トランザクションの優先度を考慮した時刻印方式による並行処理制御

7D－9

足高 正訓　中村 素典　大久保 英嗣
立命館大学理工学部情報学科

1 はじめに

トランザクションの並行処理制御方式の1つに時刻印方式がある．この方式は，2相ロック方式のようにデッドロックを起こすことはないが，トランザクションのアボート率が高いことが問題とされている．本稿で提案する優先度付き時刻印方式 (Priority based Timestamp Ordering, 以下 PTO と記す) は，その問題点を解消するものである．PTO は多重版時刻印方式を基本とし，並行実行されているトランザクションが直列可能であるための条件を見直すことによって，トランザクションのコミット率を向上させることを可能にしている．また同時に PTO は，トランザクションの優先度を考慮した並行処理制御も行なう．これによりトランザクションのコミット制御に関して，よりユーザの意図を反映した処理が可能となり，さらにシステムの性能向上も期待できる．

2 処理方式

多重版時刻印方式では，主に遅い更新 (delayed write) と呼ばれる操作要求を行なった時にアボートが発生する．これは，時刻印の順で直列化が可能でなくてはならないという時刻印方式の規則に基づくものである．しかし実際には，遅い更新が起きても直列化が可能となる場合は存在する．そこで PTO は遅い更新による競合発生時に，トランザクションの優先度を基にアボート対象を決定するとともに，時刻印の順以外での直列化の可能性を解析し，不必要なアボートを阻止している．

以降，PTO の処理方式を述べる際に使用する記号を次のように定義する．

- $TS(TR_n)$: トランザクション n の開始時刻印
- $P(TR_n)$: トランザクション n の優先度
- DW : 遅い更新を発生させたトランザクション
- $\mathcal{PR}$: 競合したオブジェクトを先に参照した後発トランザクションの集合
- $\mathcal{PR}_c$: コミット済の $\mathcal{PR}$
- $\mathcal{PR}_u$: まだコミットしていない $\mathcal{PR}$

(このとき $TS(DW) < min\{TS(\mathcal{PR})\}$)

PTO は，多重版を管理していることを前提とし，トランザクション間に競合が起きていない間は，従来型多重版時刻印方式と同様の処理を行う．オブジェクトの競合，いわゆる遅い更新が発生した場合に行われる処理が異なっている．

遅い更新が発生した場合，具体的には以下の3つの場合に分けて処理を行なう．

(1) $\mathcal{PR}$ の要素すべてが未完了の場合．
- $max\{P(\mathcal{PR})\} \geq P(DW)$ なら DW をアボートする．
- $max\{P(\mathcal{PR})\} < P(DW)$ なら $\mathcal{PR}$ をアボートする．

(2) $\mathcal{PR}$ の要素すべてが既にコミットを完了している場合．

まず DW が今までにアクセスしたオブジェクトのログを調べる．その結果，それらのオブジェクトすべてに $TS(DW)$ 以降のアクセス記録がない場合 (ただし Read-Read の関係を除く) は以下の処理を行い，アボートは回避できる．条件を満たさない場合は DW がアボートすることになる．アボートを回避できた DW は，以後通常のトランザクションと同様の扱いを受け，いかなる特別な制約も受けない．

1．$TS(DW)$ を現在の時刻印に付け替える．
2．DW が今までに行った操作のログ (オブジェクトのアクセス時刻) を現時刻に書き換える．

Concurrency Control by Priority based Timestamp Ordering.
Masanori Adaka, Motonori Nakamura, Eiji Okubo
Department of Computer Science, Ritsumeikan University

(3) $\mathcal{PR}$ に，既にコミットしたものとまだコミットしていないものが含まれている場合．
· $max\{P(\mathcal{PR}_u)\} \geq P(DW)$ なら DW をアボートする．
· $max\{P(\mathcal{PR}_u)\} < P(DW)$ ならば DW と $\mathcal{PR}_c$ の間で (2) の検査を行う．条件を満たしている場合は (2) と同様の処理を行ない，$\mathcal{PR}_u$ をアボートする．条件を満たさない場合は DW をアボートする．

3 優先度の継承

多重版時刻印方式では，まだコミットしていないトランザクションが更新したオブジェクトに対して参照要求があったときの処理に，保守的時刻印方式と積極的時刻印方式と呼ばれる 2 つの方式を選択することができる．2 つのうち積極的時刻印方式においては，アボートしたトランザクションが作成したバージョンを先読みしていたトランザクションが連鎖的にアボートするというカスケーディングアボートが発生する可能性がある．

PTO を，積極的時刻印方式と併用した場合には，これによる優先度逆転現象に注意しなくてはならない．例えば $P(TR_1) < P(TR_2) < P(TR_3)$ のような 3 つのトランザクションがあるとする．TR_1 が更新を行なったバージョンを TR_3 は既に先読みを行なっている．次に TR_1 より大きな時刻印を持った TR_2 が参照したオブジェクトに，その後 TR_1 が更新要求を行なうと，優先度の関係から TR_1 はアボートすることになる．するとその影響で，先読みを行なった TR_3 もアボートすることになってしまう．つまり中間優先度の TR_2 が，最高優先度の TR_3 のアボートを誘発したことになる．

これに対応するために，優先度の継承を行なう．つまり先読みによって確定していない最新バージョンを参照したトランザクションは，そのバージョンを作成したトランザクションに，自分の優先度の方が高いならばその優先度を継承させる．これにより高い優先度を持つトランザクションが，低い優先度のトランザクションのアボートに巻き込まれて失敗することは回避できる．

4 実験結果

図 1 は PTO で処理を行なった場合の他の方式との性能比較である．この図は，各方式で 1000 個のトランザクションを 100000 単位時間の間に実行したときのコミット率を示している．対象オブジェクト数はいずれも 20 であり，単位時間を基準とした平均トランザクション長と，各トランザクションごとのオブジェクトに対する平均アクセス数を変化させることにより，競合の発生確率を操作している．

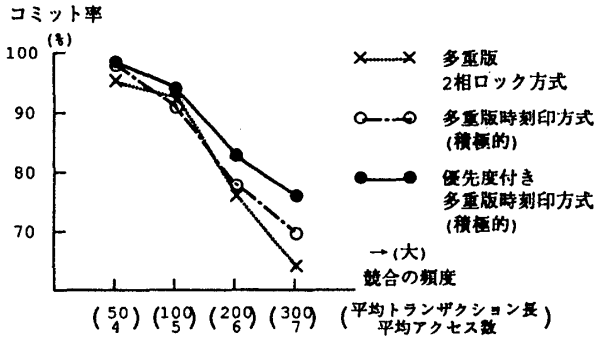

図 1: 各方式によるコミット率の比較

この図より PTO は競合が多く起こるような場合特に有効であると考えられる．また表 1は，PTO による優先度別のコミット率の変化を表している．これにより優先度が高いトランザクションは他のものに比べ，より確実にコミットしていることがわかる．

表 1: 優先度ごとのコミット率 (%)

	(低) ← 優先度 → (高)					
優先度	1	2	3	4	5	平均
コミット率	80.7	83.0	86.6	92.1	94.4	87.4

5 おわりに

本稿では，優先度と時刻印による新しい並行処理制御方式である PTO について述べ，他の各方式との性能比較を行なった．その結果，PTO はロックを用いた方式に比べ時刻印方式の長所を損なうことなく，従来の時刻印方式よりも全体的に高いコミット率を実現することが判明した，特に，優先度の高いトランザクションはより高いコミット率を得ることが可能である．

今後は，トランザクションのリアルタイムスケジューリングを含めた並行処理制御，特に時刻印方式のリアルタイム処理への適合性について検討して行く予定である．

参考文献

[1] 國枝和雄, 畑田孝幸, 大久保英嗣, 津田孝夫: 適応型時刻印方式に基づく同時実行制御方式, 情報処理学会論文誌, Vol. 33, No. 6, pp. 802-811 (1992).