

プリフェッチを主体とするページングアルゴリズム† (PDP 法) の作成と評価

近 藤 正 人^{††} 山 口 純 一^{†††} 近 藤 留 美 子^{††††}

仮想記憶方式計算機の効率向上を目的とした新しいページング方式 PDP 法を提案する。本方式はプログラム実行状況のデータを活用したプリフェッチを実現していることを特徴とし、プリコンパイラ部とページ管理部から成る。プリコンパイラ部は、ページ管理部に情報を伝達する処理を行う命令（通信命令）を、ソースプログラム中に挿入する役目を持つ。ページ管理部はプログラムからの情報を得てプリフェッチを行う部分と、情報の得られない場合に WS 法によるページ管理を行う部分から構成される。

FACOM M160 の OSIV/F4 のもとに PDP 法の実験システムを実現した。実験システムで収集したデータと人工的に作成したデータを組み合わせてシミュレーションにより性能を評価し、WS 法との比較検討を行った結果、PDP 法は WS 法と比べ全般的に効率が優るが繰り返し参照型フェーズで参照ページが順次変化を行う場合には効率が悪化することが明らかとなった。

本論文は実験システムの構成とシミュレーションによる評価結果について述べ、本方式の有効性を示す。

1. ま え が き

仮想記憶方式の普及は、我々に多大な恩恵をもたらした。計算機利用者は実メモリ量を意識せずプログラムを作成できるようになった。また、実メモリを効率的に使用できるので、計算機管理者は実行プログラム多重度の向上が図れるようになった。しかし、仮想記憶方式の導入により新たな問題点が提起されたのも事実である。利用者にとってはプログラミングの如何により経過時間が大きく異なる事態が生じたこと、管理者にとってはスラッシングが、その代表的な問題である。これらを解決し、仮想記憶方式をより効率的に運用するため、多くの研究が行われてきているが、それらは次の3つの方向に大別される。

1) 仮想記憶を意識したプログラム作成法

プログラムで使用するアルゴリズムやコーディングを改善することにより、実行する際に実メモリ上に存在しなければならないページ数（これをプログラムの実効的サイズと呼ぶ）を小さくすることをめざしている^{1)~5)}。

2) プログラムの再構成

Hatfield, Gerald⁶⁾ に代表され、その後 Ferrari⁷⁾ や

益田⁸⁾ が発展させている。この方法は、プログラム実行時のページ参照状況のデータを基に、参照される時期が比較的近いプログラム単位を集めて同一ページに割り付けるものである。これによりプログラムの実行的サイズを小さくし、効率よい実行を図ることを目的としている。

3) ページングアルゴリズムの研究

ページングアルゴリズムは、i) fetch policy（必要なページを、どの時点で二次記憶から実メモリへ移動させるかを定める方針）と ii) replacement policy（どのページが不必要であるかを定める方針）から成る。fetch policy としては、実際に必要となった時点でページを実メモリに持ってくるデマンドフェッチ方式と将来必要であろうと予測されるページをあらかじめ実メモリ上に確保しておくプリフェッチ方式の2つがあるが、現在では参照ページの予測が困難と思われるためデマンド方式が主体となっている。replacement policy にはプログラム動作モデルの違いにより各種の方式があるが、その代表的なものに WS (Working Set) 法、LRU (Least Recently Used) 法等がある。さらに最近フェーズの概念を導入した DWS⁹⁾ (Damped Working Set) 法が提案されている。

これら3つの方向で、1) は利用者が積極的に仮想記憶方式とかわり、各自のプログラムをより効率的なものに改善する方法である。個々のプログラムにとって効果は期待できるが、確実かつ根本的な解決はむずかしい。2) はプログラム実行時のデータをあらかじめ収集し、プログラムの特性をとらえたページ割り

† Design and Experiment of New paging algorithm based on Prefetch-policy by MASATO KONDO, JUNICHI YAMAGUCHI (Department of computer Science, Tokyo Institute of Technology) and RUMIKO KONDO (Department of Physical Electronics, Tokyo Institute of Technology).

†† 東京工芸大学工学部電子工学科

††† 東京工業大学大学院情報工学専攻

†††† 東京工業大学工学部電子物理工学科

付けを行うので、繰り返し利用されるプログラムにとっては有効ではあるが、一般的な活用には問題が残る。3) は仮想記憶方式の根本的な問題にかかわる研究であるが、現在の方法では fetch policy にデマンド方式が採用されているため、実行プログラムのフェーズ遷移等に対し有効に対応できていない。以上の考察から、仮想記憶方式の効率向上をはかるには、新しいページング方式の開発が必要であると考え、そしてそれは、fetch policy と replacement policy に十分にプログラムの動作情報が反映できるものでなければならない。

そこで、新しいページングアルゴリズム Program flow Driven Paging 法 (PDP 法と略す) を提案する。PDP 法の特徴は、実行プログラムの全体像を把握しながら状況に応じたページングを行うことである。これはプログラムがページ管理部に情報を提供できるように、ソースプログラム中に命令を加えることにより実現されるが、この操作はプリコンパイラによって行われる。ページ管理部は、プログラムからの情報を受けて、プリフェッチを主体としたページングを行う。

本論文は、PDP 法を実現した実験システムの構成および従来のページング方式との比較検討を行った結果について述べる。

2. 実験システムの概要

本システムは、プリコンパイラ部とページ管理部から成り、FACOM M-160 上に実現されている。本計算機は、メモリ 2 キロバイトごとに参照ビット (R ビット) と変更ビット (C ビット) が備わっている。当該メモリの参照があった場合は R ビットが、変更が行われた場合は R ビットと C ビットが自動的にセットされる。また、ページ割込みはプログラム割込みの一種として処理される。

制御プログラムは OS IV/F4 を用いている。これには、登録されているプログラムだけにシステムの特定の機能や資源の使用を許可する認定プログラムという制度があり、ユーザ独自のシステム開発に便宜をはかっている。ページ管理部の作成にあたりこの制度を利用しているので、一般ユーザが使用できない PAGE-FIX, PAGELOAD 等、仮想記憶管理用の特殊なマクロ命令を使用したり、プログラムの状態をユーザモードからスーパーバイザモードへ切り換え、RRB (Reset Reference Bit) 命令や LRA (Load Real Address)

命令等のシステム制御命令を用いた処理が可能となっている。

2.1 プリコンパイラ部の構成

本実験システムは、対象とする言語として FORTRAN を採り上げ、FORTRAN のプリコンパイラを作成した¹⁰⁾。その役目は、一般の FORTRAN プログラムがページ管理部に情報を伝えることができるようにするため、あらかじめソースプログラムを変換することである。実際には、実行時のページ参照状況をページ管理部へ伝える処理を行ういくつかのサブルーチン (これを通信命令群と呼ぶ) を用意しておき、これら呼び出す命令を一定の規則に従って挿入していく方法をとっている。

本節では通信命令を挿入する場所をどのように選定すべきかの議論をした上で、通信命令の種類と機能、およびソースプログラムの変換規則と通信命令の挿入規則について述べる。

2.1.1 プログラムの局所性

仮想記憶上でのプログラム動作はページ参照列で表現できる。このページ参照列には、i) いくつかのフェーズが存在し、ii) フェーズの遷移は短時間に行われ、iii) 異なるフェーズでは参照するページ群が異なる、等の性質があることが知られている。それでは、フェーズの遷移はプログラム中のどのような実行過程により引き起こされるのであろうか。

一般にプログラムは主プログラムといくつかの副プログラムから構成されている。それらのルーチンではそれぞれ命令が存在するアドレスや一時的に使用するデータ領域が異なる。したがってルーチン間の制御の移行はフェーズ遷移とかわかりがあることが予想される。また、プログラムには多くの繰り返し処理が含まれる。繰り返し部はある特定のデータ群に対しての処理を行う場合が多く、異なる繰り返し部では使用されるデータ部の違いから参照されるページ群が異なる可能性を持つと考えられる。したがって、参照ページ群を変化させる可能性を持つ実行命令として、i) CALL 文・関数引用文、ii) RETURN 文、iii) DO 文、iv) GOTO 文、v) IF 文等が考えられる。図 1 は実際のプログラムの i) ~ v) の個所に通信命令を挿入し実行して得られたものである。このプログラムは、主ルーチンから順々にサブルーチン SUB1, SUB2, SUB3, SUB4 が呼ばれており、SUB4 からさらに SUB41, SUB42, SUB43, SUB44 が呼ばれている。また、SUB41 と SUB43 には DO ループが形成され

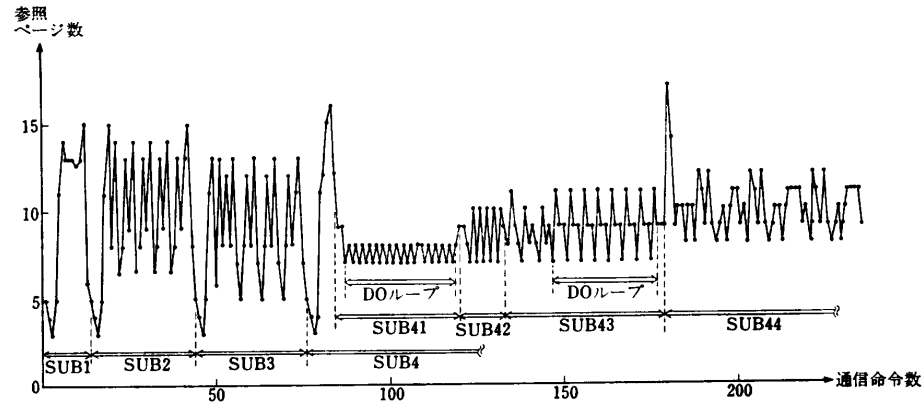


図 1 プログラムの動作状況

Fig. 1 Program behavior and referenced pages.

ている。横軸は時間の推移を表わすもので、通信命令の実行回数を単位としている。縦軸にはその時点での参照ページ数を表わす。これは、通信命令実行のたびにページごとにある R ビットを調べ、R ビットが on であれば off にする一方その数を求めたものである。ルーチン間で制御が移行する時にページ数の変化が生じていることがわかる。また、DO ループの実行時には周期的な変化が現われている。したがって、先に述べた i) ~ v) が実際の参照ページ群の変化に対応する個所であることが示された。換言すると、参照ページ群に変化が生じたならば、その時点で i) ~ v) のいずれかが実行されたことになる。しかし、逆は必ずしも正しくないことは明らかである。

以上より、プリコンパイラが通信命令を挿入する個所としては、CALL 文、関数引用文、RETURN 文、DO 文、GOTO 文、IF 文の前後を選べばよいという結論を得た。

2.1.2 通信命令の機能

通信命令群は 3 つのサブルーチンからなる。以下にこれらのサブルーチンの種類と機能について述べる。ここで仮引数 N は、プリコンパイラがソースプログラム中に通信命令を挿入していく順番に与える通し番号である。

1) ¥ESCP (N)

分岐が生じる個所、DO ループが形成される個所を示す。この時点におけるページ参照列を情報として提供する。

2) ¥PUSH (N)

副プログラムの引用を示す。副プログラムへ分岐する直前のページ参照列を情報として提供する。

3) ¥POP (N)

副プログラムからの復帰を示す。¥POP は ¥PUSH と一対になっており、対応する ¥PUSH と同じ番号を与える。

2.1.3 通信命令挿入規則

ソースプログラムに通信命令を挿入する際、ソースプログラムをあらかじめ一定の形式に整えておく都合がよい。ソースプログラムは次のように整えられる。

- 1) DO 文の端末文は CONTINUE 文に変える。
- 2) 実行文の文番号は CONTINUE 文だけに付ける。
- 3) 副プログラムを引用する文は、論理 IF 文に含ませない。

1) ~ 3) により整えられたソースプログラムに対し、次の規則で挿入が行われる。

規則 1 文番号のある CONTINUE 文の次に ¥ESCP を呼び出す命令を挿入する。

規則 2 副プログラムを引用する文の前に ¥PUSH を、後には ¥POP を呼び出す命令を挿入する。

規則 3 DO 文の後には ¥ESCP を呼び出す命令を挿入する。

2.2 ページ管理部の構成

今回提案したページング方式は、プログラムの流れを考慮したプリフェッチを主体とし、デマンドフェッチを補助的な手段として用いているという点において従来のページング方式と大きく異なっている。

プログラムの実行に伴うページ参照列には、いくつかのフェーズが現われる。これらのフェーズは互いに独立で大きな相関は持たないと考えられている。した

がってページ参照列だけから次にどのような状態になるかを推定することは困難であり、プリフェッチは有効でないとされていた。そこで、プログラムの流れに関する情報を活用して将来必要となるであろうページ群を決定するアルゴリズムを考案し、プリフェッチを実現させた^{11), 12)}。

ページ管理部は、i) WS 法によるページ管理を行う部分と、ii) プログラムの流れについての情報を得てページングに活用する部分の2つが主な構成要素である。これらの概要を次に示す。

2.2.1 ページ管理部の処理とデータ構造

ページ管理部は、初めて実行する部分についてはまだ情報がないので WS 法によるページ管理を行い、同時にページ参照に関する情報を管理表に格納しておく。再度実行されたなら、すでに管理表に蓄積されている情報をもとに本方式独自のプリフェッチによるページ管理を行い、必要に応じて管理表の内容を更新する。管理表の情報はプログラム中に挿入した通信命令によりもたらされるわけであるが、挿入命令が多すぎるとオーバーヘッドが大きくなるので、実行を一定時間間隔で区切り、その間で実行された挿入命令の中から1つをその区間の代表に選ぶことにする。

したがって、ページ管理部に制御が渡されるのは、i) ページ割込み発生時、ii) 時間割込み発生時、iii) 挿入命令実行時の3つの場合がある。各々の処理で参照、更新される管理表とシステム変数の名前とその役割を次に示す。

1) 管理表

・ MTAB (Main memory control TABLE)

各ページの参照状態を示すカウンタ値を入れる表で、WS 法によるページ管理を行う場合どのページを引き続き主メモリに置くかを決定するために使用される。ページごとに2バイトの領域を必要とする。

・ PTAB (Program flow information TABLE)

プログラム中に挿入した命令からもたらされる情報を格納する表で、命令ごとに10バイトの領域を必要とし、次の要素から構成される。

Eflag (1バイト) 対応する命令の実行状況(未実行、実行された、代表として選ばれた等)の情報を示す。

Pflag (1バイト) プリフェッチが実施されたか否かを示す。

Lpointer (2バイト) 代表として選ばれなかった命令の場合には、その代表の番号を入れる。

Pno (2バイト) ルーチンのネストを示す。当該命令のあるルーチンを呼び出した所の ¥PUSH の番号を入れる。

RP pointer (4バイト) Reference Pattern へのポインタを入れる。

2) システム変数

CEN Current Entry No. 現在の代表命令番号

LEN Last Entry No. 1つ前の代表命令番号

CFG Clock FlaG 定期的にセットされる。

CPN Current Push No. 現在実行しているルーチンのネストのレベルを示す。当ルーチンを呼び出した所の ¥PUSH の番号を入れる。

2.2.2 ページングアルゴリズムの概要

本ページングアルゴリズムの概要を次に述べる。

1) プログラムの実行に伴う情報は、一定時間内に對し1つの代表通信命令によりもたらされる。代表以外の通信命令は PTAB 中の Lpointer で代表通信命令番号を示す。

2) 初めて実行された所か否かを Eflag で判定する。初めて実行された個所ならば、どのようなページ群を必要とするのか不明である。そこで、現在参照しているページ群はそのまま使用されると仮定し、WS 法に基づいたページの管理を行う。これらの操作を行いながら PTAB に情報を蓄積する。その方法は、代表命令実行時にページ群の Rビットをすべて調べる。これで得られる情報は LEN の参照ページに関するものであり、この情報を参照パターン記憶領域に格納し、その番地を LEN の RP pointer にセットする。

3) 過去に実行した個所を再び実行した場合、これから参照されるページ群は RP pointer が指す情報から求めることができる。そこで、その情報に基づき必要なページをプリフェッチし、不必要なページはページアウトする。

4) 副プログラムへ制御が移行すると、参照ページ群が大きく変化する。復帰した場合は副プログラム中で参照していたものに加え、副プログラムを呼び出す直前に参照していたページ群を再び参照すると考えられる。そこで ¥PUSH が実行されたら、現在参照しているページ群情報を記憶領域に格納し、その番地を LEN の RP pointer にセットし、¥PUSH に対応する PTAB のエントリの Lpointer には、その LEN をセットする。¥POP が実行されたら、その Lpointer から LEN 値を復元できるので、これを用いてページ群情報を復元し、それに基づき主記憶を管

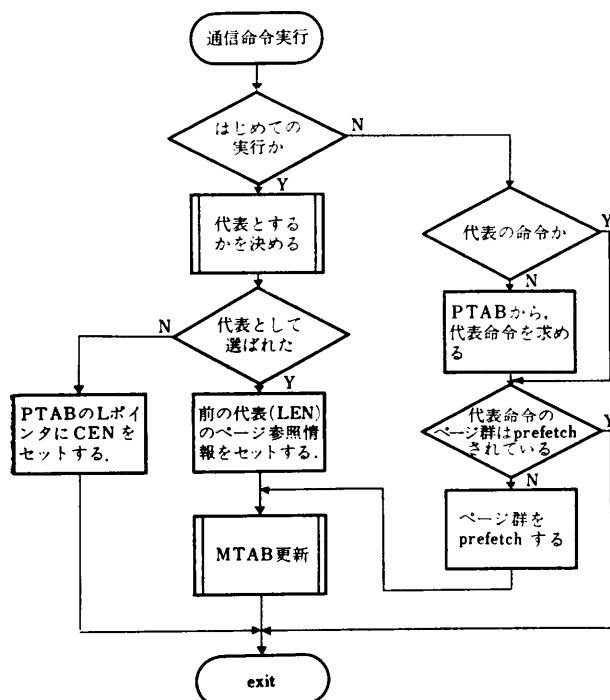


図 2 PDP 法の概要

Fig. 2 General flow of PDP algorithm.

理する。

5) 副プログラムに制御が移行してきたとき、このプログラムが過去に実行されたならば、前回蓄えた参照ページ群をプリフェッチするわけであるが、呼び出した個所が前回と違う場合には、引き渡されるデータ群の違いにより前回の情報は妥当なものではなくなる。そこで、PTAB 中の Pno とシステム変数 CPN との比較を行い、等しければ PTAB に蓄積されている情報は有効と判定し、情報を活用する。もし等しくなければ情報を無効とし、情報がない場合と同等の処理を行う。

処理の概要を図 2 に示す。

次に代表命令選定の方法を述べる。

1) 一定時間間隔で割込みを生じさせ、その都度 CFG をセットする。

2) $\forall$ ESCP が初めて実行された場合、CFG を判定する。CFG がセットされていたらこれを代表命令とし、その表示を Eflag にセットした後 CFG をリセットする。CFG がセットされていなかったならば、CEN 値を Lpointer にセットし、代表命令でない表示を Eflag にセットする。

3) $\forall$ PUSH/ $\forall$ POP は処理が非常に短い場合を除き代表命令とする。

3. 性能の評価

WS 法を比較対象とし、PDP 法の性能を評価する。評価項目としては、平均空間時間積 (STP と略す)、ページ割込み数 (Pin と略す)、平均ページ入出力回数 (PIO と略す) の 3 項目を採り上げた。

STP は主記憶に占めるページ数を実時間で測定したもので、メモリ使用コストを表わしていると考えられる。時刻 t におけるページ量を $R(t)$ 、時刻 t_i においてページ割込みが生じ、その時にページ転送に必要な時間を $T(t_i)$ とすると次のようになる。

$$\begin{aligned} \text{STP} &= \int R(t) dt \\ &= \int R(t') dt' + \sum_{i=0} T(t_i) \times R(t_i) \\ &\cong T_{\text{mean}} \sum R(t_i) \end{aligned}$$

t' : CPU 時間

T_{mean} : 平均転送時間

Pin はページ割込み数であり、プログラムの動作状況を表わすもので、この値が大きいということは実行に必要なページ群が確保されていないことを示し、方式が不合理である証明といえる。PIO は、ページ割込み数 Pin、ページアウト数 Pout、プリフェッチに基づき転送されるページ群 Ppre と表わすと、

$$\text{PIO} = \text{Pin} + \text{Pout} + \text{Ppre}$$

で表わせる。WS 法のようにデマンドフェッチを採用している場合には、Ppre=0 となる。PIO は DASD と主記憶との間で往来するページ数を表わしており、これはチャンネルや DASD の使用率あるいは入出力管理に要するオーバーヘッドに対応する。

シミュレーションにより種々の参照ページ群データに対する両方式の STP, Pin, PIO 値を求め解析を行った。採用したデータおよび解析結果について次に述べる。

3.1 参照ページ群データの作成方法

評価用のデータとしては、実際のプログラムから得たデータ (S と表わす) と種々の変化を与える要素 (N と表わす) を合成した人工的なデータ (X と表わす) を採用した。

入力データの作成手順は次の通りである。

1) 実際のプログラムをいくつか用意し、それらをプリコンパイラに入力して通信命令を挿入する。

2) プログラムを実行し、ページ参照情報を収集する。この情報は通信命令列 I , $I = \{i_1 i_2 \dots\}$ とそれに

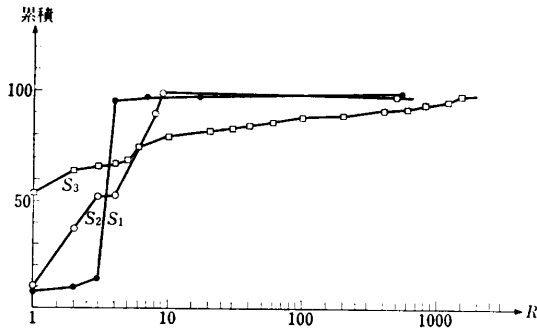


図3 プログラムの特徴

Fig. 3 Behavioral characteristics of three programs.

対応するページ参照情報 P , $P = \{p_1 p_2 \dots\}$ から構成される。今, $i_j = i_k$ ならば p_j と p_k の間には強い相関がある。プログラムから得たデータ S は, $S = (I, P)$ と表わすことができる。

3) ページ数が n で各ページが平均 l 個の代表通信命令を経過する間参照され続けられるようなページ群の列を考える。この特性を $N(n, l)$ で表わす。 N に従うページ列 Q を $Q = \{q_1, q_2, \dots\}$ と表わす。

4) S と N を合成したデータ X , $X = (I, Y) = (I, P+Q)$ をシミュレーションの入力データとする。

S については3種類のプログラムについて情報を収集した。これらのプログラムの特性を表わすために、通信命令列 $I = \{i_1, i_2, \dots\}$ において同一番号の通信命令が再び実行されるまでの間隔 k (何命令めに同一番号の命令が再び表われるかで示す。 $k=1$ は次も同じ通信命令を実行することを意味する) と累積回数の関係を求め図3に示した。プログラム1は処理の中心が繰り返し処理であり、逆にプログラム3は特定間隔での繰り返し処理はなく、順々に処理を進めていくもので、プログラム2は両者の中間に位置する。

情報 N については、 l を1から無限大まで変化した値で n は1~3の値とする。 l は値が大きいほど遷移が生じにくくなり、WS法に有利となる。 l が無限大とは、全然遷移が生じないことを意味する。逆に l が小さいほど遷移が生じやすいことを表わしており、 $l=1$ は必ず遷移が生じることを意味する。

3.2 解析結果の検討

$n=2$ における STP 値の PDP/WS 比を図4に、Pin と PIO 値の PDP/WS 比を図5に示す。横軸は N のパラメータ l を示し、縦軸には WS 法と PDP 法の各項目の比率 (WS 法を1とする) を示す。各項目の結果を解析する。

1) STP

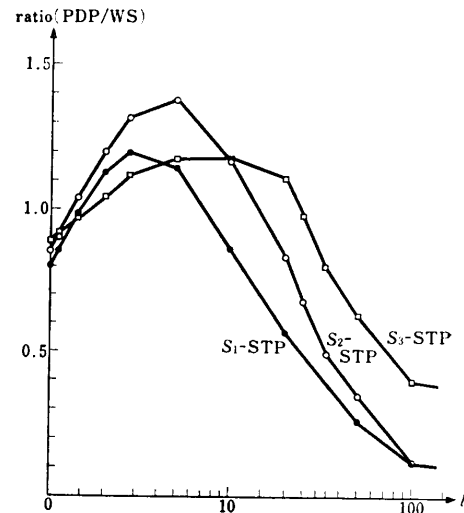
図4 STP 比の変化 ($n=2$)

Fig. 4 Comparison of STP ratio (PDP/WS).

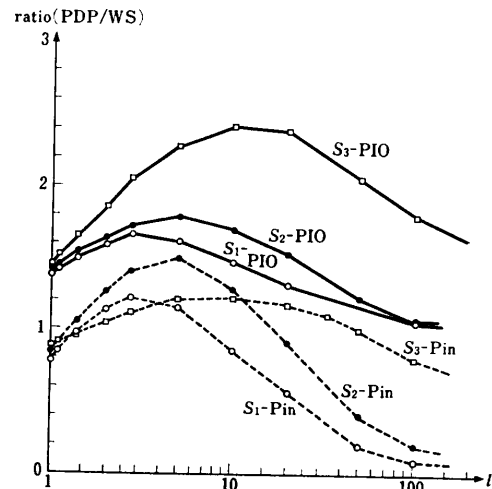
図5 PIO, Pin 比の変化 ($n=2$)

Fig. 5 Comparison of PIO, Pin ratio (PDP/WS).

STP には各方式の総合的な効率が現われている。PDP 法が WS 法に比べ効率よく動作するのは $l \approx 1$ ならびに $l \gg k_m$ (k_m は k の平均値) の区間である。これらの区間がどのようなページ参照状況に対応するかを検討する。まず、 $l \approx 1$ はページ参照期間が短く次々に新しいページが参照されることを示している。つまり、一時的な記憶領域を使用している状況にあてはまる。このような状況がどうして PDP 法に有効かを検討する。通信命令列 $I = \{i_1, i_2, \dots\}$ とページ列 $Q = \{q_1, q_2, \dots\}$ を考えると、 $i_j = i_k$ のとき $q_j \neq q_k$ となり PDP 法のプリフェッチは有効に動作しない。しかし、WS 法でも Q に関する過去の情報はほとんど活用で

きない。ページ列 Q に関しては両者は同じであるが、ページ列 P に関して PDP 法が有利に動作しており、そのため両者の差が生じるわけである。次に $l \equiv k_m$ の場合を考える。これは繰り返し処理において参照ページ群が順次変化する状況にあてはまる。つまり、 Q に関して変化が生じる間隔は大きく、先のように次々と新しいページを参照するのではないが、通信命令について、 $i_j = i_k$ のとき $p_j \neq p_k$ となってしまう状況である。この場合、WS 法が現時点で参照しているページ群はそのまま参照され続けると考え、ページ管理を行っているわけであるから、 l が大きくなればページ列 Q に関しては有効になる。一方 PDP 法は通信命令からの情報をもとに管理を行っているので、 $l \equiv k_m$ ではこれらの情報をもとにプリフェッチを行うと不適切なページを選ぶ確率が高くなってしまふ。したがって WS 法に比して PDP 法は効率が悪化している。次に $l \gg k_m$ の場合になると、通信命令とページ列 Q との間には $i_j = i_k$ のとき、 $q_j = q_k$ が成立する機会が多くなりプリフェッチは有効に働く。この状況は過去の情報が十分に活用できる場合であり、順次変化を行わない繰り返し処理に対応する。

2) Pin

Pin と l の関係は STP と同様の傾向である。 $l \gg k_m$ において Pin は十分に小さな値となる。これは、PDP 法の予測が正確に行われていると、WS 法に比べページ割込みによる処理の中断がはるかに少ないことを示している。つまり、予測がうまくいくとジョブ終了までの経過時間は短縮し、ユーザに利点があると同時にシステムの稼働率も向上できる。

3) PIO

PIO は、Pin, Pout, Ppre から構成されている。Pin は先の 2) で示した値である。WS 法はデマンドフェッチであるので、Ppre=0 である。そのため、PDP 法は WS 法に比べ PIO に関し Ppre の項だけ多くのページ転送を行うことになる。この影響によりページ用 DASD の利用率が高くなるが、必要とするページ群は一括して入力されるので、入力によるジョブ処理中断はあまり大きくならないと考えられる。

ページ参照列にはフェーズが存在し、フェーズ内のページ参照形態には一様参照 (I 型) と繰り返し参照 (II 型) の 2 種類がある。PDP 法は I 型ならびに II 型の一部に対し WS 法より有効であることが示された。II 型参照で順次変化を行う場合には、PDP 法は効率悪化が大きくなる。この点は、プリフェッチの有効性

を検討する機構を組み込むことで対処できる。たとえば、現在主記憶にあるページ群と前回プリフェッチしたページ群 (これは LEN から求められる) とを比較し、違いが大き過ぎればプリフェッチを採用せず WS 法によるページ管理に切り替えるアルゴリズムが考えられる。フェーズ遷移に対しては、それが過去に実行した個所であれば PDP 法が有効に対処する。一方、初めて実行するのであれば、PDP 法は WS 法と同じ処理を行うので両者に差は生じてこない。したがって PDP 法は WS 法と比べ全般的に効率が良い。

4. む す び

新しく開発したページング方式 PDP 法の原理とその実験システムの処理内容について述べてきた。PDP 法はそのページングの方針として、i) プリページングを導入すること、ii) フェーズの概念を反映したページングを行うこと、の二点に重点を置いており、実験システムとして FORTRAN のプリコンパイラとページ管理部を作成した。PDP 方式の性能評価を行い、WS 法に比べフェーズ遷移を意識してページ管理を行っているので STP, Pin の値が優位であるという結論が得られた。

今回は対象言語として FORTRAN を採り上げたが、他言語についても同様のシステムが作成でき、十分効果も期待できる。

参 考 文 献

- 1) Moler Clever B.: Matrix Computations with Fortran and Pagings, CACM, Vol. 15, No. 4 (Apr. 1972).
- 2) Elshoff James L.: Some programming techniques for processing multi-dementional matrices in paging environment, NCC (1974).
- 3) 村田健郎, 堀越清視: 対称帯行列を三重対角化するための新アルゴリズム, 情報処理, Vol. 16, No. 2 (Feb. 1975).
- 4) 村田健郎, 二村良彦, 門間三尚: 仮想記憶方式の下での大形行列計算技法, 情報処理, Vol. 21, No. 4 (Apr. 1980).
- 5) 三橋鎮雄, 庄野英二: SSOPTRAN プリコンパイラ, 情報処理, Vol. 21, No. 4 (Apr. 1980).
- 6) Hatfield, D.J. and Gerald, J.: Program restructuring for virtual memory, IBM system Journal, No. (1971).
- 7) Ferrari, D.: Improving Locality by Critical Working Sets, CACM, Vol. 17, No. 11 (Nov. 1974).

- 8) Masuda, T.: Memory Utilization and Improvement of Program Locality, IEEE transaction on software engineering, Vol. 1 SE-5, No. 6 (Nov. 1979).
 - 9) Smith Alan J.: A Modified Working Set Paging Algorithm, IEEE transaction on computer, Vol. C-25, No. 9 (Sep. 1976).
 - 10) 牛島和夫: Fortran プログラミングツール, 産業図書.
 - 11) Trivedi Kishor S.: Prepaging and Application to Array Algorithms, IEEE transaction on Computer, Vol. C-25, No. 9 (Sep. 1976).
 - 12) Trivedi Kishor S.: On the Paging Performance of Array Algorithms, IEEE transaction on Computer, Vol. C-26, No. 10 (Oct. 1977).
(昭和 56 年 7 月 10 日受付)
(昭和 56 年 11 月 18 日採録)
-