論　文

並列ディスクデータベースマシンにおける大量データアクセストランザクションの並行制御方式

正員　大森　匡[†]　　正員　喜連川　優[†]　　正員　田中　英彦[†]

Concurrency Control of Bulk Access Transactions on a Parallel Disk Database Machine

Tadashi OHMORI [†], Masaru KITSUREGAWA [†] *and* Hidehiko TANAKA [†], *Members*

あらまし　本論文では，大量のデータを処理するトランザクションの並行制御方式として「重み付きトランザクション順序グラフ」を用いた先読みスケジューラを提案する．実行環境としては，複数のディスクモジュールから構成された非共有型並列データベースマシンを想定している．このグラフはトランザクションの直列可能性とその実行コストを表しており，これを用いてデータ競合・リソース競合のより小さい直列可能スケジュールを生成できる．シミュレーションでは，提案したスケジューラは2相ロック，一括ロック，楽観的ロックに対し1.3倍から2倍の性能向上が得られている．

1. まえがき

データベースサービスにおけるバッチ処理は通常，ファイルの全数読出しや全数更新のように大量のデータをアクセスするトランザクションである．銀行業務などのオフライン処理ではこうしたトランザクションを短時間内に大量に実行する必要があり，データベースマシン上でのその並行実行は不可欠である．本論文では，こうしたトランザクションを大量データアクセストランザクション(Bulk Access Transaction，以下BATと略す)と呼び，その直列可能スケジューラを提案する．

BATは長時間トランザクション(Long Lived Transaction，以下LLT)に類別される[1]．従来の並行制御の研究はオンライン時の銀行業務などの短時間トランザクション処理[8]やCAD指向のLLT処理を扱っており，BAT間の並行処理は十分議論されていなかった．

BAT処理の特徴はデータ競合およびリソース競合が大変高い点にある．データ競合とはデータベースの各データが排他的に使用される際に生じる競合を指し，リソース競合とはディスク等の資源のふくそう(congestion)を言う[2]．BAT処理ではファイル単位ロックなどの粗いロック単位が必要であるためデータ競合は非常に高く，ロック要求はブロックされる確率が高い．更に大量データ処理のため資源のふくそうも高く，BATのアボート(取消操作)は無駄が大きすぎる．従来のロック規約はこうした環境を想定していなかった．

本論文では，並列ディスクモジュールからなるデータベースマシン上でのBAT処理を扱う．このとき，高いデータ競合・リソース競合はディスクモジュール間の負荷不均衡を起こし処理性能を低下させる．そこで，この二つの競合をできるだけ抑えた直列可能スケジュールを生成することが望ましい．

そのため我々は重み付きトランザクション順序グラフ(Weighted Transaction Precedence Graph，以下WTPG)と，これを用いた先読みスケジューラを提案する．WTPGはトランザクションの実行コストを辺に与えたトランザクション順序グラフである．WTPG上で臨界経路を最短とする直列可能順序を計算しこの順序に従うロック要求のみを認めることで，データ競合・リソース競合をより小さくしたスケジュールを出力できる．この計算問題は一般にNP困難であるため，ここではWTPGを鎖型WTPGと呼ぶ形に限定し，

† 東京大学工学部電気工学科
Faculty of Engineering, The University of Tokyo, Tokyo, 113 Japan

$O(n^2)$でこれを計算する．

提案されるスケジューラでは，トランザクションは実行開始時にそのアクセスデータ集合と「コスト情報」を宣言する．その代わり，このスケジューラではデッドロックがなく，ブロック連鎖による性能低下が少なく，実際に並列に動くトランザクション数が高く保たれる．

以下，仮定する環境とモデルについて**2.**で述べた後に**3.**でWTPGによる先読みスケジューラを提案し，**4.**で鎖型WTPGの最短臨界経路を計算する多項式時間算法を述べる．シミュレーション結果と議論は**5.**に示す．最後に**6.**で本論文を要約する．

2. モデルと仮定

2.1 実行環境

本論文では非共有型(Shared Nothing)データベースマシン[3]上で，各リレーションがレンジ分割されたファイル配置を仮定する．

非共有型データベースマシンは複数のディスクモジュール(Disk Module，以下DM)をネットワークで結合した構成をとる．各DMは，データベースを格納する複数のディスクをもつ計算機であり，フィルタ処理・クラスタ処理などを実行する．

レンジ分割[4]では，各リレーションは，指定された属性の値のレンジ(範囲，range)(例えば属性Idに対し，$0 \leq Id < 10$，$10 \leq Id < 20$，$20 \leq Id$の3レンジ)に応じて複数の区画(partition)に水平分割され，1区画ずつ各DMに配置される．

レンジ分割では，分割属性上のレンジ問合せはすべてのDMのうち，少数のDM上でのみ大量データ処理を実行する．これはDM間の負荷不均衡を引き起こし処理性能を低下させる．この負荷不均衡に対する単純な対策は各リレーションをランダムやハッシュによって全DM上に分割することである．しかし，この手法はソートやクラスタ化されたファイルをそのまま格納できない．これらのファイル編成法は短時間トランザクション処理には不可欠なものである．また，CPU負荷を低くするためには各リレーションをごく少数のDM上に分割した方が有利である[5]．この場合，リレーションの全数アクセスでもやはりDM間の負荷不均衡を引き起こす．従って本論文で前提とするDM間の負荷不均衡は一般的なものである．

2.2 トランザクションモデル

BATは，各区画への読出し・書込みステップの直列系列としてモデル化する．各ステップはアクセスする区画のもつ分割属性上のレンジに対応する選択条件をもつものとする．読出し・書込みには共有ロック・占有ロックを獲得し，障害回復のため一度得たロックはコミット時まで離さない．各BATは，自分の実行開始時に，その読出しデータ集合および書込みデータ集合を宣言する．ロックの単位としては区画を用いる．1区画へのロックは，その区画の分割属性上のレンジへの述語ロックである．更新時にレコード単位の占有ロックを使わない理由は，BATの場合大量データを更新するため，更新する区画上の共有述語ロックと衝突する確率が高いこと，またロック処理の負荷が高いことによる．

BATのコストは，データアクセスのコストのみによって決める．すなわち，区画Pのa%を読み出すとき，読出しステップ$r(P)$にはコスト$a|P|$を，書込みステップ$w(P)$にはコスト$2a|P|$を与える．($|P|$は区画Pのサイズ)．係数2は更新するデータをディスクから読み出すコストを表すために付けた．$r(P)$，$w(P)$はPに対してロックをもってもP上のインデクスによってその一部のみをアクセスできる．区画のサイズ単位としては特に値は定めず「単位(unit)」自体を使う．1単位時間は1単位サイズのデータをDM 1個上でアクセスする時間とする．1単位サイズはそのデータアクセス以外のコストを無視できるほど十分大きいとする．例えば，1単位サイズを10 MB，DM 1個のデータ処理速度を1 MB/secとすると1単位時間は10秒となり，制御負荷などは十分無視できる．

大量データ更新のため更新したデータはすぐにディスクにはき出し，コミットまで主記憶におかない．従ってコミットから終了までのコストは無視する．DMは，各ステップを他ステップによる割込みなしで実行する．大量データアクセスの高速化には，データをディスクの連続領域において割込みなしでアクセスする手法が効果的だからである．

並行制御装置はあるDMがアイドルになった際に呼び出され，そのDMの実行待ちキュー(Ready Queue)から次に動かすステップを選び出す．これは，ロックをとったトランザクションが各DMのふくそうによって実行されない状況をできるだけ回避するためである．各QueueはFCFS(First Come First Service)を仮定する．

2.3 スケジュール例

例題として，図1にトランザクション$T1$から$T4$の

コストモデルと区画5個のディスクモジュール $DM1$, $DM2$ への配置を示す．$r_i(P)$, $w_i(P)$ は，T_i の区画Pへの読出し，書込みステップである．この例では読出しは区画の100％，書込みは50％アクセスとしている．コミットは各トランザクションの最後にくるが以後省略する．

$T1$, $T2$, $T3$, $T4$ がこの順に並行制御装置(CC)のReady Queue(RQ)に並んでいるとする．図2(a)は先読み2相ロック規約(Cautious 2 Phase Lock，以下C2PL)によるスケジュールをガント図[7]で表したものである．但し，図2の1クロック(1 clock)は**2.2**に述べた1単位時間としている．C2PLは，ロック要求がブロックされずかつ将来にわたってデッドロックが発生しない限りそれを認める[6]2相ロック規約である．ここでは直列可能順は{$T1$, $T2 \rightarrow T3 \rightarrow T4$}となり，そのスケジュールはDM1上の $r1(D)$ によるふくそうとブロックの連鎖 $T2 \rightarrow T3 \rightarrow T4$ によって $DM1$, $DM2$ 上にアイドル時間を作る(図2(a))．

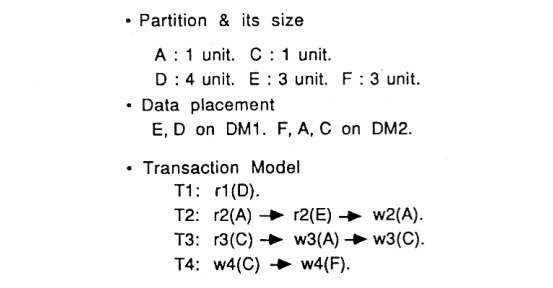


図1　トランザクションモデル例
Fig. 1　Transaction model and data placement.

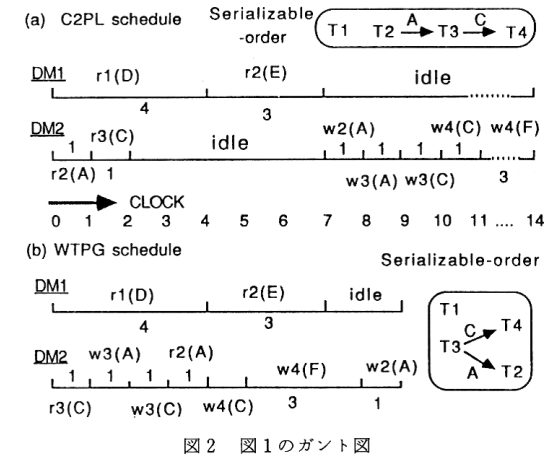


図2　図1のガント図
Fig. 2　Gantt chart of the transactions in Fig. 1 by (a) C2PL (b) WTPG.

このディスク上のふくそうとブロック連鎖は，リソース競合・データ競合として文献(2)に指摘されている．BAT処理では，粗いロック単位と大量データ処理のためにこの二つの競合は本質的に高く，ディスクモジュール間の並列動作を妨げる．そのため，この二つの競合ができるだけ小さくなるようにスケジュールを行う必要がある．

図2(b)は $DM2$ 上時刻0で $r3(C)$，時刻1で $w3(A)$ を実行した場合のガント図である．直列可能順は{$T1$, $T3 \rightarrow T4$, $T3 \rightarrow T2$}となる．この順序の方が図2(a)の場合よりも競合が小さく，ディスクモジュールが並列に動作できる．スケジューラはこうした直列可能順を予測し，この順に従うロック要求のみを認めるべきである．

3. アプローチ

3.1 スケジュール戦略

スケジューリングの理論では，本論文の問題は直列可能性制約のもとでの一般ジョブショップの動的スケジューリングである[7]．我々は，スケジューラにデータ競合・リース競合のより少ない直列可能順を予想させ，この順序に従うロック要求のみを認める戦略スケジュールを行う．この言わば「望ましい」直列可能順を予想するため，重み付き辺をもつトランザクション順序グラフ(Weighted Transaction Precedence Graph，以下WTPG)を定義する．

[定義1]　WTPGは次のグラフ $\langle N, E, C, w \rangle$ である．($T0$ は初期トランザクション，Tf は最終トランザクション，Ti, Tj はその他一般のトランザクションを表す．)

①　N：トランザクションを表すノード集合．

②　E：順序辺 $Ti \rightarrow Tj$(「Ti は Tj より直列可能順で先行する」を表す)の集合．次に示す選択辺(Ti, Tj)を置き換えることによってのみ生成される．各 Ti に対し，$T0 \rightarrow Ti$，また $Ti \rightarrow Tf$ が成立する．

③　C：選択辺(Ti, Tj)の集合．(Ti, Tj)は辺 $Ti \rightarrow Tj$, $Tj \rightarrow Ti$ の対であり，Ti と Tj が直列可能順で競合することを示す．辺(Ti, Tj)は，Ti と Tj があるデータに競合するアクセスを宣言した際に生成される．Ti が Tj に直列可能順で先行することが決まれば，E の元 $Ti \rightarrow Tj$ に置き換えられる．この操作を選択辺の解消と呼ぶ．

選択辺(Ti, Tj)，順序辺 $Ti \rightarrow Tj$ は Ti と Tj があるデータに競合するアクセスを行うときのみ生成される．

④ $w: E \cup C$ の辺 $Ti \rightarrow Tj$ に対する重み $w(Ti \rightarrow Tj)$ を与える関数であり，次のように重みを決める．

・$w(T0 \rightarrow Ti)$: 現時刻 $t0$ から Ti が最も早くコミットする場合の時刻 $t1$ までの時間 $t1-t0$．Ti の最早コミット時間または準備時間と呼ぶ．時刻 $t1$ は，時刻 $t0$ 以後 Ti のステップがブロックされず，かつ，実行可となった際には，現時刻 $t0$ で動作中のステップの次にすぐ実行されると仮定して計算する．

・$w(Ti \rightarrow Tf)$: Ti のコミットからその終了までの時間．

・$w(Ti \rightarrow Tj)$: Ti がコミットした時刻 $t2$ から，Tj が最も早くコミットする場合の時刻 $t3$ までの時間 $t3-t2$．但し Tj は，Ti と競合していたステップの実行を始めた後にはブロックされず，また，その実行可能なステップは DM のふくそうによって待たされることはなく，すぐに実行されると仮定する． □

$w(T0 \rightarrow Ti)$ は各 DM のふくそうと Ti のデータアクセス系列とから計算され，スケジュールの進行に伴い更新される．$w(Ti \rightarrow Tj)$ は Ti と Tj の両方が宣言された際に DM のふくそうとは無関係にそのデータアクセス系列のみから計算され，以後更新されない．

[例] 図3は図2(a)，時刻0で $r1(D)$ が実行開始した後の WTPG である．簡単のため $T1$ は省略する．辺 $T0 \rightarrow T2$ は $T2$ の最早コミット時間8を重みとしてもつ: $r2(E)$ が $r1(D)$ の終了後 $DM1$ 上で時刻4に開始し次ステップ $w2(A)$ が時刻7で開始する場合に，$T2$ は最も早く終了するからである．辺 $T2 \rightarrow T3$ の重みは $w3(A)$，$w3(C)$ のコスト和2となる: $T3$ は，$T2$ がコミットし競合するデータ A の占有ロックを解放後に $w3(A)$ を実行開始するからである．**2.**のコストモデルの仮定より，すべての辺 $Ti \rightarrow Tf$ は重み0をもつ． □

WTPG の全ノードからなる直列可能順（全直列可能順と呼ぶ）が与えられると，WTPG のすべての選択辺は解消される．このとき $T0$ から Tf への臨界経路長は，この全直列可能順に従うスケジュールの最早終了時間である．

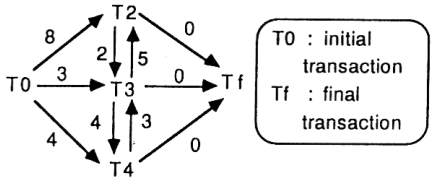


図3 WTPG の例
Fig. 3 Example of WTPG.

我々の戦略は，この臨界経路が最短になるように WTPG の選択辺を解消する全直列可能順 W を計算し，W に矛盾しないロック要求のみを認めることである．これによってデータ競合・リソース競合がより小さくなるように直列可能スケジュールを生成できる．ブロック連鎖は臨界経路を長くし，ディスクのふくそうはトランザクションの準備時間を長くするからである．

[例] 図3では $W=\{T3 \rightarrow T4,\ T3 \rightarrow T2\}$ が最短臨界経路を作る全直列可能順である．図4(b)が W によって（その選択辺を）解消された WTPG であり，その臨界経路は $\{T0 \rightarrow T3 \rightarrow T2\}$，長さ8となる．**2.3** の C2PL スケジューラは直列可能順 $\{T1,\ T2 \rightarrow T3 \rightarrow T4\}$ を作り，その臨界経路は $T0 \rightarrow T2 \rightarrow T3 \rightarrow T4$，長さ14となる（図4(a)）．スケジューラは，$W$ に従うロック要求のみを認めることで $T3$ のステップ $r3(C)$，$w3(A)$ を $r2(A)$，$w4(C)$ よりも優先して実行する． □

3.2 WTPG による先読みスケジューラ

WTPG による先読みスケジューラ CC は以下の手順で動作する．

〈手順0〉 あるディスクモジュール DM_i がアイドルになり CC を呼び起こす．

〈手順1〉 現在の WTPG において，最短臨界経路を生成する全直列可能順 W を計算する．

〈手順2〉 DM_i の Ready Queue にあるステップのうち，ブロックされず，かつ W に矛盾する直列可能順序を生成しないステップ q を探す．

〈手順3〉 q があればそのロックを認め DM 上で実行する．このとき WTPG 上の辺 $T0 \rightarrow Ti$（i は任意）の重みを必要なら更新する．

〈手順4〉 q がなければ DM_i は指定時間後に再び CC を呼び起こす． □

新しくトランザクション T が到着したとき WTPG

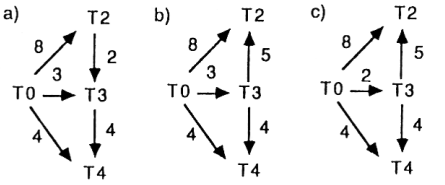


図4 時刻0での WTPG
Fig. 4 WTPG resolved by (a) C2PL at clock 0 (b) the shortest critical path at clock0 (c) the shortest critical path at clock1.

内でその選択辺の重みを計算する必要がある．そのため，Tはその開始時に全読出し・書込み集合と共にそのためのロックの「残り時間」を宣言しなければならない．あるロックの「残り時間」とは，Tがそのロックをとった時刻から最も早くコミットする場合の時刻までの時間である（DMのふくそうは無視する）．これはTのデータアクセス系列のみから計算できる．

[例] 図2(b)は上のスケジューラによるスケジュールである．前提は図2(a)のそれと同じである．図2(b)で時刻0，$r1(D)$が開始された後，最短臨界経路を作る直列可能順Wは図4(b)に示した．時刻0，$DM2$において$r2(A)$を認めた場合，それによって直列可能順$\{T2 \rightarrow T3\}$が生成され，Wに矛盾する．そのため，$r2(A)$はロックは認められず遅延される．代わりに$r3(C)$が制約$\{T3 \rightarrow T4\}$がWに従うため実行される．時刻1で$T3$の準備時間は変更されWTPGは図4(c)のようになる．ここでもWは変わらない．故に$r2(A)$は再び遅延され$w3(A)$が$DM2$上で実行される． □

4. 鎖型WTPGによるスケジューラ

4.1 鎖型WTPGの最短臨界経路計算法

3.のスケジューラでは，WTPGにおいて最短臨界経路を作る直列可能順を計算する必要がある．以下，経路は$T0$からTfへの経路を意味する．この問題は任意のWTPGではNP困難である（付録参照）．ここではWTPGを次に示す部分クラスに限定しこの問題の厳密解を求める$O(n^2)$算法を述べる．以下，$n(k)$はラベルkのノードを表し，$n(1)$，$n(2)$等は$n1$，$n2$と表記する．

[定義2] 鎖型WTPGとは$T0$，Tf以外のノードに次のようにラベル1，2，…，Nを付けることができるWTPGである：$n(k)$（但し$2 \leqq k \leqq N-1$）はたかだか$n(k-1)$または$n(k+1)$とのみ直列可能順で競合する． □

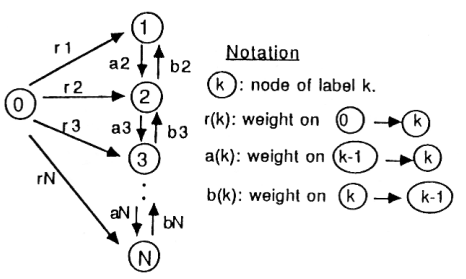


図5 鎖型WTPG $G(1, N)$
Fig. 5 Chain-form WTPG $G(1, N)$.

図5はノード$n1$，$n2$，…，$n(N)$からなる鎖型WTPG $G(1, N)$であり，$n(k)$が必ず$n(k-1)$, $n(k)$と競合している場合である．$n0$は初期トランザクションであり最終トランザクションは以後省略する．定義2では$n(k)$は$n(k-1)$，$n(k+1)$の両方と必ず競合するとは限らないが，簡単のため以後，図5の場合のみを考える．また，ある選択辺$(n(k), n(k+1))$が$n(k) \rightarrow n(k+1)$（または$n(k) \leftarrow n(k+1)$）に既に解消されている場合には，これと対の辺の重み$b(k+1)$（または$a(k+1)$）を無限大として扱う．

以下，$G(i, j)$は$n0$と$\{n(i), n(i+1), \cdots, n(j)\}$からなる$G(1, N)$の部分グラフを表す．また，選択辺$(n(i), n(i+1))$が$n(i+1) \rightarrow n(i)$に解消されるときこれを「上向きに解消する」と言い，$n(i) \rightarrow n(i+1)$に解消するときは「下向きに解消する」と言う．以下に定義するパラメータ$L(k)$，$R(k)$を計算することで最短臨界経路が計算される．

[定義3] $L(k)$，$R(k)$は各々次に定義される三つ組$(curr, crit, rev)$である（各メンバは$L(k).curr$のように〈パラメータ名〉.〈メンバ名〉で表記する）．

今，$G(k-1, N)$において選択辺$(n(k-1), n(k))$が下向きに解消されたとする．このとき，この$G(k-1, N)$において最短臨界経路を作る直列可能順を$S1(k-1, N)$としよう．$L(k)$は辺$n(k-1) \rightarrow n(k)$上で次のように定義される．

・$L(k).crit$：$S1(k-1, N)$によって選択辺を解消された$G(k-1, N)$における最短臨界経路長．

・$L(k).rev$：$S1(k-1, N)$において選択辺$(n(i), n(i+1))$が上向きに解消されているラベルiのうち，最小のラベル．

・$L(k).curr$：経路$n0 \rightarrow n(k-1) \rightarrow n(k) \rightarrow \cdots \rightarrow n(L(k).rev-1) \rightarrow n(L(k).rev)$の長さ．

選択辺$(n(k-1), n(k))$が上向きに解消された$G(k-1, N)$において，その最短臨界経路を作る直列可能順を$S2(k-1, N)$とする．$R(k)$は辺$(n(k) \rightarrow n(k-1)$上で次のように定義される．

・$R(k).crit$：$S2(k-1, N)$によって選択辺を解消された$G(k-1, N)$における最短臨界経路長．

・$R(k).rev$：$S2(k-1, N)$において選択辺$(n(i), n(i+1))$が下向きに解消されているラベルiのうち，最小のラベル．

・$R(k).curr$：$S2(k-1, N)$により選択辺を解消された$G(k-1, R(k).rev)$における$n0$から$n(k-1)$への臨界経路長． □

[例1] 図6(a)は鎖型WTPG $G(2,\ 4)$ と上で定義された $L(4)$, $R(4)$ を示す．図6(b)に $L(3)$ を計算する場合を示す．このときまず選択辺 $(n2,\ n3)$ が下向きに解消される．すると $n3 \rightarrow n4$ と $n4 \rightarrow n3$ のうち，前者の方が $G(2,\ 4)$ でより短い臨界経路を作る．従って $L(3).curr$ と $L(3).crit$ を構成する経路は $n0 \rightarrow n2 \rightarrow n3 \rightarrow n4$ であり，$S1(2,\ 4)=\{n2 \rightarrow n3 \rightarrow n4\}$, $L(3)=(8,\ 8,\ n4)$ となる．$R(3)$ の場合も同様にして $S2(2,\ 4)=\{n3 \rightarrow n2,\ n3 \rightarrow n4\}$ が最短臨界経路 $n0 \rightarrow n3 \rightarrow n4$ を生成する(図6(c))． □

$L(k)$, $R(k)$ の定義より明らかに次の定理が成立する．

[定理1] $G(k,\ N)$ において $L(i)$, $R(i)$ (但し $i=k+1, \cdots, N$) が与えられたとする．このとき $S1(k, N)$, $S2(k, N)$, また $G(k, N)$ で最短臨界経路 P を作る直列可能順 $S(k, N)$ は次式で計算できる．

$S1(k, N)$
$=\{n(k) \rightarrow n(k+1) \rightarrow \cdots \rightarrow n(L(k+1).rev)\}$
$\cup S2(L(k+1).rev,\ N)$

$S2(k, N)$
$=\{n(k) \leftarrow n(k+1) \leftarrow \cdots \leftarrow n(R(k+1).rev)\}$
$\cup S1(R(k+1).rev, N)$

但し $S1(k, k)=S2(k, k)=\phi$ for all k.

P の長さ $=\min(L(k+1).crit,\ R(k+1).crit)$

If $L(k+1).crit \leqq R(k+1).crit$ then
$S(k, N)=S1(k, N)$;
else
$S(k, N)=S2(k, N)$; □

[例2] 例1 (図6(a)) では $L(3).crit=8>R(3).crit$ より $S(2,\ 4)=S2(2, 4)=\{n2 \leftarrow n3\} \cup S1(3, 4)=\{n2 \leftarrow n3 \rightarrow n4\} \cup S2(4, 4)=\{n2 \leftarrow n3 \rightarrow n4\}$ □

[定理2] $G(k-1, N)$ (図5) において $L(i)$, $R(i)$ (但し $i=k+1, k+2, \cdots, N$) が与えられたとする．このとき $L(k)$, $R(k)$ は $O(N-k)$ で計算される． □

(略証) $L(k)$ を計算する算法 Lcomp を以下にC言語風の記法で示す．ここで変数 $a(i)$, $b(i)$, $r(i)$ は図5の $G(1, N)$ の各辺の重みである．

```
Lcomp(  ){
  /*procedure for L1(k) */
temp=L(k+1).curr−r(k)+r(k−1)+a(k);
if  (temp<=L(k+1).crit)
  L1(k)=(temp, L(k+1).crit, L(k+1).rev);
else{/*temp>L(k+1).crit*/
  L1(k).crit
    =min(max(V(h), R(h+1).crit)); 式(1)
      for all h=k+1 to L(k+1).rev
  L1(k).rev=h0;
  L1(k).curr=C(h0);
/*     h0 は上の式(1)で最小値をとる h の値.
 *     変数 C(h), V(h) は次式で計算される.
 * V(k−1)=C(k−1)=r(k−1);
 * V(h)=max(r(h),  V(h−1)+a(h)); (k=<h)
 * C(h)=C(h−1)+a(h); (k=<h)
 */
}  /*end of L1(k)*/
    /*procedure for L2(k): */
L2(k).curr=r(k−1)+a(k);
L2(k).crit=max(L2(k).curr,  R(k+1).crit);
L2(k).rev=k; /*end of L2(k)*/
if (L1(k).crit<=L2(k).crit)
    L(k)=L1(k);
else
    L(k)=L2(k);
}
```

算法 Lcomp 中，$L(k)$ は $L1(k)$ と $L2(k)$ の $crit$ の小さい方の値である．ここで，$L1(k)$ は辺 $(n(k),\ n(k+1))$ が下向きに解消されたときの $L(k)$ の値であり，$L2(k)$ は上向きに解消されたときの $L(k)$ である．$L1(k)$, $L2(k)$ はそれぞれ最初は直列可能順 $\{n(k-1) \rightarrow n(k)\} \cup S1(k,\ N)$, および $\{n(k-1) \rightarrow n(k)\} \cup S2(k, N)$ によって $G(k-1,\ N)$ の選択辺を解消した場合を考えればよい．

$L1(k)$ を計算する場合，辺 $n(k-1) \rightarrow n(k)$ を加えることで新しく経路 $P0=n0 \rightarrow n(k-1) \rightarrow n(k) \rightarrow n(k+1) \rightarrow \cdots \rightarrow n(L(k+1).rev)$ が生成される．算法 Lcomp 中，変数 temp が $P0$ の長さである．

$P0$ が $G(k, N)$ の臨界経路長 $L(k+1).crit$ より長い場合，直列可能順 $\{n(k-1) \rightarrow n(k)\} \cup S1(k, N)$ では $P0$ が $G(k-1, N)$ での臨界経路である．しかし $P0$ を

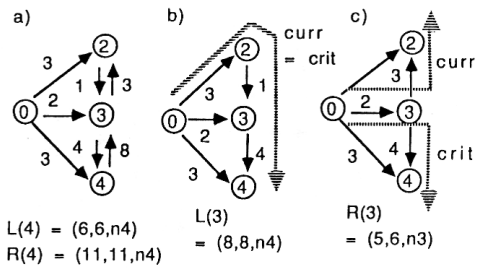


図6 $L(k)$, $R(k)$ の例
Fig. 6 Example of $L(k)$ and $R(k)$.

構成するある選択辺 $(n(h), n(h+1))$ を上向きに解消すれば $P0$ よりも臨界経路が短くなる場合がある．

今，h を $G(k-1, N)$ において辺 $(n(h), n(h+1))$ が上向きに解消されている最小ラベルとしよう．すなわち，直列可能順 $S(h)=\{n(k-1)\rightarrow n(k)\rightarrow \cdots \rightarrow n(h)\} \cup S2(h, N)$ を想定する（但し，$k+1 \leqq h \leqq L(k+1).rev$）．Lcomp において変数 $C(h)$ は経路 $P(h)=\{n0 \rightarrow n(k-1)\rightarrow n(k)\rightarrow \cdots \rightarrow n(h)\}$ の長さであり，$V(h)$ は $P(h)$ で解消された $G(k-1, h)$ の臨界経路長である．

$S(h)$ では辺 $(n(h), n(h+1))$ が上向きに解消されているため $S2(h, N)$ が定義3より $G(h, N)$ の最短臨界経路を作る．このとき $G(k-1, h)$ と $G(h, N)$ とは連結していない．故に $G(k-1, N)$ の臨界経路は両者のそれの長い方，すなわち $\max(V(h), R(h+1).crit)$ となる．

算法 Lcomp( ) のうち，式(1)は h を $k+1$ から $L(k+1).rev$ まで変えて $G(k-1, N)$ の臨界経路を最短にする $h=h0$ を求める．$V(h)$，$C(h)$ は式(1)のループ内で $O(1)$ で計算されるため $L1(k)$ は $O(N-k)$ で正しく計算される．これ以外の $L1(k)$，$L2(k)$ の計算はその定義により自明であり省略する．$R(k)$ の場合は付録補題1に示した． □

$L(k)$，$R(k)$ を $k=N$ から $k=2$ まで定理2より計算することで $L(2)$，$R(2)$ が求まる．従って定理1より次の系が成立する．

[系1] 鎖型 WTPG $G(1, N)$（図5）の最短臨界経路を作る直列可能順は $O(N^2)$ で計算できる．

[例3] 図7(a)は図6(b)の WTPG にノード $n1$ とその辺 $n0 \rightarrow n1$ 重み5，$n1 \rightarrow n2$ 重み10を加えたものに対し $L1(2)$ を計算する場合を示す．$S1(2, 4)$ で解消された $G(2, 4)$（図6(b)）に辺 $n1 \rightarrow n2$ を加えると $P=n0 \rightarrow n1 \rightarrow n2 \rightarrow n3 \rightarrow n4$，長さ20の経路が $G(1, 4)$ の臨界経路である．しかし，P を構成する選択辺 $(n3, n4)$ を上向きに解消すれば $G(1, 4)$ の臨界経路を更に短くできる（図7(b)）．この場合臨界経路長は $\max(|n0 \rightarrow n1 \rightarrow n2 \rightarrow n3|, R(4).crit)=16$ であり，P よりも短い．故に $L1(2)$ は $(16, 16, n3)$ となる． □

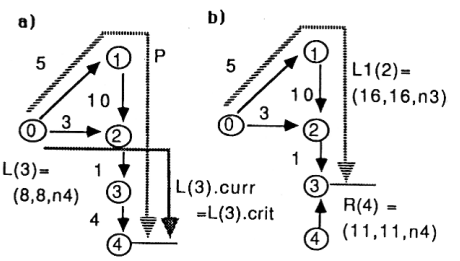


図7 $L1(k)$ の計算例
Fig. 7 Example of the algorithm for $L1(k)$.

4.2 実装に関する問題点

鎖型 WTPG によるスケジューラを実装する際，次の2点が必要である．

① 新しくトランザクション T を開始する際，T を追加した WTPG が鎖型か否か検査する．これは深さ優先探索で計算できる．鎖型でなければコミットかアボートが起きるまで T はブロックされる．

② スケジュールの進行に際し，WTPG の各パラメータ $w(T0 \rightarrow T)$ を更新する．これは各 DM 上で T がコミットするまでの残り時間を追跡することで，各 DM のふくそうの更新に対し $O(1)$ で更新できる．

5. 性能評価

5.1 シミュレーションモデル

本節では図2のようにガント図を生成するシミュレータを用いて性能評価を行う．シミュレーションの1クロックは **2.2** に述べた1単位時間である．

パラメータとして次の値を用いる．

Numdisk：ディスクモジュールの総数．

Numdata：区画の総数．

λ：指数分布における到着率．

使用した規約は先読み2相ロック[6]（C2PL），一括ロック（Atomic Static Lock，ASL，文献(2)），楽観的ロック[9]（Optimistic Lock，OPT）と鎖型 WTPG によるスケジューラ（CHAIN）である．また性能の上限を出すためいつどんなロック要求でも認める規約 NONE も使う．

ASL では，トランザクションは開始時にすべてのアクセスデータ集合にロックを要求し，これに失敗すると一定遅延後に再実行される．OPT では，トランザクションはそのコミット時に直列可能検査を行い，これに失敗すればアボートされ再実行される．使用する規約のうち，ASL，CHAIN，C2PL はアボートが起きないことに注意してほしい．

性能評価の基準としては飽和点での処理性能 θ を用いる．これは1シミュレーションクロック当り終了するトランザクション数である．θ は，システムの多重度を無限大として到着率 λ を変えて測定し，処理性能が λ の 90 % となるときの値とする．測定は定常状態に達してから 1000 クロック区間の測定値である．

パラメータ値としてはNumdisk＝8，Numdata＝24とした．8区画で1リレーションを表しており，番号iの区画は番号＝<i modulo Numdisk＞のディスクモジュールに配置される．このNumdisk，Numdataの値は従来の短時間トランザクションで用いられてきたものよりはるかに小さく[3]，極端に高いデータ競合とリソース競合を引き起こす．この値を用いる理由は，BATのアクセスするデータ量はデータベースの増大に比例すると考えるからである．この値では，BATの1レンジ選択が1 relationの1/8にアクセスする場合を仮定している．このとき，たとえDMがn個あったとしても，そのn/8個は1ステップを並列に実行し，$7n/8$個が他のステップを実行する．そのためNumdisk，Numdataの実質的な値はここで用いるものと同じである．

一般にデータ競合にはブロック連鎖と，アクセスが集中する一部のデータ上の長時間ロックによる性能低下（ホットセット競合）とが挙げられる．次のトランザクションの読出し・書込み系列（パターンと呼ぶ）はこのどちらかを少なくともつ．各ステップは<r/w>（アクセスする区画，このステップのコスト）と表記する．

[パターン1] 系列$r(F1:1)\rightarrow r(F2:5)\rightarrow w(F1:0.2)\rightarrow w(F2:1)$．はじめ二つの読出し時に占有ロックを要求する．これは，区画$F1$の20％選択の後に$F2$とのjoin演算を行い，その結果に基づいて，これら読み出したデータ量の10％を更新する処理をモデル化している．$F1$，$F2$は5単位サイズとし，24区画からランダムに選んだ．joinを行う区画を直列に読み出す動作は，文献(4)のhy-brid hash join算法に基づく．このパターンでははじめの2ステップでブロック連鎖が発生する．

[パターン2] 系列$r(B1:1)\rightarrow r(B2:2)\rightarrow r(B3:2)\rightarrow w(F1:1)\rightarrow w(F2:1)$．このパターンは，区画$B1$，$B2$，$B3$のjoinの結果に基づいて区画F1，F2を更新する処理をモデル化している．$B1$，$B2$，$B3$は2単位サイズの区画8個（読出し専用の1リレーションを表し，Numdisk個のディスクに1区画ずつ配置される）からランダムにそれぞれ選び，$B1$は50％，$B2$と$B3$は100％読出しとする．$F1$，$F2$は1単位サイズの区画16個からランダムにとりその50％を更新する．この16区画上でホットセット競合が起こる．

[パターン3] 系列$r(B:4)\rightarrow w(F1:1)\rightarrow w(F2:4)$．$B$は読出し専用の4単位サイズの区画8個（1区画ずつ各DMに配置する）からランダムに選ぶ．$F1$，$F2$は4単位サイズの16区画からとり，各々1/8，1/2を更新する．このパターンは書込み時にブロック連鎖とホットセット競合を両方引き起こす．

表1 飽和点での処理性能

	NONE	ASL	C2PL	CHAIN	OPT
パターン1	1.01	0.81	0.39	0.80	0.29
パターン2	1.06	0.66	0.89	0.90	0.69
パターン3	0.82	0.46	0.40	0.63	0.40

5.2 実験結果と議論

5.1の各トランザクションパターンについて飽和点での処理性能を表1に示す．

パターン1ではASLがブロックを起こさないため最も処理性能が高くなる．表1ではCHAINはASLとほぼ同じ性能であることから，ブロック連鎖を完全に回避できることがわかる．C2PLではブロック連鎖のためCHAIN，ASLに対し1/2の性能しか得られていない．この実験ではOPTの場合，リソース競合が飽和点付近で95％以上と高くなった．これより，高いデータ競合はアボート率を引き上げるため，結局はディスクの有効稼働率が低くなり低い処理性能しか得られてないことがわかる．

パターン2では，実際に並行に動くトランザクション数（有効多重度）の多い規約ほど読出しステップが並列に動作でき処理性能が高くなっている（表1）．ASLでは区画1個に対してトランザクション1個しか並行に動けないため，ホットセットがあると有効多重度が大きく低下する．これに対してCHAINの有効多重度は鎖型の制約によりASLに対して最大2倍になる．kノードからなる鎖型WTPG 1個は$k/2$個の孤立点をもつからである．更にC2PL，OPTでは有効多重度への制約はない．表1より，ホットセット上でデータ競合が起きると，この有効多重度の差が処理性能に大きく影響することがわかる．OPTではパターン1のときと同じく高いデータ競合がアボート率を高くし，ASLと同じ性能しかでていない．

パターン3ではC2PL，ASLともにブロック連鎖・ホットセット競合のいずれかを被る．一方，CHAINはこの両方を回避でき，C2PL，ASLよりも高い処理性能を達成している（表1）．

6. むすび

本論文では大量データアクセストランザクション（BAT）の並行制御方式として，重み付きトランザクショ

ン順序グラフ（WTPG）による鎖型WTPGスケジューラを提案した．BAT処理ではデータ競合とリソース競合が非常に高くなるためこれらの競合が小さくなるようにBATのスケジュールを行う必要がある．鎖型WTPGスケジューラではWTPGを用いてこれらの競合ができるだけ小さくなる直列可能順序が計算され，この順序に従うロック要求のみが認められる．このとき，多項式時間でこの順序を計算するためにWTPGのトポロジーが鎖型に限定されている．従って鎖型にWTPGを保てない場合，新しいBATの開始は遅延される．ガント図上でのシミュレーションでは鎖型WTPGスケジューラは2相ロック，一括ロック，楽観的ロック規約に対し1.3倍から2倍の性能向上が得られた．

今後の課題として，コスト予測の正確さに対する性能依存度や制御負荷を評価していく必要がある．また鎖型へのWTPGの制約は一括ロックとの性能比を抑えているため，任意トポロジーのWTPGを用いた別戦略によるスケジューラを検討する必要がある．

文　献

(1) J. Gray : "The transaction concept : Virtues and limitations", Proc. 7th Int'l Conf. Very Large Data Bases, pp. 144-154 (1981).

(2) Y. Tay, N. Goodman and R. Suri : "Locking performance in centralized databases", ACM Trans. Database Syst., **10**, 4, pp. 415-462 (1985).

(3) A. Bhide : "An analysis of three transaction processing architectures", Proc. 14th Int'l Conf. Very Large Data Bases, pp. 339-350 (1988).

(4) D. DeWitt, R. Gerber, G. Graefe, M. Heytens, K. Kumar and M. Muralikrishna : "GAMMA－A high performance dataflow database machine", Proc. 12th Int'l Conf. Very Large Data Bases, pp. 228-237 (1986).

(5) G. Copeland, W. Allexander, E. Boughter and T. Keller : "Data placement in Bubba", Proc. ACM -SIGMOD'88, pp. 99-108 (1988).

(6) S. Nishio, M. Watanabe, Y. Ohiwa and T. Hasegawa : "Performance evaluation on several cautious schedulers for database concurrency control", Proc. 5th Int'l Workshop Database Machines, pp. 212-225 (1987).

(7) R. Graham, E. Lawler, J. Lenstra and A. Rinnooy Kan : "Optimization and Approximation in Deterministic Sequencing and Scheduling : A Survey", in Annals of Discrete Mathematics, 5, pp. 287-326, North Holland (1979).

(8) R. Agrawal, M. Carey and M. Livny : "Models for studying concurrency control performance : Alternatives and implications", Proc. ACM -SIGMOD'85, pp. 108-121 (1985).

(9) H. T. Kung and J. T. Robinson : "On optimistic methods for concurrency control", ACM Trans. Database Syst., **6**, 2, pp. 213-226 (1981).

付　録

[定理3] 任意のWTPGの最短臨界経路を作る直列可能順の計算問題SSはNP困難である．

(略証) 一般ジョブショップの静的スケジューリング問題JSは，disjunctive graphで初期ノードから最終ノードへの臨界経路最短とするdisjunctive edgeの解消問題である（文献(7)参照）．このグラフは，各ノードの出辺にそのノードのコストを与えたWTPGにほかならない．故に問題SSは問題JSを含んでおり，JSがNP困難なのでSSもNP困難である． □

[補題1] 定理2で$R(k)$は$O(N-k)$で計算される．

(略証) $R(k)$を計算する算法Rcompを以下にC言語風の記法で示す．変数$r(i)$，$b(i)$はすべて図5に示したもの．

```
Rcomp(  ){
/*procedure for R1(k)*/
temp=R(k+1).curr+b(k);
if  (max(r(k-1), temp)=<R(k+1).crit)
   R1(k)=(max(r(k-1),temp, R(k+1).crit,
   R(k+1).rev);
else if  (max(r(k-1),  temp)==r(k-1))
   R1(k)=(r(k-1),  r(k-1),  R(k+1).rev);
else{/*temp>R(k+1).crit*/
   R1(k).crit
   =min(max( V(h),  L(h+1).crit));式(2)
     for all h=k+1 to R(k+1).rev
   R1(k).rev=h0;
   R1(k).curr=V(h0);
/*   h0は上の式(2)で最小値をとるhの値.
 *   V(h),  C(h)は下に示した.          */
}/*end of R1(k)*/
/*   procedure for R2(k)*/
R2(k).curr=max(r(k)+b(k),  r(k-1));
R2(k).crit=max(R2(k).curr,  L(k+1).crit);
R2(k).rev=k;     /*end of R2(k)   */
if (R1(k).crit<=R2(k).crit)
     R(k)=R1(k);
else
     R(k)=R2(k);
}
```

ここで変数$V(h)$は$G(k-1,\ h)$においてすべての選択辺が上向きに解消された際の$n0$から$(k-1)$への臨

界経路長であり，$C(h)$はこのときの経路 $n0 \rightarrow n(h) \rightarrow n(h-1) \cdots \rightarrow n(k+1) \rightarrow n(k) \rightarrow n(k-1)$ の長さである．$V(h)$，$C(h)$は次式で計算される．

$V(k-1)=C(k-1)=r(k-1)$;

$C(h)=C(h-1)-r(h-1)+r(h)+b(h)$;

但し $k \leqq h$.

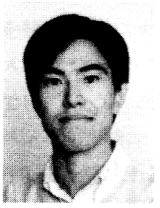
$V(h)=\max(C(h),\ V(h-1))$; 但し $k \leqq h$.

$R1(k)$は辺 $(n(k),\ n(k+1))$が上向きにされた場合の$R(k)$の値であり，$R2(k)$は下向きにされた場合のそれである．$R(k)$は$R1(k)$，$R2(k)$のうち，$crit$の値の小さい方となる．

$R1(k)$の場合，$S2(k,\ N)$で解消された$G(k,\ N)$に辺 $n(k) \rightarrow n(k-1)$を追加した場合のみ考えればよい．このとき，経路 $n0 \rightarrow n(k-1)$が新しく臨界経路となるならこれは短くできない．

$G(k,\ N)$内の経路と辺 $n(k) \rightarrow n(k-1)$とからなる経路Pが新しく臨界経路となる場合，Pを構成する選択辺$(h,\ h+1)$を下向きにすれば$G(k-1,\ N)$の臨界経路を短くできる場合がある．式(2)はhを$k+1$から$R(k+1).rev$まで変えてこの場合を$O(N-k)$で計算する．これ以外の$R1(k)$，$R2(k)$は自明であり，省略する． □

（平成元年3月31日受付，8月18日再受付）



大森　匡

昭60東大・工・電気卒．昭62同大大学院情報工学専攻修士課程了，工学修士．現在，同大・後期博士課程在籍．トランザクション管理，ファイル編成法，演えきデータベース，ルールベースなどを研究している．情報処理学会会員．

喜連川　優

昭53東大・工・電子卒．昭58同大大学院情報工学博士課程了．工博．昭59東京大学生産技術研究所助教授．昭60情報処理学会学術奨励賞．並列コンピュータアーキテクチャ，データベース工学の研究をしている．「第5世代コンピュータ」，「Database Machines and Knowledge Base Machines」(共著)．IEEE，ACM，情報処理学会各会員．

田中　英彦

昭40東大・工・電子卒．昭45同大大学院博士課程了．工博．東大講師，助教授，ニューヨーク市立大学客員教授を経て昭62より東大工学部教授．計算機アーキテクチャ，並列推論マシン，分散処理などの研究を行っている．「計算機アーキテクチャ」，「VLSIコンピュータI，II」(いずれも共著)，「情報通信システム」著．情報処理学会，日本ソフトウエア科学会，人工知能学会，IEEE，ACM各会員．