プログラミング 18−19
(1998. 3. 23)

ページのプリフェッチングにおける動的調整機構

小野 貴寛, 大澤 範高, 弓場 敏嗣
電気通信大学大学院情報システム学研究科
〒182-8585 東京都調布市調布ヶ丘 1-5-1
E-mail: ono-t@yuba.is.uec.ac.jp

あらまし　仮想記憶システムのようなページキャッシュシステムにおいて、システムとアプリケーションプログラムが協調的に動作できるようにし、性能向上を図ることが望ましい。このためには、システムがアプリケーションにメモリ領域の使用状況などのシステム情報を与え、アプリケーションがシステム情報に応じてページ置換えのヒントやプリフェッチ等の注釈をシステムに与えることが重要になる。アプリケーションは、システム情報を参照することにより、より適切な注釈を与えられるようになる。システム情報や注釈を、アプリケーション・システム間で交換するための機構とそれを用いた性能向上のための動的調整機構を提案し、基本的な評価を示す。

キーワード　仮想記憶、ページキャッシュ、注釈、プリフェッチ

A Machanism for Dynamic Adjustment of Page Prefetching

Takahiro ONO, Noritaka OSAWA, Toshitsugu YUBA
Graduate School of Information Systems
The University of Electro-Communications
Chofugaoka 1-5-1, Chofu, Tokyo 182-8585, JAPAN
E-mail: ono-t@yuba.is.uec.ac.jp

Abstract　In a page cache system like a virtual memory system, it is desirable for the system and application programs to work cooperatively and to enhance performance. In order to improve performance cooperatively, it is important for the system and the applications to exchange information. Application programs can dynamically give the system appropriate annotations by utilizing the system status information such as memory usage statistics, and then the system can manage pages better based on appropriate annotations like hints for page replacement and prefetching. This paper proposes a dynamic annotation mechanism for performance improvement using the system information. and shows a preliminary evaluation of the proposed mechanism.

key words　virtual memory, page cache, annotation, prefetch

1 はじめに

現在利用されている仮想記憶システムやファイルシステム、分散共有記憶システムなどのページキャッシュシステムでは、Least Recently Used (LRU) もしくはそれに類似したアルゴリズムが主に使われている。しかし、固定的な置換えアルゴリズムではすべてのアプリケーションには対応できず、十分な性能を発揮させることができない場合がある。アプリケーションに性能を発揮させるためには、ページ管理をアプリケーション毎に適応させることが重要である。アプリケーションへの適応のためにユーザタスク毎にサーバを用意し、そのサーバが置き換え方式を決定できるシステムがある。これによって自由にポリシを選ぶことができ、メモリ領域間の関係も組込むことができる。しかし、サーバのバグなどによってシステム全体が影響を受けることがある。これらとは異なった性能向上の方法に、アプリケーションプログラムがキャッシュシステムに注釈を与える方法がある。

ここで、注釈は、アプリケーションプログラムが性能向上のために行うカーネルへの指示であり、不適切に付与してもプログラムの誤動作を引き起こさない指示である。これまでに、ページ操作に対する低オーバヘッド注釈機構の提案と実装を行った [7]。

しかし、従来の注釈では、システムの性能に応じて注釈を適切に与えなければ十分な効果を得ることができない。そこで、カーネルがシステム情報を与え、ユーザタスクがそれに基づいて注釈の付与を調整するような処理を行うことを考える。本稿では、注釈機能とカーネルからの情報提供機能を利用した性能向上のための動的調整機構の提案及び評価を行い、考察する。

2 ページのプリフェッチングにおける動的調整機構

この章では、ページのプリフェッチングにおける動的調整機構について説明する。

2.1 システム情報を利用した動的調整機構のモデル

システム情報を利用した動的調整機構のモデルが図 1である。ユーザタスクは注釈を付与することで、カーネルに対してページング方式を指示することができる。一方、カーネルはユーザタスクから指示されたこれらの注釈を参考にページ管理を行うが、それと共にユーザタスクに対してシステム情報を与える。注釈を用いたページキャッシュ機構の実現 [7] と同様に、システム情報の取得にはメモリアクセスを利用することでオーバヘッドを削減する。ユーザタスクは、プログラム実行中に動的にカーネルからシステム情報を得て、その情報に基づいて注釈付与の調整を行う。単にカーネルに対して注釈を付与するのではなく、システム性能を反映した注釈付与調整を行うことでより効果的なページ管理が実現できる。

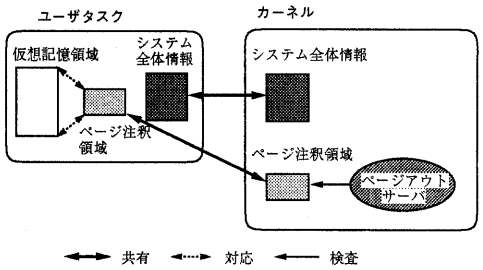

図 1: ユーザタスクとタスクとカーネル間の関係図

2.2 プリフェッチ注釈による時間短縮効果

ユーザタスクはシステム情報を利用して適切に注釈を発行することによって、効果的なページキャッシングを行うことができ、実行時間は短縮される。プリフェッチ注釈の調整による効果は、ページフェッチ (ディスク入出力) 時間に現れる。例えば、シーケンシャルアクセスを行うプログラムの場合、次にアクセスされる領域 (ページ) があらかじめ予測できるので、処理の速度 (計算時間) に合わせてタイミングよくプリフェッチを行わせることでページフォルト数を減少させ、ページフェッチ時間を隠蔽することができる。

図 2では、ページアクセスを行ったときに、対象ページがメモリ上に存在しない場合を示している。このとき、CPU によるプログラムの処理はページフェッチ処理により、対象ページをメモリに取り込むまで待たされる。この間、CPU の処理とページフェッチ処理は並行に動作することができず効率が悪い。

一方、図 3では、事前にプリフェッチを行い、これからアクセスする対象ページをメモリ内に読み込んでおくために、ページアクセスされる際には直ちに処理が行われる。またそれと同時に次にアクセスされるページをプリフェッチすることができる。これにより CPU の処理とディスク入出力処理は並行に行われ、CPU の処理はページフェッチ処理に待たされることなく行われる。

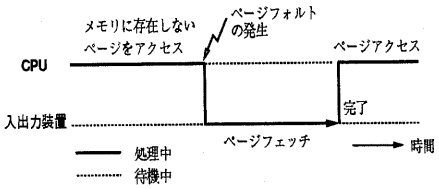

図 2: ページアクセス処理とページフェッチ処理が並行に行われなかった場合

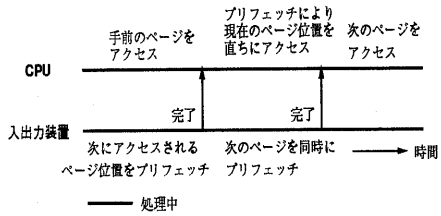

図 3: ページアクセス処理とページフェッチ処理が並行に行われた場合

2.3 プリフェッチ注釈による動的調整の効果

アプリケーションプログラムは、注釈を用いることで、カーネルに対してページ管理を指示できる。しかし指示される注釈が必ずしも適切でなかったり、また注釈の数が多すぎるとシステム全体として余計なオーバヘッドになってしまう。そこでアプリケーションプログラム側から一方的に指示するだけでなく、カーネル側からシステム情報を受け、その情報に基づいて適切な注釈を発行できるように調整を行うことが必要である。以下にページキャッシュの例としてプリフェッチ注釈を用いた場合の動的調整の必要性を示す。

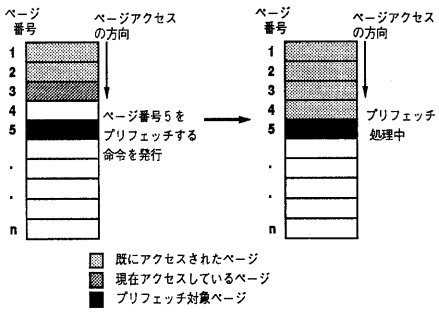

図 4: プリフェッチ注釈プログラムの例

図 4に、与えられた領域全体に対して、シーケンシャルアクセスを行うプログラムのアクセスパタンを示す。この例ではあるページをアクセスした際に、同時に 2 ページ先のページをプリフェッチする処理を行う。しかしプリフェッチする際に、この 2 ページ先が適当であるか否かはシステムの負荷状態や性能によって異なる。システム負荷が大きい時には、2 ページ先のページをプリフェッチすることは有効かもしれないが、負荷が小さい時にはプリフェッチにより 2 ページ先のページをメモリに読み込む前にそのページがアクセスされてしまうかもしれない (図 4 の右側)。結果として、ページフォルトが生じ、プリフェッチの効果がなくなる。このような場合には、プリフェッチすべき時期を早くする (さらに先のページを取得する) ようにすればよい。

プリフェッチすべき時期を早くすることで、必要なページがアクセスされるときにメモリに存在する割合は大きくなるが、プリフェッチ時期は必ずしも大きくすればよいとは限らない。アクセスされる領域全体の大きさが、ユーザタスクが用いることのできる物理メモリ量を越えていた場合、あまり先のページをプリフェッチするとすべてのページをメモリに置くことができないために、アクセスされる前にメモリから追い出されてしまう場合がある。したがってシステム性能や実行するプログラムにおける割り当てられるメモリ領域の大きさなどを考慮して、プリフェッチ距離を決定する必要がある。

2.4 動的調整の手順

この節では、カーネルとユーザタスク間でページのプリフェッチングにおける動的調整を行う手順について述べる。ここでの動的調整の手順は、シーケンシャルメモリアクセスプログラムの場合を例とする。動的調整は、プログラム実行前に決定する事項とプログラム実行時に決定する事項がある。

プログラム実行前に決定する事項として、実行するアプリケーションプログラムがどのような処理を行い、どのメモリ領域をアクセスするかを解析する。実行するアプリケーションプログラムにより、利用するメモリ領域の大きさやアクセスパタン、またアクセスするタイミングが異なるためにこれらの解析が必要になる。ここで解析して得られた情報は、次の実行時間予測式を決定する際に用いる。

解析して得られた情報に基づき、アプリケーションプログラムの実行時間予測式を決定する。シーケンシャルメモリアクセスプログラムの場合、実行時間は、ページ当たりの処理時間と処理されるページ数との積で表されるが、ページ当たりの処理時間はアプリケーションプログラムによって異なるので、前述の解析に基づき決定する。さらにアクセスされるページがメモリ上に存在していなければ、

ページをディスクからメモリに取り込む時間がこれに加算される。このページフェッチ時間は、プリフェッチを効果的に行い、ページ処理時間と並行に動作させることで隠蔽することができる。したがって実行時間予測式は、注釈の与え方に伴う実行時間の変化を表した式になる。この予測式中のパラメータの中には、前述の解析に基づき、システム情報から動的に得られるページ当たりの処理時間やページフェッチ時間などが含まれている。

次にプログラム実行時に決定する事項について説明する。ここでは、実行時間予測式中のパラメータを決定するために必要なシステム情報を取得する。システム情報は、アプリケーションプログラム実行中に動的に取得される情報であり、ハードウェア的、ソフトウェア的な環境によって異なる。この情報を用いることで、システム性能を反映した注釈付与が可能である。必要なシステム情報は、アプリケーションプログラムの処理によって、また用いる注釈の種類によって異なる。プリフェッチ注釈を用いたシーケンシャルメモリアクセスプログラムの場合、N ページ当たりの実行時間及びページフォルト回数などがシステム情報として必要である。これらの情報から実行時間予測式中のパラメータである 1 ページ当たりの処理時間や 1 ページ当たりのページフェッチ時間を計算する。

最後に、システム情報として取得されたパラメータを実行時間予測式に当てはめる。この実行時間予測式の時間値を最小にするように注釈の与え方を調整する。具体的には、取得されたパラメータを実行時間予測式に代入して、その最小値を与えるプリフェッチ距離 (時期) を求める。そして決定された注釈を付与することで、効果的なページキャッシングを実現する。これらの調整を一定周期ごとに行うことによって、システム負荷を適切に反映した注釈を与えることができる。

3 ページのプリフェッチングにおける動的調整機構の性能評価

これまでにページのプリフェッチングにおける動的調整機構について述べてきた。本章では提案した動的調整機構を実装し、プログラムとして動作させることでその性能評価及び考察を行う。本評価では、実機で動作させ計測した値を用いた。

本評価で用いた環境は、以下の通りである。

<使用するハードウェア>

- 機種: IBM-PC/AT 互換機 (Gateway 2000 P5-90),
  CPU: i586-90MHz, 物理メモリ: 16MB,
  SCSI インタフェース: Adaptec AHA-1542CF,
  SCSI ディスク 1GB

<使用するソフトウェア>

- OS: Mach4 マイクロカーネル (version UK22)[1]
  注釈やシステム情報は、カーネルの仮想記憶の部分を変更して実現した。
- Emulator: 4.4BSD Lites Server(version 1.1.u3)[2]

4 評価プログラム

単純なシーケンシャルメモリアクセスパタンを持つプログラムに対して、ページのプリフェッチングにおける動的調整を行う。

4.1 メモリアクセスモデル

ここでは評価に用いるプログラムの説明を行う。図 5にプログラムに割り当てられるメモリ領域の構成を示す。プログラムではユーザタスクに 4MB の連続領域が割り当てられる。Mach で扱われるデータ領域の単位は、ページサイズ (4096bytes) である。したがって領域全体は、1000 ページである。この 4MB の領域に対してページ単位でシーケンシャルアクセスを行う。そしてプリフェッチ距離 (時期) を調整していくことによって、実行時間の変化を調べる。

まず指示されたページに対してプリフェッチを行い、その後ページをアクセスする。例えば、プリフェッチ距離を 2 ページ先として指定すれば、最初に現在のページ位置の 2 ページ先をプリフェッチし、その後、現在のページ位置をアクセスする。アクセスされるページが、メモリに存在していれば直ちにアクセスが行われるが、メモリに存在しなければページフォルトが発生し、ディスクからページがフェッチされる。

ページフォルトが発生すると、ページがメモリに取り込まれるまでメモリアクセスの処理は待たされる。CPU の処理 (プログラムでいうところのページアクセスに相当する) とページフェッチ (ディスク入出力) 処理は並行に行うことが可能である。したがって、プリフェッチが適切に行われていれば、ページアクセスの際には常にメモリ上に必要なページが存在し直ちにアクセスできる。ページフォルト数が減少すれば、実行時間もそれに応じて短縮する。

4.2 モデルに基づいた実行時間予測式

プリフェッチが行われた場合の実行時間予測式は、式 (1) である。実行時間は、1 ページ当たりの処理時間 L と、ページアクセス処理とページフェッチ処理が並行に行われるとき、ページ処理時間で隠蔽できなかったページフェッチ時間 $S(d, L)$ との和で表される。これにプリフェッチシステムコール時間 prs とプリフェッチ距離 (時期) によるオーバヘッド時間 $V(d)$ が加えられる。プリフェッチを指示する際に、同じ 1 ページをプリフェッチしてもプリフェッ

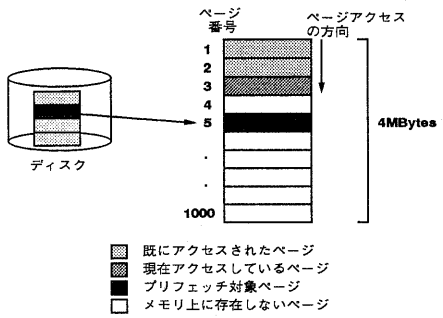

図 5: ユーザタスクに割り当てられるメモリ領域の構成について

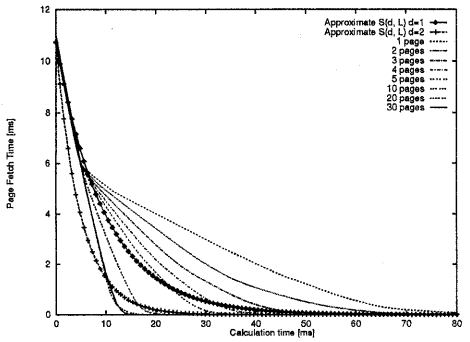

図 6: ページ処理時間によるページフェッチ時間の隠蔽の度合と近似した関数

チ距離によってページフェッチ時間が変化する。具体的には、プリフェッチ距離が大きくなる程、ページフェッチ時間も増加する。これはカーネルのページ管理法に起因すると考えられる。

これが 1 ページ当たりの実行時間である。この 1 ページ当たりの時間の総ページ数分の和が全体の実行時間になる。

実行時間予測式

$T(d,L)$: 総実行時間
L: 1 ページ当たりの処理時間
N: アクセスする総ページ数
I: ページフェッチ時間の平均値
Prs: プリフェッチシステムコール発行時間
d: プリフェッチするページまでの距離
$S(d,L)$: ページフェッチ時間の関数
$V(d)$: プリフェッチ距離によるページフェッチ時間のオーバヘッドの関数

$$T(d,L) = N * (prs + L + S(d,L) + V(d)) \quad (1)$$

$$V(d) = a * d(a \text{ は定数}) \quad (2)$$

式 (1) 中の $S(d,L)$ は、プリフェッチが行われページアクセス処理とページフェッチ処理が並行に行われる際に、ページ処理時間で隠蔽できなかったページフェッチ時間の関数を意味する。ページフェッチ時間は分散が大きいために統計的に処理する必要がある。以下に関数 $S(d,L)$ の近似方法について述べる。

プリフェッチが行われ、アクセスするページがメモリに存在していれば、直ちにアクセスが行われる。このアクセス処理は、プリフェッチ処理と並行に行われるので、ページフェッチ時間をページ処理時間で隠蔽できる。ページフェッチ時間をどの程度隠蔽できるかは、システム性能やページ処理時間に依存する。この並行処理を効果的に行うために、プリフェッチを効率よく行い、プリフェッチ距離 (時期) を適切に調整することが必要になる。

したがってプリフェッチの効果が大きい程、ページフェッチ時間とページ処理時間の並行の度合も増加し、結果として実行時間が短縮される。ここで $S(d,L)$ をプリフェッチ距離とページ処理時間によるページフェッチ時間隠蔽の度合を示す関数とする。

ページ処理時間で隠蔽できないページフェッチ時間を近似する際には、以下の特徴を持つことに注目して行う。

- 単調減少
- 非負値

これらの条件を満たす関数を用いて近似を行った。実際に用いた関数を、式 (3) に示す。プログラムを動作させる環境で動的に、ページフェッチ時間 I とページ処理時間 L をシステム情報から計算し代入することで、近似関数 $S(d,L)$ は具体的に定まる。

< 実行時間予測式中で用いた近似関数 >

I : 1 ページ当たりのページフェッチ時間
L : 1 ページ当たりの処理時間
d : プリフェッチ距離

$$S(d,L) = I * e^{-bdL}(b \text{ は定数}) \quad (3)$$

図 6にページ処理時間で隠蔽できなかったページフェッチ時間のグラフと近似した関数によるグラフを示す。グラフ中の pages はプリフェッチ距離 d による変化に対応している。同じページ処理時間において、プリフェッチ距離が 1 ページ先と 2 ページ先とでは、1 ページ当たりのページフェッチ時間を隠蔽できる割合が異なる。例えば 1 ペー

ジの処理時間が30msのときに1ページ先をプリフェッチした場合、ページ処理時間で隠蔽できないページフェッチ時間は約4msであるが、2ページ先をプリフェッチした場合ではページ処理時間で隠蔽できないページフェッチ時間は2ms程度である。つまりプリフェッチを行うページ先を大きくすることで、ページフェッチ時間を隠蔽できる程度が増加しているのがわかる。

一方、近似した関数 Approximate S(d, L) は、プリフェッチ距離がそれぞれ1ページ先、2ページ先の場合を示している。

4.3 実行時間予測式中のパラメータ決定

ここでは決定した実行時間予測式中で用いるパラメータについて述べる。

式(1)において、システム情報として動的にカーネルを利用して得るパラメータには以下のものがある。

- 1ページ当たりの処理時間：L
- 1ページ当たりのページフェッチ時間：I
- プリフェッチシステムコール発行時間：prs

これらのシステム情報は、ページをアクセスしつつ動的に測定する。式(4)から式(6)は、1ページ当たりの処理時間L及び1ページ当たりのページフェッチ時間Iの計算方法である。

パラメータ取得のための計算方法

＜カーネルから与えられるシステム情報＞
t_1, t_2: Nページ当たりの実行時間
pf_1, pf_2: Nページ当たりのページフォルト数

＜実行時間予測式中のパラメータ＞
I: 1ページ当たりのページフェッチ時間
L: 1ページ当たりの処理時間

$$t_1 = pf_1 * I + L * N \quad (4)$$

$$t_2 = pf_2 * I + L * N \quad (5)$$

(4)と(5)より

$$I = \frac{t_2 - t_1}{pf_2 - pf_1},\ L = \frac{t_1 - pf_1 * I}{N} \quad (6)$$

またプリフェッチシステムコール発行時間prsは、プログラム実行時に動的に計測した値を用いる。

4.4 実行時間予測式と実測値の比較

$$T(d, L) = N * (prs + L + I * e^{-bdL} + a * d) \quad (7)$$

$$d = \frac{log\frac{b*I*L}{a}}{bL} (a、b \text{ は定数}) \quad (8)$$

次に近似した関数$S(d, L)$を用いた実行時間予測式から、実行時間が最小になるプリフェッチ距離を計算する。具体的な計算方法は、式(3)を式(1)に代入して(式(7))プリフェッチ距離dによる関数とし、最小値を与えるdを求めることによりプリフェッチ距離を決定した。最小値を与えるdの式を(8)に示す。計算で求められたプリフェッチ距離は4ページであった。この調整したプリフェッチ距離を用いて注釈を発行した際の実行時間を図7に示す。

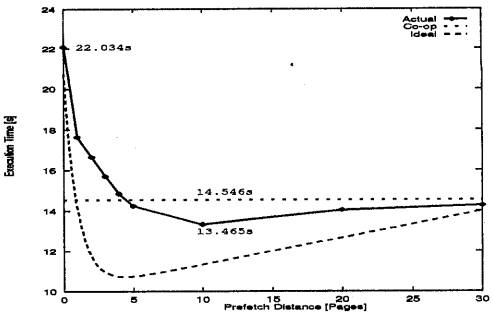

図7: 実行時間予測式により調整したプリフェッチ距離を用いた場合の総実行時間

図7の実線で表されている部分Actualが、実測でのプリフェッチ距離(時期)を変化させた際の実行時間、Idealは実行時間予測式によるものである。直線Co-opはモデルに基づいて動的調整を行った場合の実行時間である。この直線は、調整により決定したプリフェッチ距離(時期)が一定の4ページ先でプリフェッチを行った場合の実行時間を示している。

＜シーケンシャルメモリアクセスプログラムにおける動的調整の効果＞

- プリフェッチを行わない場合に対して34.0%の時間短縮効果がある。
- 静的に調整したプリフェッチによる理想的な時間短縮効果に対して87.9%近づけた。

5 考察

この章では、これまでに示した動的調整機構とモデル化について考察を行う。

5.1 アプリケーションプログラムとそのメモリアクセスパタンについて

動的調整機構では、事前に決定した実行時間予測モデルを利用してカーネルから与えられたシステム情報を基に注釈の調整を行う。実行時間予測式は、実行するアプリケーションプログラムのアクセスパタンを解析することで得られる。アプリケーションプログラムのアクセスパタンは、プログラムの種類によって異なる。アクセスパタンがあまり複雑になると適切な予測ができず、プリフェッチの効果が望めない。実行時のパラメータによってアクセスパタンが大きく異なるようなものは予測が難しくなる。つまり注釈を与えてもうまくいかないプログラムの場合には、動的調整を行っても効果が望めない。

5.2 プリフェッチの動的調整による効果

シーケンシャルメモリアクセスプログラムにおけるプリフェッチ注釈の動的調整では、ある程度の時間短縮効果を実現できた。これは実行するプログラムのメモリアクセスパタンが明確であることにより、実行時間予測式が適切に決定できたためだと思われる。しかしプログラムのアクセスパタンが事前に解析できたとしても、結果として実行時間短縮効果が得られない場合がある。それはプリフェッチを行う際に、対象ページのページ処理時間がページフェッチ時間に対してあまりにも小さい場合や、また全体の実行時間におけるメモリアクセス以外の処理時間があまりにも長い場合はプリフェッチの効果が得られない。

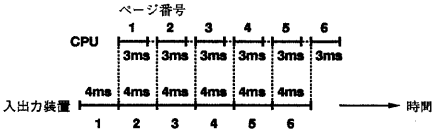

図 8: ページ処理時間がページフェッチ時間の最小値よりも小さい場合

図 8は、ページ処理時間がページフェッチ時間の最小値よりも小さい場合の処理を示している。ページ処理時間は 3ms で一定であるものとし、ページフェッチ時間の最小値は 4ms とする。図 8では、現在アクセスするページの 2 ページ先をプリフェッチしている。しかし、仮にページフェッチ時間が最小の 4ms だったとしても、2 倍のページ処理時間分のページフェッチ時間を隠蔽することはできない。プリフェッチ距離を大きくしても図 8と全く同様のことが起きる。ページ処理時間が起こり得るページフェッチ時間よりも必ず小さいので、これ以上の隠蔽効果が出ないのである。したがってページ処理時間がページフェッチ時間の最小値よりも小さい場合には、プリフェッチ距離を 1 よりも大きくしても実行時間が短縮されることはない。

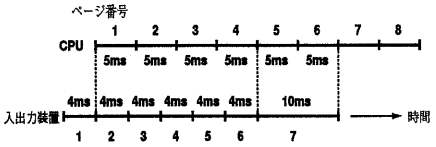

図 9: ページ処理時間がページフェッチ時間の最小値よりも大きい場合

図 9は、ページ処理時間がページフェッチ時間の最小値よりも大きい場合の処理を示している。ページ処理時間は 5ms で一定であるものとし、ページフェッチ時間の最小値は 4ms とする。図 9では、現在アクセスするページの 2 ページ先をプリフェッチしている。ページフェッチ時間は 4ms よりも大きくなることがあるが、仮に図 9のようにページ番号 6 まで最小時間の 4ms でページフェッチできたとすると、ページ番号 7 を 10ms 以内にメモリにロードすれば、ページフェッチ時間をページ処理時間で完全に隠蔽できることになる。つまりプリフェッチ距離を 2 ページ先にすれば、2 倍のページ処理時間分のページフェッチ時間を隠蔽できる場合があることを示している。このことは、ページ処理時間がページフェッチ時間の最小値よりも大きい場合、プリフェッチ距離を n ページにした時、n 倍のページ処理時間分のページフェッチ時間を隠蔽できる場合があることを意味している。この隠蔽できる割合は、ページ処理時間が大きくなるにつれ増加する。したがってページ処理時間がページフェッチ時間に対して大きいときには、プリフェッチ距離を大きくすることでページフェッチ時間を隠蔽できる割合が大きくなる。

この考えをもとにして、プリフェッチ距離を大きくしていくと、理論的にはある距離で起こり得るすべてのページフェッチ時間を隠蔽できるところに到達する。それ以上にプリフェッチ距離を大きくしても、実行時間が短縮されることはない。ページ処理時間がページフェッチ時間の最大値よりも大きい場合、プリフェッチ距離を 1 ページ先にすれば、完全に隠蔽できることになる。それ以上にプリフェッチ距離を大きくしても、実行時間の短縮は望めない。

プリフェッチ注釈を静的に行うことに対する動的調整の優位性は、システム負荷や実行環境の変化に適応できることである。システム性能や実行環境が常に一定の条件であれば、静的な方法が最適な場合もあり得るが、それでは注釈を適用できる範囲が狭まってしまうことになる。動的調整機構では、状況が変化しても適切な時間短縮効果が望める。本稿では単一の環境で評価を行ったが、異なるハードウェア環境での評価は今後の課題である。

5.3 ページ処理時間で隠蔽できないページフェッチ時間の近似関数について

プリフェッチの際、ページアクセス処理とページフェッチ処理は並行に行われる。このときページフェッチ時間はページアクセス時間で隠蔽できる。隠蔽できる程度は、プリフェッチを指示するときのプリフェッチ距離によって変化する。シーケンシャルメモリアクセスプログラムの実行時間予測式では、1ページ当たりの実行時間は、ページ処理時間とページ処理時間で隠蔽できないページフェッチ時間との和で表される。本稿では、プリフェッチ距離(時期)によるページ処理時間で隠蔽できないページフェッチ時間として、比較的単純な関数で近似した。ここで用いた近似関数は概ね実測値に近いが、十分に正確だとはいえない。複雑な関数を用いることで、より的確なプリフェッチ距離を求めることができる。しかし、実行時にプリフェッチ距離を求める計算自体が複雑になり、オーバヘッドが増加する。近似関数を決定する際には、正確さを求めるだけでは不十分であり、計算が容易であることも必要である。この点を考慮にいれた方法で近似関数を決定し、調整の効果を検証する必要がある。

6 おわりに

カーネルからのシステム情報を利用したページのプリフェッチング注釈の動的調整機構を提案し、評価した。提案した機構では、異なる環境でも動的にシステム性能などの情報を取得し、それを実行時間予測式中で適用することで、より適切な注釈の調整を行うことができる。本稿では、基本的なシーケンシャルメモリアクセスの場合の時間短縮効果を示せた。

今後の課題としては、プリフェッチ注釈について、一般的応用プログラムでの動的調整を評価することと、異なるハードウェア環境での動的調整の評価を行うことである。

また、その他の注釈(優先度、フラッシュ、書戻し、保持期限付き常駐)についての動的調整の実装を行い、有効性を検証することである。

7 謝辞

この研究を行う上で、数々の参考となる助言をいただいた弓場研究室の諸氏に感謝いたします。

参考文献

[1] http://www.cs.utah.edu/projects/flexmach/mach4/html/mach4-proj.html.

[2] http://www.cs.utah.edu/projects/flux/lites/html/index.html.

[3] Boykin, Kirschen, Langerman, and LoVerso. *Programming under Mach.* Addison Wesley, 1993. (邦訳: 岩本信一,Mach オペレーティングシステム, トッパン (1994)).

[4] Cao, Pei, Edward W. Felten, and Kai Li. Implementation and performance of application-controlled file caching. In *the First Symp. on Operating Systems Design and Implementation*, pp. 165–177. USENIX, 1994.

[5] 乾和志, 菅原圭資. 分散 OS Mach がわかる本. 日刊工業新聞社, 1992.

[6] 小野貴寛, 大澤範高, 弓場敏嗣. ページ操作に対するシステムとアプリケーション間 の協調動作支援機構. 第54回情報処理全国大会講演論文集 (分冊 1), pp. 235–236, Mar 1997.

[7] 大澤範高, 弓場敏嗣. ページ操作に対する注釈の低オーバーヘッド実現機構. 信学技報 CPSY96-34, pp. 37–42, May 1996.