DCT＝cu
DTPWT＝7u+bu+au

と求めることができる。したがって、図１のプログラムの１ステージの実行時間t_dは、

$$t_d = DCTPFT + DAPFT + DCT + DTPWT \qquad (4.1)$$

となる。ここで、ａ×ｕ、ｂ×ｕ、ｍ×ｕは、連想時間、調停時間、メモリアクセス時間とする。また、ＰＵとスイッチングネットワーク間のパケットの転送時間をｕとする。

５．パラレルコントロールフローコンピュータＰＣＦＣ

５．１　モデル化

ＰＣＦＣ[7),(10)]も、ＤＦＣと同様図５のようにモデル化される。ＴＭに、データを表すトークンのかわりに実行制御を表すトークン（コントロールトークン）が格納されることと、アクタパケットにデータを指示するためのオペランド部があることと、データがメモリＭ内に格納されることを除けば、ＤＦＣとほとんど同じ構成である。コントロールトークンは、あるかないかを表せばよいので１ビットで表わされる。このコンピュータのアクタとトークンは、ＤＦＣの時と同様にトークンパケットＴＰ、アクタパケットＡＰとして、図１０のようにモデル化される。アクタパケットＡＰには、データの格納場所を示すオペランドopr 1 、opr 2 、opr 3 と、このＡＰ終了後に実行されるべきアクタを示すポインタであるdes 1 、des 2 が接続されている。ＰＣＦＣでは、アクタの実行終了後、このdes 1 、des 2 に従って、制御権を示すコントロールトークンがＴＭに送られる。

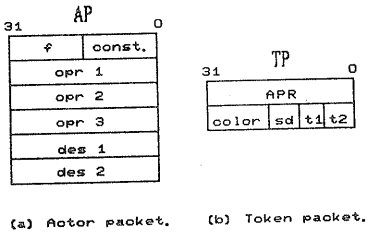


図１０　ＰＣＦＣのアクタとトークンの形式

５．２　プログラムの実行と実行時間

ＰＣＦＣの動作は、図１１のようになされる。まず、ＴＭでＣＴＰが見付けだされ、ＰＵに送りだされる（ＣＴＰフェッチサイクル）。ＣＴＰを受け取ったＰＵは、

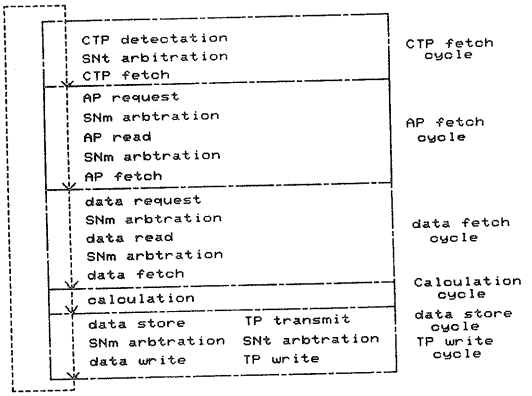


図１１　ＰＣＦＣの実行シーケンス

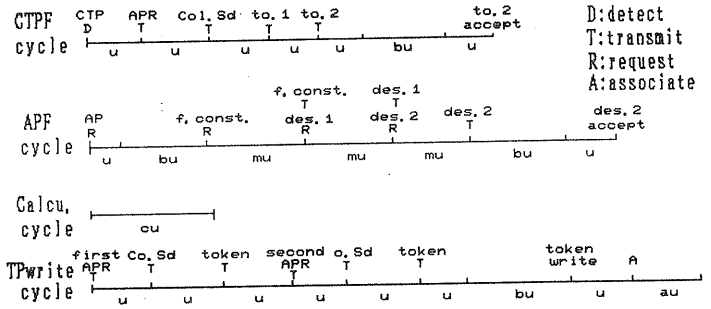


図９　ＤＦＣの実行シーケンスの詳細

ＣＴＰのＡＰＲによって示される命令――アクタパケット（ＡＰ）――をメモリＭへ要求する。要求がメモリＭに到着すると、メモリＭは、アクタパケットＡＰを１ワードづつＰＵに転送する（ＡＰフェッチサイクル）。ＡＰがＰＵに到着したなら、ＡＰのファンクションｆを実行するため、opr 1 、opr 2 によって示されるデータをメモリＭから取り出すため再び要求を出す。アクタパケットフェッチと同じように、データがメモリから読み出され、ＰＵに送られると（ｄａｔａフェッチサイクル）、ＰＵは、命令の実行を行い（Calculation サイクル）、その結果をopr 3 で示されるメモリ内に書き込み（ｄａｔａストアサイクル）、コントロールトークンに行先きアクタdes 1 、des 2 を接続して、ＴＰとしてＴＭへ転送する。接続されたdes 1 、または、des 2 は、トークンパケットのアクタポインタＡＰとなる。ＴＭに送られたＴＰのうち、ＡＰＲとｃｏｌｏｒが同じものはまとめられ、ＴＭの１ワードに格納される（ＴＰライトサイクル）。ｄａｔａストアサイクルとＴＰライトサイクルは、Calculation サイクル終了後、それに必要なデータは、全てそろっているので同時に実行できる。各サイクル時間を、それぞれＣＣＴＰＦＴ、ＣＡＰＦＴ、ＣＤＦＴ、ＣＣＴ、ＣＤＳＴ、ＣＴＰＷＴとする。これらのサイクルの実行時間は、ＤＦＣの時と同様にして、

$CCTPFT = 4u+bu$

$CAPFT = 2u+6mu+2bu$

$CDFT = 2u+2bu+2mu$

$CCT = cu$

$CDST = u+bu+mu$

$CTPWT = 5u+bu+au$

と求めることができ、１ステージの実行時間t_cは、次のようになる。

$$t_c = CCTPFT+CAPFT+CDFT+CCT+Max\{CDST,CTPWT\} \quad (5.1)$$

ここで、Ｍａｘは、要素の最大値を求める関数とする。

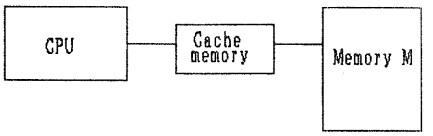


図１２　　ＮＣのキャッシュメモリの配置

６．　キャッシュメモリとパイプライン処理の導入

キャッシュメモリは、メモリアクセス時間を短縮するために考えだされたメモリである。キャッシュメモリの配置は、ＮＣの場合、図１２に示すようにメインメモリＭとＣＰＵの間に設け、ＤＦＣ、ＰＣＦＣの場合は、図１３のようにＳＮmとメモリＭの間に設け、各Ｃａｃｈｅメモリブロックには、対応するメモリブロックの内容を入れる。

６．１　ＮＣへのパイプライン処理の導入

図１４に、パイプライン処理を行なうためのＣＰＵの構造を示す。各ユニットは、図４の各動作を担当しており、ユニット内には、各ユニットの処理速度の違いを調整するためのバッファが装備されている。図の○印から入力された命令とデータは、デコードを通ったのち、命令の種類により、図のように、３つの異なったユニットにふり分けられる。また、計算ユニットは、○にＡＳＩ、ＲＲＩの順序でデータ及び命令が入力された場合、それらの関数を同時に実行できるように、図のようにCalculation 1，2と２つ設ける。ユニット内に書かれている２ｕ、ｕなどの記号は、そのユニットの処理時間を表わす。

パイプライン処理では、各ユニットは、お互いに同期

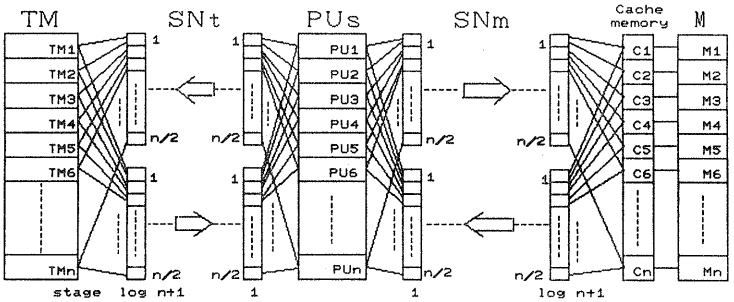


図１３　　ＤＦＣ，ＰＣＦＣのキャッシュメモリの配置

して動かなくてはならないので、同期間隔をユニット内の処理の最大値にしなければならない。ところが、inst fetch ユニットと data load store ユニットは、同一のメモリをアクセスするので、メモリ競合がおこる。このため、メモリアクセスを順番にせざるをえないので、メモリアクセス時間は、実質上２倍となってしまう。したがって、いまt_{np}１～3を各動作の同期時間とすると

ＲＲＩの場合　$t_{np}1=Max\{2,m,1,c\}u$　(6.1)
ＬＳＩの場合　$t_{np}2=Max\{2,2m,1\}u$　(6.2)
ＡＳＩの場合　$t_{np}3=Max\{2,2m,1,c\}$　(6.3)

となる。これらの式と文献９のＮＣのプログラムの実行形態の分類より、パイプライン処理時の実行時間t_{np} は、

p＜rの時
$t_{np}=p*t_{np}1$　(6.4)
p＝rの時
$t_{np}=\frac{1}{2}(t_{np}1(2p-1)+2*t_{np}2+t_{np}3)$　(6.5)
p＞rの時
$t_{np}=t_{np}1(r-1)+t_{np}2(p-r+1)+t_{np}3(2p-2r-1)$　(6.6)
となる。

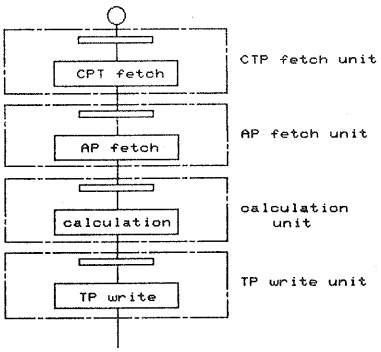


図１５　ＤＦＣのパイプライン処理

６．２　ＤＦＣへのパイプライン処理の導入

ＤＦＣのＰＵの部分を、ＤＦＣでは、ＰＵを図１５のように、４つのユニットから構成する。各ユニットは、図８に表わされている各サイクルを実行する。ＣＴＰｆｅｔｃｈユニットは、ＣＴＰになったＴＰをＴＭから取り込み、その情報をＡＰｆｅｔｃｈユニットに送る。ＡＰｆｅｔｃｈユニットは、ＣＴＰｆｅｔｃｈユニットから送られてきたアクタのＡＰをフェッチするユニットである。フェッチを終了したらCalculation ユニットにデータを転送する。Calculation ユニットは、送られてきたデータをもとに、そのアクタの関数を実行する。関数の実行終了後ＴＰｗｒｉｔｅユニットにデータを転送し、ＴＰｗｒｉｔｅユニットは、そのデータをＴＭに格納す

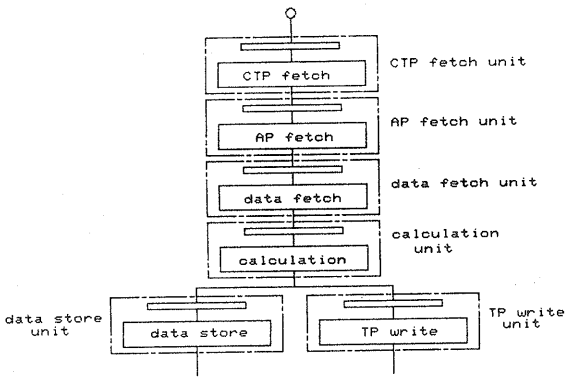


図１６　ＰＣＦＣのパイプライン処理

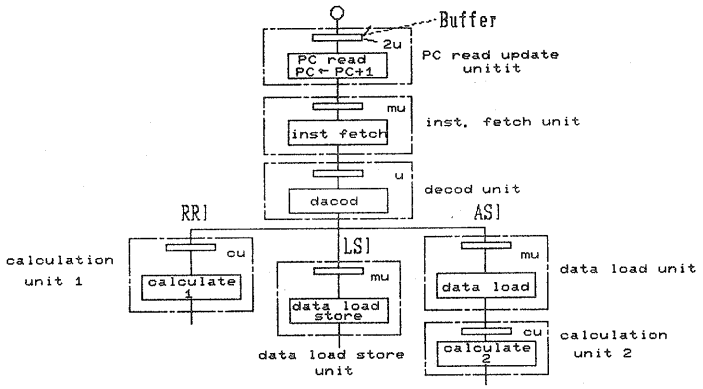


図１４　ＮＣのパイプライン処理のためのＰＵの構造

る。ここでもＮＣのときと同様に、各ユニットは、それぞれのユニット内の処理時間の最大値に同期して動かなければならない。したがって、プロセッサの台数をｎとすると、並列度ｐのプログラムの１ステージの実行時間 t_{dp} は、

$$t_{dp}=\mathrm{Max}\{DCTPFT,DAPFT,DCT,DTPWT\}\times p/n \quad (6.7)$$

となる。

６．３　ＰＣＦＣへのパイプラインの導入

ＰＵを図１６のように６つのユニットから構成する。ＰＣＦＣでは、ＤＦＣと違ってデータの取り出しと、書込みが必要なので、data fetchユニットと、data storeユニットが用意されている。また、ｄａｔａ　ｓｔｏｒｅとＴＰｗｒｉｔｅが同時に実行できるので、そのユニットを並列に接続する。したがって、パイプラインのステージ数は５段となる。プロセッサの台数をｎとすると、並列度ｐのプログラムの１ステージの実行時間 t_{cp} は、

$$t_{cp}=\mathrm{Max}\{CCTPFT,CAPFT,CDFT,CCT,\mathrm{Max}(CDST,CTPWT)\}\times p/n \quad (6.8)$$

となる。

ＮＣの処理速度に対するＤＦＣ、ＰＣＦＣの処理速度増加率を S_{dr}、S_{cm} とすると、キャッシュメモリ、パイプライン処理を導入しない場合には、式（３．１）～（３．３）、（４．１）、（５．１）より、$S_{dr}=t_n/t_d$、$S_{cm}=t_n/t_c$ となる。また、ＮＣには、パイプライン処理を導入せず、ＤＦＣ、ＰＣＦＣに導入した場合は、式（３．１）～（３．３）、（６．７）、（６．８）より、$S_{dr}=t_n/t_{dp}$、$S_{cm}=t_n/t_{cp}$ となり、ＮＣ、ＤＦＣ、ＰＣＦＣにパイプライン処理を導入した場合は、式（６．４）～（６．８）より、$S_{dr}=t_{np}/t_{dp}$、$S_{cm}=t_{np}/t_{cp}$ となる。また、キャッシュメモリを使用した場合の処理速度増加率は、キャッシュメモリを使用したコンピュータの実行時間中のメモリアクセス時間ｍ×ｕをｃｍ×ｕに置換えることによって得ることができる。

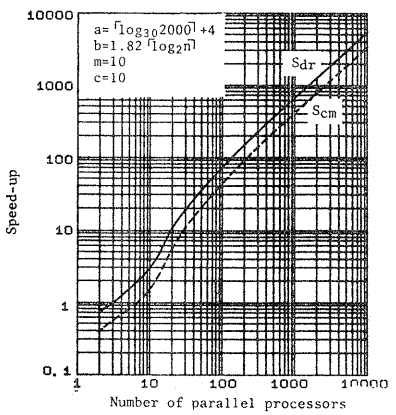


図１７　一般的な比較

７．　比較

１ブロック２０００ワードのＴＭの連想時間ａ×ｕを、ａ＝⌈ｌｏｇ$_{30}$２０００⌉＋４とし、調停時間ｂ×ｕを、ｂ＝１．８２⌈ｌｏｇ$_2$ｎ⌉ [12] とする。このように設定した場合で、キャッシュメモリもパイプライン処理も導入しない場合（ノーマルな処理と呼ぶことにする）の比較を図１７に示す。ＮＣとの比較は、ほぼ、プロセッサの台数に比例して処理速度が増加している。例えば、プロセッサの台数が１０２４の時、ＤＦＣは、ＮＣの６４０倍の性能を持ち、ＰＣＦＣは、４１０倍の性能を持っている。また、ＰＵ数が７以上では、処理速度の順番がＤＦＣ、ＰＣＦＣ、ＮＣになっている。ＤＦＣ、ＰＣＦＣのどちらも、スピードアップの割合は、ｎが小さいところでは、大きく、ｎが大きくなるほど、小さくなるのは、ネットワークの遅延のためと考えられる。また、ＤＦＣの方がＰＣＦＣの約１．６倍のスピードを持つが、これは、ＰＣＦＣのメモリのアクセス回数がＤＦＣの３倍あるためと考えられる。ｎが１２以上での特性の特異な変化は、ＮＣのレジスタ数の制限が原因であると考えられる。

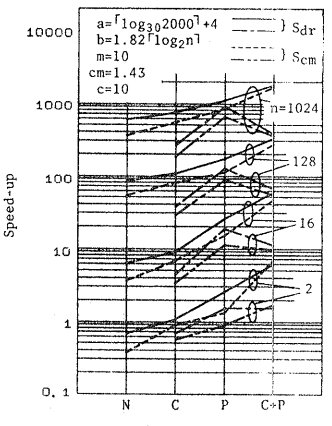


図１８　キャッシュメモリとパイプライン処理の導入による効果

７．１　キャッシュメモリとパイプライン処理による影響

１）　ＤＦＣとＰＣＦＣにキャッシュメモリとパイプライン処理を導入した場合

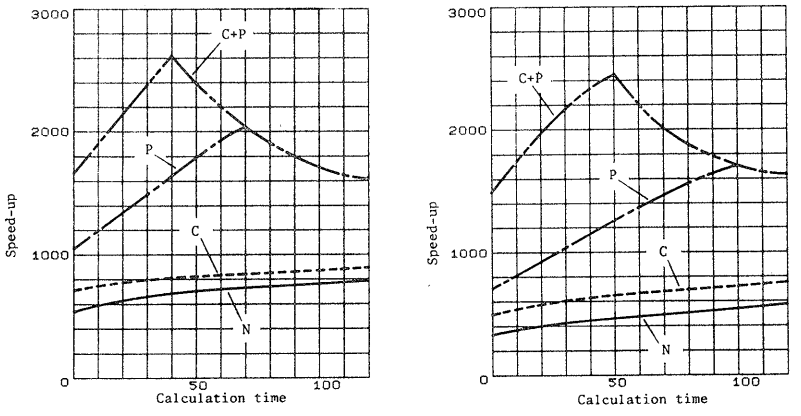
ＤＦＣ、ＰＣＦＣにキャッシュメモリ、パイプライン処理を導入して得られた結果をＰＵ数２～１０２４の場合について、図１８の実線と破線に示す。横軸のＮは、各コンピュータがノーマルな処理の場合、Ｃは、キャッシュメモリを導入した場合、Ｐは、パイプライン処理を導入した場合、Ｃ＋Ｐは、キャッシュメモリとパイプライン処理を併用した場合を示している。この図より、ＰＵが１０２４台の場合をみると、ＤＦＣでは、キャッシュメモリの導入により、１．２倍、パイプライン処理の導入により、２．０倍、キャッシュメモリとパイプライン処理を併用した場合には、３．２倍となり、ＰＣＦＣでは、それぞれ、１．５倍、２．２倍、４．７倍となっていて、ＰＣＦＣの方がＤＦＣよりも、全ての効果が大きく表われている。これは、ＰＣＦＣは、ＤＦＣよりもメモリのアクセス回数が多いこと、更にパイプラインのステージ数が多いためである。実行時間に対する、ネットワーク遅延の割合は、ノーマル処理の場合よりも、キャッシュメモリやパイプライン処理を導入した場合の方が大きい。そのため、ＰＵの台数を増やすほど、キャッシュメモリやパイプライン処理の効果は、低下してしまう。

２）　ＮＣ、ＤＦＣ、ＰＣＦＣがキャッシュメモリとパイプライン処理が可能な場合

この結果は、図１８の一点破線と二点破線で示される。図より、ＰＵ数が２の時を除きＤＦＣ、ＰＣＦＣ、共にキャッシュメモリ使用は、ＮＣの方が良く、パイプライン処理は、ＤＦＣ、ＰＣＦＣの方が良い。まず、キャッシュメモリ使用時にＮＣが良い理由は、全体の実行時間に対する、メモリアクセス時間の割合が、ＮＣが他のコンピュータより大きいため、キャッシュメモリを使用すれば、メモリアクセス時間が擬似的に減少するので、それだけＮＣの方が大きい効果が得られる。次に、パイプライン処理でＤＦＣ、ＰＣＦＣの方が効率が良い理由であるが、これは、パイプラインの処理方法の違いによる。ＮＣでは、パイプライン処理を行っても同時に命令のフェッチとデータのロードまたは、データのストアができないものとしているので、メモリアクセスがボトルネックになっている。しかし、ＤＦＣ、ＰＣＦＣは、バケットのフェッチ要求が同時に実行できるものとしているため、ＤＦＣ、ＰＣＦＣの方が効率が良い。また、キュッシュメモリ＋パイプライン処理の場合は、前の２つを合せたような結果になっている。

７．２　計算時間による影響

ＤＦＣ、ＰＣＦＣのプロセッサの台数が１０２４台の場合で、計算時間ｃを０～１５０まで変化させた時のスピードアップの割合をＤＦＣ、ＰＣＦＣそれぞれを図１９（ａ）、（ｂ）に示す。図で実線は、ノーマルな処理（Ｎ）、破線は、キャッシュメモリを導入した場合（Ｃ）、一点破線は、パイプライン処理を導入した場合（Ｐ）、二点破線は、キャツシュメモリとパイプライン処理を併用した場合（Ｃ＋Ｐ）を表している。ＤＦＣでは、ノーマル処理の場合、及びキャッシュメモリを使用した場合は、計算時間を増すごとに少しづつではあるが、スピードアップしている。これは、ＤＦＣの方がＮＣよりも同



（ａ）　ＤＦＣ　　（ｂ）　ＰＣＦＣ

図１９　計算時間による影響

じ計算時間に対して、全体の実行時間が大きいためである。パイプライン処理の場合、計算時間が、６８ｕまでスピードアップし、それ以後は、ダウンしている。これは、式（６．８）より、実行時間は、計算が６８ｕまでは、t_{dp} ＝ＤＡＰＦＴとなり、計算時間に関係しないので、ＮＣに比べるとスピードアップする。しかし、６８ｕ以後は、t_{dp} ＝ＤＣＴとなり、実行時間が計算時間に等くなるので、スピードダウンするためである。パイプライン処理と、キャッシュメモリ使用も同様に、式（６．８）より約４０ｕまでは、ＤＡＰＦＴが実行時間になり、それ以後は、ＤＣＴが実行時間になっている。次に、ＰＣＦＣでは、グラフの外観は、ＤＦＣとほぼ同じであるが、基本的にＤＦＣの方が速いので、グラフ全体として、ＤＦＣの方が高いスピードアップを保っている。パイプライン処理の場合、及びパイプライン処理＋キャッシュメモリ使用の場合は、ＤＦＣで述べた理由と同じで、式（６．９）より、それぞれ９８ｕ、５０ｕが最高のスピードアップになっている。

８．　むすび

以上、ＤＦＣ、ＰＣＦＣのＮＣに対するハードウェアの性能比較を行った。その結果、一般には、ＤＦＣ、ＰＣＦＣのスピードアップは、ＰＵの台数に比例して増加し、ＤＦＣでは、ＰＵの台数の約６３％のスピードアップになり、ＰＣＦＣでは、約４０％のスピードアップになることがわかった。また、キャッシュメモリとパイプライン処理の効果は、ＤＦＣ、ＰＣＦＣどちらも、パイプライン処理の効果がキャッシュメモリの効果より大きく、また、その効果は、ＰＣＦＣの方がＤＦＣよりも大きく表われるという結果となった。その他、ネットワークの遅延の影響、連想時間の影響、レジスタの影響等についても研究されている(14)。

文献

（１）　Dennis, J.B. and Misunas, D.P.,"A preliminary architecture for a basic data flow processor," IEEE Proc. 2nd Ann. Symp. Computer Architectures,pp. 126-132, 1975.

（２）　Davis,A.L.,"The architecture of DDM1:A recursively structured data driven machine," " Dept. of Computer Sci.,Univ. Utah, Tech. Rep. UUCS-77-113, Oct. 1977.

（３）　Arvind and Gostelow, K.P.,"Data flow computer architecture; Research and goals, " Dept. Inform. and Comput. Sci., Univ. California,Irvine,tech.Rep.113, Feb.1978.

（４）　Plas, A.D., Comte, D., Gelly, O. and Syre, C.,"LAU system architecture:Parallel data-driven processor based single assignments, " IEE Proc. 1976 Int'l.Conf.Parallel Processing, Aug. 1976.

（５）　Sowa,M.," Real time multi- microprocessor system introduced concept of data driven, " Proc. of the IFAC/IFIP Workshop, August 1981.

（６）　Sowa, M. and Murata, T.,"A data flow computer architecture with program and token memories," IEEE Trans. on Computers, Vol. C-31, pp. 820-824,Sep. 1982.

（７）　Sowa, M.,"Control flow parallel computer architecture," 情報処理学会計算機アーキテクチャ研資、４８－２、昭和５８年３月.

（８）　曾和、上村、“試作データフローコンピュータのトークンメモリの構造と連想処理、”信学会、電子計算機研究会、ＥＣ８３－１７、昭和５８年７月.

（９）　曽和、”データフロー、コントロールフローパラレル、ノイマンコンピュータのハードウェア性能比較”、情報処理学会、計算機システムの制御と評価の発表会，２２－３，昭和５９年３月.

（10）　Chang, D.Y.,Kuck, D.J., and Lawrice, D.H., " On the effective bandwith of parallel memories," IEEE Trans. on Computers, Vol.C-26, No.5, May 1977.

（11）　Treleaven, P.C. Brownbrige, D.R., and Hopkins, R.P.,"Data-drivenand demand-driven computer architecture," ACM Computer Surveys, Vol.14,No.1, March 1982.

（12）　Chi-Yuan Chin and Kai Hwang " Connection Principles for Multipath packet switching networks,"IEEE.1943.

（13）　山田、” コンピューター・アーキテクチャ”、コンピューター・サイエンス・ラボラトリー、PP123-131、昭和５２年３月.

（14）　渡部、” データフロー、パラレルコントロールフロー、ノイマンコンピュータのハードウェア性能比較” 群馬大学工学部情報工学科内部報告、ＳＬ－Ｌ－８５００２１、昭和５９年９月.