データベース・システム 64 － 8
（１９８８．３．１５）

並行処理における二次記憶スケジューラの最適化

掛下哲郎　上林弥彦
九州大学工学部

従来行われている並行処理方式の評価は、データアクセス時間が一定であるという仮定をおいていた。しかし、仮想記憶管理を行う計算機システムにおいては、必要データが主記憶上に存在するかどうかによってデータアクセス時間が大幅に変化する。本稿では、この点に着目して最適な直列可能スケジューラに関する考察を行う。主な成果は、次の2点である。

(1) データの変更を考慮した上で、必要主記憶容量と二次記憶アクセス回数の線形結合からなるコスト関数を最小化するページングアルゴリズムを示した。

(2) 上記のアルゴリズムを用いる最適な直列可能スケジューラは、P＝NPでない限り、多項式アルゴリズムでは実現できないことを示した。

An Optimal Secondary Storage Scheduler for Transaction Processing

Tetsuro KAKESHITA and Yahiko KAMBAYASHI

Department of Computer Science and Communication Engineering,
Kyushu University, Hakozaki, Higashi, Fukuoka 812, Japan

Most of the work on the performance evaluation of concurrency control algorithms assumes that the data access time is invariant. In a virtual memory environment, however, the access time depends largely on whether the data is in main memory. From this viewpoint, this paper characterizes optimal serializable schedulers. Two main results are:

(1) A paging algorithm, which takes data updates into consideration and minimizes a linear cost function consisting of the memory size and the number of secondary storage I/O; and

(2) A proof which shows that, unless P＝NP, no polynomial time algorithm can realize an optimal cost scheduler which uses the paging algorithm described above.

1. まえがき

記憶階層を持つ計算機上でトランザクション処理を行う場合、各処理単位は必要データを主記憶上に転送してからアクセスしなくてはならない。主記憶が実行中処理単位の必要とするデータを全て納めるだけの容量を持たないならば仮想記憶管理が必要になる。本稿では、このような仮想記憶管理に伴うI/Oコストや主記憶コストに基づいて、並行処理スケジューラを評価する。そのために、データの操作モード(読み出し･書き込み)を考慮した最適ページングアルゴリズムを示し、それを用いた最適スケジューラの性質を調べる。

以下では、まず2節で関連する研究についての概観を行って、3節で基本的なモデルを提案した後、4、5節でデータ操作モードを考慮して、可変領域のもとでの最適ページングアルゴリズムを求める。6節では、そのページ追い出し方式を用いる最適な並行処理スケジューラについて考察し、その実現が一般に手に負えないことを証明する。

2. 関連研究

計算機上で並行処理を行う目的は、(1)速度差の大きなハードウエアを並列に動作させることによって、低速デバイスの動作中にも高速デバイスを遊ばせないようにする、(2)複雑さが同程度の処理単位に対する応答時間を均一化する、の2つである。(1)の中で最も重要なのが主記憶バッファと二次記憶デバイスとの速度差である。しかし、共有データベース上で任意の並行処理を許すと、正しくない結果を生ずる場合があるので、並行処理制御が必要になる。これまでに、種々の並行処理方式が提案されており、それらの性能評価も行われている[AGRA87, TAY87]。しかし、それらは記憶階層を考慮していないため、必要データが主記憶上にある場合とない場合とでデータアクセス時間が大幅に異なるという事実を反映できない。

データベースの研究において二次記憶アクセスを問題にしているのは、質問処理だけである。一般の質問処理は単一の質問に対する最適化を考えているので、主記憶上で複数の質問がデータを共有するということは考慮されていない。複数質問に対する最適化も研究されているが[KIM85]、基本的に各処理の実行は直列処理であり(質問の並行実行は許す)、データの変更を行う処理の並行実行は許さない点でトランザクション処理には向かない。

また、OSにおいて種々の記憶管理方式が提案されており、理論的にも研究されているが[COFF73]、ページングアルゴリズムの入力はページ参照列であり、共有データの変更を許していないという点で問題がある。これは、OSにおいて共有データの変更がまれであり、その上での操作競合が問題になりにくかったことに原因があると考えられる。本稿ではデータの変更を考慮することによって、OSの場合よりもさらに低コストの最適アルゴリズムを求めている。

3. 基本的事項

3.1 処理単位

処理単位T_iは操作集合$Refs(T_i)=\{O_{ij}|1\leqq j\leqq n_i\}$、$O_{ij}=r_{ij}(x)$ or $w_{ij}(x)$ ($r_{ij}(x)$や$w_{ij}(x)$は$x\in D$(データ集合)に対する読み出し･書き込み)と、その上で定義されている半順序集合$<_i$によって定義される。したがって、同一データに対する複数回のアクセスを許している。処理単位集合$T=\{T_0, T_1, \cdots, T_n, T_f\}$について参照列は組$(Refs(T), <)$である。ここに、$Refs(T)=\cup Refs(T_i)$、$<$は各$<_i$に矛盾せず、$w_{01}(D)$と$r_{f1}(D)$をそれぞれ下限･上限とする全順序集合、$T_0(T_f)$はデータベース上の全てのデータに対して書き込み(読み出し)を行う仮想的な処理単位$(T_0=(w_{01}(D), \phi), T_f=(r_{f1}(D), \phi))$である。[BRZO85]

3.2 システムモデル

本稿で考察する基本システムは、主記憶と二次記憶からなっている。最近のLSIの高密度化に伴って大容量主記憶が実現できるようになりつつあるが、全てのデータを保持できるだけの容量は持たないので、データベースは二次記憶上に記録される。各処理単位はデータアクセスが必要になると、まず主記憶上で目的のデータを探し、あればそれをアクセスし、なければ主記憶上に必要データをロードする。主記憶上には、(1)データベースの一部(主記憶バッファ)、(2)処理単位の実行形式コード、(3)データベース管理システム(DBMS)、の3つが置かれるが、それらに割り当てられる主記憶サイズは固定しているとは限らない。そこで、主記憶バッファ上では、必要な領域をあまり大きくしないように、不要データは早めに追い出される。二次記憶上にはデータベースが置かれるが、多重版並行処理を行う場合には、読み出し専用の旧版も置かれる。さらに、本稿では単一処理単位が同一データに複数回の書き込みを行うことを許すので、中間的な版も二次記憶上に置かれることがある。この様子を示したのが図1である。この中で、本稿では主記憶バッファに注目して考察する。

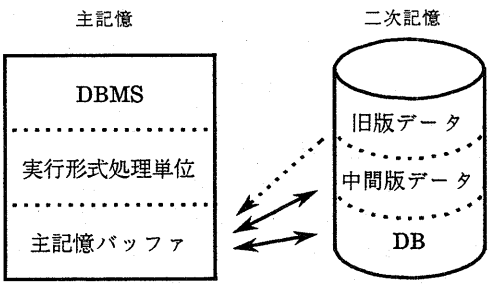

図1　システムモデル

主記憶バッファを操作するために必要なソフトウエアモジュールが、(1)スケジューラ、(2)ページインタプリタ、(3)ページングマネージャーの3つである(図2)。スケジューラは与えられた処理単位集合に対して適当な直列可能参照列(6節で定義)を出力する。この参照列は、各データに対するr/w列なので、データに対して適当なページを割り当てる必要がある。それを行うのがページインタプリタである。しかし、本稿では議論を単純化するために、各データは独立したページに割り当てられると仮定する。このようにして生成された操作列を処理するのがページングマネージャーである。ページングマネージャーは操作列に応じて最適な追い出しを行う。本稿のモデルでは、この3つのモジュールによって記憶コストを最小化する。以下、ページングマネージャー·スケジューラの順で議論する。

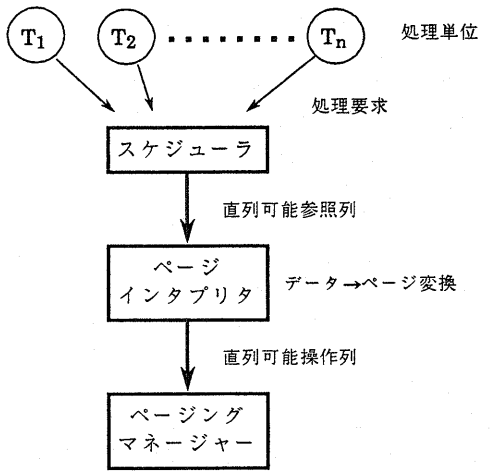

図2　ソフトウエアモデル

4. データの操作モードを考慮した追い出し方式

本節では、(1)スケジューラによって出力された参照列中の各データがそれぞれ1ページに対応する、(2)同一データであっても異なる書き込み操作によって書き込まれたものは異なるページに対応する、(3)デマンドページングのみを考える、の3点を仮定してページングマネージャーの解析を行う。問題は次のように定式化できる。

【問題1】　ページ集合P、操作集合O={r, w}に対して、長さnの操作列$o_1(x_1)\cdots o_n(x_n)$を考える。ここに、$o_i(x_i)$ ($i=1, \cdots, n$, $o_i \in O$)は時刻iに到着したx_iへの操作である。$i=2, \cdots, n$-1に対しては$x_i \in P$であるが、特別な操作として、$o_1(x_1)=w(D)$, $o_n(x_n)=r(D')$ ($D, D' \subseteq P$は、各データへの最初または最後の書き込みに対して割り当てられたページの集合)を考える。時刻iにおいて主記憶上に保持されることになったページ数をP_i、時刻iに必要なI/O回数をQ_iとしたときに、次式を最小にするページングアルゴリズムを求めよ。

$$C=U\sum_{i=1}^{n}P_i+R\sum_{i=1}^{n}Q_i$$

ここにUは主記憶コスト(1ページを主記憶上で単位時間保持するために必要なコスト)であり、RはI/Oコスト(1ページを二次記憶と主記憶の間で転送するために必要なコスト)を示す。□

$o_1(x_1)$と$o_n(x_n)$はデータベースに初期値を書き込む操作と最終値を読み出す操作である。この両者とも仮想的な操作なので、Q_1, P_n, Q_nの値は0であると考えられる。しかし、本稿では、データベースの一部を主記憶上に置いた状態を初期状態と考えるので、P_1については考慮する必要がある。

容易に分かるように、各ページに関する追い出し方式は互いに独立である。したがって、各ページxに関するコストを最小化すれば、全体のコストCを最小化できる。そこで、操作列からxに対する操作のみを取り出した列$o_{i_1}(x)o_{i_2}(x)\cdots o_{i_{n(x)}}(x)$について考える。ここで、仮定(1), (2)によって各ページは1回しか書き込まれないので、$o_{i_1}=w$, $o_{i_k}=r$ ($k=2, \cdots, n(x)$)が成立する。この変換によって、考えるべき問題は次のようになる。

【問題2】　ページxに対する操作列$w(x)r_1(x)\cdots r_m(x)$と、操作列中の$i+1$番目とi番目の操作の到着時刻の差d_i ($i=1, \cdots, m$)について、そのi番目の操作と$i+1$番目の操作の間に必要とされるコストC_iの和ΣC_iを最小化する追い出し方策を求めよ。□

本節で考慮する参照列は最初と最後に仮想的な操作を含むので、それらの操作が問題2で考慮する操作列に含まれる場合と含まれない場合とで最適な追い出し方策が異なる。以下、仮想的な操作が操作列

に含まれない場合について考察し、節の最後で、それらが含まれる場合について触れる。

本稿のコスト基準のもとでは、各操作を行うためのコストは必要ない。また、$r_m(\mathbf{x})$を行った後には$\mathbf{x}$は使用されないので、$\mathbf{x}$を単に捨てることができる。したがって、最小化の目安としてのコストは上記のものでよい。

各iについてi番目の操作終了後、$\mathbf{x}$は、(1)時間z_i後に追い出される(一般性を失うことなく$z_i<d_i$と仮定できる)、(2)$i+1$番目の操作が到着するまで主記憶上に保持される、のいずれかを選択される。$\mathbf{x}$が追い出される場合には、次に$\mathbf{x}$を読み出すときにI/Oが必要になる。また、最初の追い出しに限って、二次記憶上に$\mathbf{x}$が存在しないので追い出しのためのI/Oが必要になる。したがって、そのときのC_iは$2R+z_iU$、2度目以後の追い出し時のC_iは$R+z_iU$になる。この式によって、$\mathbf{x}$を追い出す場合にコストを最小化するためには、$\mathbf{x}$を直ちに追い出す($z_i=0$)ものだけを考えればよいことが分かる。その場合には、C_iはそれぞれ$2R, R$になる。また、$\mathbf{x}$を主記憶上に保持する場合には、$C_i=d_iU$が成立する。したがって、$\mathbf{x}$を最初に追い出す時刻の選択によってC_iの値が影響されることが分かる。

$\mathbf{x}$に対する与えられた操作列について、$\mathbf{x}$の最初の追い出しがk番目($1\leqq k\leqq m$)の操作直後に発生したとする。このとき、それ以前の時刻には$\mathbf{x}$は主記憶上に保持されているので、追い出すまでの時間にUを乗ずることによって$k-1$番目までのC_iの和を求めることができる。一方、$k+1$番目以後のC_iに対してはRまたはd_iUを自由に選択できる。$C_k=2R$であることを考慮すると、$i>k$について、

$d_i>R/U$のとき　i番目の操作直後に$\mathbf{x}$を追い出す、

$d_i\leqq R/U$のとき　i番目の操作後、$\mathbf{x}$を主記憶上に保持する、

ことによって$\mathbf{x}$の最初の追い出しをk番目の操作直後に行った場合の最適追い出しを行うことができる。これによって、問題はkの値を決定することに帰着されるが、それについて以下の補題が成立する。

【補題1】　k番目の操作直後に$\mathbf{x}$を追い出すと仮定する。このとき、その追い出し方式が最適になるためには$d_k>R/U$を満足しなければならない。□

【証明】　$d_i\,(i=1,\cdots,m)$についていくつかの場合分けを行う。

全てのd_iがR/Uより小さい(または等しい)場合

　$m+1$番目の操作直後まで$\mathbf{x}$を主記憶上に保持しておく方式

R/Uより大きい$d_i(i<k)$が存在する場合

　条件を満たすd_iの中で、iが最も大きいものをd_tとする。t番目の操作直後に$\mathbf{x}$を追い出す方式

R/Uより大きい$d_i(i>k)$が存在する場合

　条件を満たすd_iの中で、iが最も小さいものをd_tとする。t番目の操作直後に$\mathbf{x}$を追い出す方式

各場合について、それに続いて示した追い出し方式を考えると、その条件のもとでの最適追い出しにおいて補題の方式のコストよりも、それらの方式のコストが小さくなることは容易に示せる。したがって、補題が成立する。□

この補題によって距離d_kがR/Uより小さいようなk番目の操作直後を最初の追い出し点として選択する方法は、最適にはならないことが示された。また、$d_k>R/U$を満足するkが複数個存在する場合には次の補題が成立する。

【補題2】　$d_i, d_j>R/U$とする($i<j$)。このとき、i番目の操作直後に$\mathbf{x}$を追い出す方式は、j番目の操作直後に追い出す方式と比較して必ずコストを小さくできる。□

【証明】　$i<t<j$、かつ$d_t>R/U$を満たすようなtが存在しない場合について補題を証明すれば、そうでない場合にも結果を帰納的に拡張できるので、以下、そのような場合について証明を行う。i番目の操作直後に$\mathbf{x}$を追い出す場合には、j番目の操作直後に$\mathbf{x}$を追い出す時にI/Oコストを必要としない。したがって、この場合のC_i, C_jは次のようになる。

$$C_i=2R,\ C_j=R$$

また、j番目の操作直後に$\mathbf{x}$を追い出す場合にはi番目の操作直後で$\mathbf{x}$を追い出さない。したがって、この場合のC_i, C_jは次のようになる。

$$C_i=d_iU,\ C_j=2R$$

$i<t<j$を満たす各tについては、$d_t\leqq R/U$であるから、両者とも$C_t=d_tU$となる。最初の追い出しはi(またはj)番目の操作後であるから、$t<i$を満たすtについては、両者のC_tともd_tUであり、$t>j$を満たすtについては、最適な追い出しを考えると、

$d_t>R/U$のとき、$C_t=R$

$d_t\leqq R/U$のとき、$C_t=d_tU$

となるので、$t\neq i, j$については両者のC_tの値は等しくなる。これらの事実によって、両者のコスト差は、

$$R-d_iU<0$$

となる。□

この補題によって、$d_k>R/U$を満足する最小のkについて、k番目の操作を終了した直後に追い出しを行う方法と、最後まで追い出しを行わない方法の

何れかが最適になることが分かる。両者のコストを比較することによって次の定理が得られる。

【定理1】　次の式fを評価した結果によって最適アルゴリズムを決定することができる。

$$f=U\left[\sum_{i=1}^{m} max(0, d_i-\frac{R}{U})\right]-R$$

$f\geqq 0$ならば、①$d_i>R/U$であるか否かによって$\mathbf{x}$を追い出すか否かを決める。

$f<0$ならば、②$\mathbf{x}$を$\mathbf{r}_m(\mathbf{x})$が終了するまで主記憶上に保持する。□

【証明】　方法①のコストは、

$$\left[\sum_{i=1}^{m} min(d_iU, R)\right]+R$$

であり、②のコストは、

$$\sum_{i=1}^{m} d_iU$$

であるから、両者の差をfとすることにより、定理が導かれる。□

ここで②が採用された場合には、$\mathbf{x}$の値はデータベースに反映されない。これは処理単位の確定(コミット)を必ずしも行わないことに対応する。確定を考慮した場合の議論は、5.4節で行う。

最後に、問題2の操作列で、(1)$\mathbf{w}(\mathbf{x})=\mathbf{w}(\mathbf{D})$である場合と、(2)$\mathbf{r}_m(\mathbf{x})=\mathbf{r}(\mathbf{D}')$である場合について考察する。(1)の場合には、時刻1で$\mathbf{x}$を追い出したときのC_1の値がRになることを考慮すると、定理1の①,②の他に次の方法③を考慮する必要がある。

時刻1で$\mathbf{x}$を追い出し、その後は$d_i>R/U$であるか否かによって$\mathbf{x}$を追い出すか否かを決める。

また、(2)の場合には、$P_n=Q_n=0$であることを考慮すると、$\mathbf{r}_{m-1}(\mathbf{x})$の処理終了までに$\mathbf{x}$が二次記憶に反映されているならば、$\mathbf{r}_m(\mathbf{x})$を無視できる。そうでない場合には、$\mathbf{r}_m(\mathbf{x})$を考慮する必要がある。いずれにしても、これらの方法の最適コストは容易に求められる。

5. 最適ページングアルゴリズムに関する考察

5.1　コスト関数

通常のページングは、固定サイズの主記憶を割り当てておいて、その中にページを保持できなくなったときのみ追い出しを行うことが多い。それに対して、本稿の方式は、コスト関数Cの値を最小化するために主記憶バッファに余裕があってもページを追い出すことがある。このコスト関数は[PRIE76]によるものであり、主記憶バッファに割り当てられている領域のサイズが変動する場合の最適アルゴリズムに対応している。実際、本稿のモデルでは主記憶上に、(1)主記憶バッファ、(2)処理単位の実行形式、(3)DBMS、の3つが割り当てられているので、このコスト関数を用いることにより他の2つの部分(これらに対する操作は読み出しのみなので、[PRIE76]の方法によって最適なページングを行うことができる)との主記憶上での競合も解決できる。即ち、R/Uの値を調節することによってページフォルト率が均等になるようにすればよい。

5.2　必要主記憶容量

上記の議論によって、本稿の最適ページングアルゴリズムが必要とする主記憶容量は不定であることが分かった。しかし、定理1から直ちに次の系を導くことができる。

【系】　ページ$\mathbf{x}$をアクセスしたときに、$\mathbf{x}$の次のアクセスまでの距離が$2R/U$を越えるならば、最適ページングは$\mathbf{x}$を直ちに追い出す。□

【証明】　定理1のfにおいて$d_t>2R/U$を満たすtが存在するならば、fは必ず正になる。したがって、最適アルゴリズムは$d_i>R/U$を満たす各i番目の操作直後に$\mathbf{x}$を追い出す。□

時刻iに主記憶上に置かれている各ページは時刻$i-2R/U$以降に最後のアクセスをされたことが上記の系によって分かる。このうち、時刻$i-2R/U$に最後のアクセスをされたページは、時刻iに操作されないと仮定すると、上記の系によって主記憶上に置いておくことはできないので、P_iには数えられない。また、時刻iで操作される場合には、2つの操作が同一ページに対して行われるので、P_iはそのうちの高々1つしか数えない。したがって、任意の時刻iにおけるP_iは$2R/U$より大きくはならない。以上の議論は仮想的な$\mathbf{o}_1(\mathbf{x}_1)$と$\mathbf{o}_n(\mathbf{x}_n)$を除いて行ったが、それらを考慮しても同様の結論を得ることができる。ここで、次のような部分操作列を考えると、各$\mathbf{x}_i$が操作列の他の部分で操作されないならば、その最適処理において$P_k=2R/U(k=2R/U)$となる。

$\mathbf{o}_1(\mathbf{x}_1)\cdots\mathbf{o}_n(\mathbf{x}_n)$, $\mathbf{n}=4R/U$

$\mathbf{o}_i=\mathbf{w}$ $(1\leqq i\leqq 2R/U)$, $\mathbf{o}_i=\mathbf{r}$ $(2R/U<i\leqq 4R/U)$

$\mathbf{x}_i=\mathbf{x}_{i+2R/U}\in\mathbf{P}$

これによって、P_iが$2R/U$に等しくなるような操作列を構成できることが分かるので、この値は最小の上限である。したがって、比R/Uは実際のコストばかりではなく、主記憶容量も考慮にいれて選択する必要がある。

5.3 二次記憶アクセス時間

一般には二次記憶アクセス要求を出してから、それが実際に行われるまでにはある時間tが必要である。4節の解析においてはこの時間は0であるとしてきたが、ページ追い出し時にはそのページを二次記憶アクセス時間tだけ余分に主記憶上に保持する必要が生じる。それによって、コストC_iはtUだけ増加する。tの平均値t^*について考えると、それを決定する3要素は、(1)二次記憶ハードウエア、(2)ディスクスケジューリングアルゴリズム、(3)ディスクアクセス要求の到着率λである。これらのうち(1)と(2)は各システムに固有のものなので、λの値が問題になる。待ち行列理論によって一般に次の命題が成立する。

【命題】 平均二次記憶アクセス時間t^*はアクセス要求の到着率λの増加関数である。□

t^*の関数形は一般に決められない。したがって、補題2や定理1のように主記憶コストを二次記憶コストで置き換えることが必要になるものはλを変化させるので一般に成立しない。

しかし、最近の主記憶の大容量化にともなって数10メガバイトの主記憶バッファが実現可能である。ページサイズを1キロバイトとしても、この主記憶バッファは数万ページを保持することができる。したがって、追い出されるページに対する最小の参照間隔R/Uは数万参照である。現在のトランザクション処理システムは1秒間に1000個の処理単位(各処理単位は数十回のデータアクセスを行う)を実行する能力しか持たないので、数万回の参照を行うのに約1秒を必要とする。ところが、平均二次記憶アクセス時間t^*は約30ミリ秒なので、1秒と比較して十分小さいと考えてよい。したがって、二次記憶アクセス時間がコスト関数に与える影響は無視できる程度である。この観察に基づいて本稿では二次記憶アクセス時間を考慮していない。

5.4 確定を考慮した場合の一般化

4節の議論においては、処理単位の確定を考慮しなかった。それは操作列が与えられており、特定のページに対する最後の操作がどれであるかということが分かるからである。しかし、一般の並行処理においては、処理単位が変更したページを実行終了時にデータベースに反映する確定操作が必要になる。それによって4節の定理1は次のように変更されなくてはならない。

【定理1'】 ページxを必ず追い出す仮定のもとでは、$d_i > R/U$であるか否かによってxを追い出すか否かを決めるアルゴリズムが最適である。□

証明は、定理1と同様なので省略する。これは、[PRIE76]における結果と同一である。しかし、本稿の処理単位モデルでは1つのデータに対する複数回の書き込みを許しているので、全てのページについてこのアルゴリズムを適用する必要は必ずしもない。上記のアルゴリズムは、各データに対して処理単位が行った最後の書き込みに対応するページについてだけ適用すれば十分である。しかし、全てのページに対して確定を考慮したI/Oを仮定すれば、5.2節の系も次のようになるので、必要とされる主記憶容量は最大R/Uになる。

【系'】 ページxをアクセスしたときに、xへの次のアクセスまでの距離がR/U以上ならば、確定を考慮した最適ページングはxを直ちに追い出す。□

5.5 プリページング

プリページングの長所は、複数ページを一括ロードすることによってI/Oコストを下げられることにある。しかし、本稿のコスト基準によるとプリページングによってページを主記憶上に置いておく時間が増加するので、両者の差を評価する必要がある。このとき、一括ロードされたkページへの読み出しを一括ロードの直後にスケジュールすることによって、記憶コストの増加を

$$\frac{1}{2}k(k-1)U$$

に抑えることができる。これは、狭義二相ロック方式などで、確定を待っている処理単位に、データを確定後直ちに引き渡すことで実現できる。上記の一括ロードにおいても暗示されているが、記憶コストを最小化するためには、一括ロードは読み出し要求と同時に行わなければならない。即ち、書き込み操作の実行直前に一括ロードを行う方法は、その操作の後に到着した読み出し操作と共に一括ロードする方法と比較してコストが大きくなることが示せる。

6. 記憶コストを最小化するスケジューラ

本節では、ページングマネージャーに入力したとき、その処理コストCが最小になるような正しい参照列の生成を行うスケジューラについて考える。まず、参照列の正しさの基準として直列可能性の概念を導入する。

参照列中の各$r_i(x)$についてそれを参照列の中で先に出現するxへの書き込み$w_j(x)$に写像する関数I(解釈)を考える。Iの中で$r_i(x)$をその直前に出現する$w_j(x)$に写像するものを標準解釈と呼ぶ。直列参照列とは、各処理単位をある順番(但しT_0は最初、T_fは最後におく)で直列に並べたものであり、その標準解釈Ihと参照列の解釈Iが一致したとき、2つの参照列は多重版等価である。参照列に対してそれと等価な直列参照列が存在するとき、この参照列は多重版直列可能である。ここで、Iもまた標準解釈であるときには、その参照列は単版直列可能である。一般に、2つの参照列(少なくとも一方は直列)の等価性判定は多項式時間でできるが、参照列の直列可能性判定問題はNP完全である。[BRZO85]

本節でも、(1)参照列中の各データがそれぞれ1ページに対応する、(2)同一データであっても異なる書き込み操作によって書き込まれたものは異なるページに対応する、の2点を仮定して4節の最適ページング方策のもとでスケジューラの解析を行う。問題は次のように定式化できる。

【問題3】　処理単位集合$T=\{T_0, T_1, \cdots, T_n, T_f\}$, $T_0=(w_{01}(D), \phi)$, $T_f=(r_{f1}(D), \phi)$, Dはデータ集合, $T_i=(Refs(T_i), <_i)(i=1,\cdots,n)$と、コスト$R, U$に対して、Tからなる直列可能参照列の中で、記憶コストCを最小にするものを求めよ。□

Tからなる任意の参照列のコストは$R\times|Refs(T)|$を越えないことが示せるので、問題3は次の問題4を$K=pU+qR$ (p, qは整数)の値を変えながら繰り返し解くことに帰着される。

【問題4】　処理単位集合$T=\{T_0, T_1, \cdots, T_n, T_f\}$, $T_0=(w_{01}(D), \phi)$, $T_f=(r_{f1}(D), \phi)$, Dはデータ集合, $T_i=(Refs(T_i), <_i)$ $(i=1,\cdots,n)$、コストR, Uと正定数Kに対して、Tからなる直列可能参照列の中で、記憶コストCがK以下になるものが存在するか?□

ところが、次の定理が成立する。

【定理2】　問題4は、NP完全である。□

【証明】　問題4が、クラスNPに属することは次のようにして示せる。即ち、$<_i$に矛盾せず、$w_{01}(D)$と$r_{f1}(D)$をそれぞれ下限・上限にするような全順序集合<と、集合$\{1,\cdots,n\}$上の順列tをそれぞれ推測したとき、参照列$(Refs(T), <)$が直列参照列$T_0T_{t(1)}T_{t(2)}\cdots T_{t(n)}T_f$と等価になるかどうかは多項式時間で判定でき、また、その実行コストがK以下になるかどうかも定理1のfを評価することによって、多項式時間で判定できる。

また、問題4のNP困難性は、次に示すDirected Optimal Linear Arrangement問題(cf. [GARE79])を問題4に帰着することによって示すことができる。

【問題5】　有向グラフ$G=(V, E)$と正整数Kに対して、次の2条件を満足する一対一対応$f: V\to\{1, 2, \cdots, |V|\}$は存在するか?

(1) $(u,v)\in E$ならば、$f(u)<f(v)$

(2) $\Sigma_{(u,v)\in E}\{f(v)-f(u)\}\leqq K$　□

問題5の任意のインスタンス$G=(V,E)$とKに対して、図3に示すようなインスタンスを考える。有向グラフの巡回性は多項式時間で判定できるので、一般性を失うことなくGは非巡回であるとしてよい。したがって、T_1は処理単位の定義を満たすので、図3に示すものは問題4のインスタンスである。このインスタンスについて次の主張を示せば定理2が証明できる。

【主張】　与えられたGとKに対する問題5の答えがYesであるための必要十分条件は、図3のインスタンスに対する問題4の答えがYesになることである。□

紙数の関係で完全な証明は省略するが、上記の主張のうち必要性は比較的容易に証明できる。十分性に関しては、各$v\in V$に対応する$Refs(T_1)$の要素の集合

$S_v=A_v\cup B_v$

$A_v=\{w_1(y_{(v-1)k'+i}), r^{v-1}(z_i), w^v(z_i), r(y_{(v-1)k'+i}) \mid i=1,\cdots,k'\}$

$B_v=\{\eta_{v,i} \mid i=1,\cdots,|E|\}$

と最小コストを与える直列可能参照列$F^{-1}(1)F^{-1}(2)\cdots F^{-1}(|Refs(T)|)$を生成する関数$F: Refs(T)\to\{1,\cdots,|Refs(T)|\}$を考えたとき、任意の$u, v\in V$に対して$\max(F(S_u)) = F(r(y_{uk'})) < \min(F(S_v)) = F(w_1(y_{(v-1)k'+1}))$または、$\max(F(S_v)) < \min(F(S_u))$であれば明らかに主張が成立するので、そうでない場合を考える。そのときには、$F(\eta_{u,1}) < F(\eta_{v,1}) < F(\eta_{u,|E|})$を満たす$u, v\in V$が存在することが示せる。そのような$u, v$に対して$\max(F'(S_u)) < \min(F'(S_v))$を満たすようにFに関する$S_u, S_v$の像を変更した関数F'を考えると、F'に対してFは、B_u, B_vの要素によって生じる利得と、A_u, A_vによって生じる損失をこうむる。この利得はO(k')、損失は$\Omega(k'^2)$で与えられるので、k'を図3に示すように大きく取ると、F'がFより小さいコストを与えることになり、Fの最小性に反する。□

以上の証明は、対象とする処理単位が1つの場合を用いて行った。したがって、定理2は、複数処理単

処理単位集合$T=\{T_0, T_1, T_f\}$, $T_0=(w_{01}(D), \phi)$, $T_f=(r_{f1}(D), \phi)$, $T_1=(Refs(T_1), <_1)$。ここに、

$D=\{x_1, x_2, \cdots, x_{|E|}\}\cup\{y_1, y_2, \cdots, y_{k'|V|}\}\cup\{z_1, z_2, \cdots, z_{k'}\}\cup\{\xi\}$

$Refs(T_1)=$ $\{r(x_1), r(x_2), \cdots, r(x_{|E|})\}\cup\{w_i(x_1), w_i(x_2), \cdots, w_i(x_{|E|})|\ i=1, 2\}\cup\{r(y_1), r(y_2), \cdots, r(y_{k'|V|})\}\cup\{w_i(y_1), w_i(y_2), \cdots, w_i(y_{k'|V|})|$ $i=1, 2\}\cup\{r^0(z_1), r^0(z_2), \cdots, r^0(z_{k'})\}\cup\cdots\cup\{r^{|V|}(z_1), r^{|V|}(z_2), \cdots, r^{|V|}(z_{k'})\}\cup\{w^0(z_1), w^0(z_2), \cdots, w^0(z_{k'})\}\cup\cdots\cup\{w^{|V|}(z_1),$ $w^{|V|}(z_2), \cdots, w^{|V|}(z_{k'})\}\cup\{w_2(z_1), w_2(z_2), \cdots, w_2(z_{k'})\}\cup\{r^1(\xi), r^2(\xi), \cdots, r^s(\xi)\}$

ここに、$s=(|V|-2)|E|+2k'$, $k'=|E|(3|V|+5)$である。また、$k=2k'$を定義しておく。

$<_1=<^0\cup<^1\cup<^2\cup<^{|V|}\cup<^{|V|+1}$

$<^0=$ $\{w^0(z_i)<r^i(\xi), r^i(\xi)<w^0(z_{i+1})|\ i=1, \cdots, k'-1\}\cup\{w^0(z_{k'})<r^{k'}(\xi)\}\cup\{r^{k'}(\xi)<\omega|\ \omega$は、$w_1(y_{(v-1)k'+1})$, $v\in V\}$

Gの各節点v ($v=1, \cdots, |V|$)について、

$<^v=$ $\{w_1(y_{(v-1)k'+i})<r^{v-1}(z_i), r^{v-1}(z_i)<w_1(y_{(v-1)k'+i+1})|\ i=1, \cdots, k'-1\}\cup\{w_1(y_{vk'})<r^{v-1}(z_{k'})\}\cup\{r^{v-1}(z_{k'})<\eta_{v,1}\}\cup\{\eta_{v,i}<\eta_{v,i+1}|$ $i=1, \cdots, |E|-1\}\cup\{\eta_{v,|E|}<w^v(z_1)\}\cup\{w^v(z_i)<r(y_{(v-1)k'+i}), r(y_{(v-1)k'+i})<w^v(z_{i+1})|\ i=1, \cdots, k'-1\}\cup\{w^v(z_{k'})<r(y_{vk'})\}$

ここで、$\eta_{v,i}=$ $r(x_i)$ 有向枝$i=(\zeta, v)$ ($i=1, \cdots, |E|$)がEに含まれる場合

$w_1(x_i)$ 有向枝$i=(v, \zeta)$ ($i=1, \cdots, |E|$)がEに含まれる場合

$r^t(\xi)$ それ以外の場合 (tは添字が重複しないように選択する)

$<^{|V|+1}=$ $\{r(y_{uk'})<w_1(y_{(v-1)k'+1})|\ (u, v)\in E\}\cup\{r^{s-k'+i}(\xi)<r^{|V|}(z_i), r^{|V|}(z_i)<r^{s-k'+i+1}(\xi)|\ i=1, \cdots, k'-1\}\cup\{r^s(\xi)<r^{|V|}(z_{k'})\}\cup\{\omega<r^{s-k'+1}(\xi)|\ \omega$は、$r(y_{vk'})$, $v\in V\}\cup\{r^{|V|}(z_{k'})<\omega|\ \omega$は、$w_2(x_i)$ ($i=1, \cdots, |E|$), $w_2(y_i)$ ($i=1, \cdots, k'|V|$), $w_2(z_i)$ ($i=1, \cdots, k'$)のいずれか$\}$

$U=1$, $R=|Refs(T)|=(|V|+1)(|E|+5k')+2$

$K=$ $|E|+k'|V|+k'+K(|E|+2k)+k'|V|(|E|+k+1)+k'(k+1)(|V|+1)+|Refs(T)|-1+\Sigma_{i\in\{1, \cdots, |E|+k'(|V|+1)\}}i$

図3　定理2を証明するためのトランスフォーマー

位の並列実行ばかりではなく、単一処理単位のみを考慮する質問処理においても成立することが分かる。これによって、反復解法では問題3は手に負えないことが示された。しかし、一般に問題3は手に負えないことを示唆しているのが次の定理である。

【定理3】　P=NPでない限り、問題3を解く多項式アルゴリズムは存在しない。□

【証明】　問題3を解く多項式アルゴリズムAが存在すれば、それによって与えられたTとR, Uに対する最小コストを多項式時間で求められる。その値をKと比較することによって問題4を解くことができるので、問題4に対する多項式アルゴリズムA'が存在したことになる。ところが、問題4はNP完全なので、P=NPが成立する。□

7. むすび

本稿で述べた方式は、将来の情報を必要とする仮想的なページングアルゴリズムと、オフラインの直列可能スケジューラを用いて記憶コストを最小化するものである。直列可能スケジューラは、操作の意味を考慮しないという仮定のもとで最大の並行性を持つことが知られているので、本稿の方式による最小コストは、実際的なページングアルゴリズムとスケジューラを用いた場合のコストの下限を与える。しかし、本稿のスケジューラには、(1)多項式アルゴリズムによる実現が困難、(2)オンラインスケジューラでない、という2つの問題があるので、そのための議論が必要である。また、本稿ではページインタプリタの議論を行わなかったが、複数データの同一ページへの割り当て等、興味深い問題が残されている。

謝辞　本研究に関して討論頂いたNational University of SingaporeのTay教授と九州大学の今井助教授ならびに研究室の諸氏に感謝します。

参考文献

[AGRA87] Agrawal, R., et al., "Concurrency control performance modeling: alternatives and implications", ACM *Trans. Database Syst.*, Vol.12, No.4, December 1987, pp.609-654

[BRZO85] Brzozowski, J.A., Muro, S, "On serializability", International Journal of Computer and Information Sciences, Vol.14, No.6, pp.387-403, 1985

[COFF73] Coffman, E.G., Denning, P.J., Operating Systems Theory, Prentice-Hall, 1973, Chapter 5-7 (邦訳: 川口, 藤井, "オペレーティングシステムの理論", 日本コンピュータ協会, 1987)

[GARE79] Garey, M.R., Johnson, D.S., Computers and Intractability : A Guide to the Theory of NP-Completeness, Freeman, 1979

[KAKE87] 掛下, 上林, "記憶階層を考慮した並行処理のための発見的手法", 日本ソフトウエア科学会第4回大会, pp.479-482, 1987年11月

[KAKE88] 掛下, 上林, "記憶階層を考慮したトランザクション処理", 情報処理学会第36回全国大会7E-4, 1988年3月

[KIM85] Kim, W., "Global optimization of relational queries: a first step", In Query Processing in Database Systems, Springer-Verlag, pp.206-216, 1985

[PRIE76] Prieve, P.G., Fabry, R.S., "VMIN - an optimal variable-space page replacement algorithm", *Comm.* ACM, Vol.19, No.5, pp.295-297, May 1976

[TAY87] Tay, Y.C., Locking Performance in Centralized Databases, Academic Press, 1987