分散システム運用技術　1－5
（1996．5．24）

分散データベースにおける不変時刻印方式のスループット特性の評価

石森　宣行　小林　真也

金沢大学工学部

〒920　金沢市小立野 2-40-20

分散データベースマネージングシステム (*DDBMS*) の同時実行制御の 1 つである不変時刻印方式は従来の時刻印方式と同じように同一データへの操作衝突を局所的に解決できるという特徴を持つだけでなく，要求順サービスが可能なので公平性も高い．不変時刻印方式ではコミットメント制御に仮コミットと本コミットの 2 つのフェーズを設けており，本コミット許可の決定には，システム内で処理中のトランザクションの時刻印の最小値であるグローバル処理最小時刻印 (*GTA*) を用いている．本研究ではトークンの巡回による分散的なコミットメント制御を用いた場合のスループット特性の評価を行う．

Evaluation of throughput of Permanent Timestamp Method of Concurrency Control

Noriyuki ishimori and Sin-ya Kobayashi

Faculty of Engineering, Kanawaza University

Permanent timestamp method of concurrency control is one of concurrency control mechanism for a distributed database system. It can guarantee that transactions can be processed in order of arrival. According to this method, the transaction is temporally committed when it has been processed, and it is truly committed when its timestamp becomes less than the least timestamp of processing transactions. In this paper we describe a method how a permanent timestamp method determines the least time stamp, and evaluate a throughput of this method.

1 はじめに

分散データベースの同時実行制御は，データベースの一貫性を保証したうえで，システムの性能および信頼性の向上，応答時間の均一化が要求される．分散データベースに対する代表的な同時実行制御方式である時刻印方式は，同一データへの操作衝突を局所的に解決できるという特徴を有している．しかしながら従来の時刻印方式では，動作衝突時に常に先に実行をはじめた処理が後退復帰されるため，ユーザからの要求順にサービスを行うことができず公平性が損なわれている．また，処理時間の長いトランザクションは処理中に他のトランザクションが到着する確率が高いので，繰り返しアボートされコミットできないというロックアウト状態が発生する可能性がある．

一方，先に実行を開始した処理を優先する同時実行制御方式として不変時刻印方式 [1] が提案されている．本稿では，トークンを巡回させコミットメント制御を行う方式のスループット特性の評価を行う．

2 不変時刻印方式の基本動作

不変時刻印方式は，多版時刻印方式に基づいた方式であり競合操作間に時刻矛盾が発生した際に大きい時刻印を持つトランザクションを後退復帰させる方式であり，処理の公平性が高く，ロックアウトの発生しない同時実行制御方式である．また，デッドロックが発生しないという時刻印方式本来の特徴も兼ね備えている．

不変時刻印方式ではデータ項目 x において書き込みが行われる毎に新版を作成する．以下では

TS_i：トランザクション T_i の時刻印
$TR_k(x)$：データ項目 x の版 k の読み出し時刻印
$TW_k(x)$：データ項目 x の版 k の書き込み時刻印

とする．ただし，トランザクション T_i への時刻印の付与は，受け付けノードが持つローカルな時計を用いて行い，その値は全システムノードで唯一の値とする．これは時刻印を，トランザクション受け付け時の実時刻とノード識別番号の組合せにすることによって可能である．また版 k には，版 k の値の読み出しを行った操作の履歴が全て保存されている (図 1)．

以下に不変時刻印方式の操作の詳細について述べる．

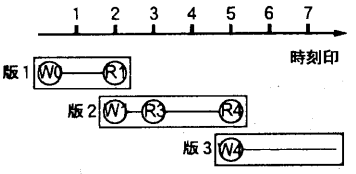

図 1: 版の管理

2.1 読み出し操作

不変時刻印方式の読み出し操作は，多版時刻印方式と同様に読み出し操作はアボートされること無く実行可能である．

具体的には，トランザクション T_i による読み出し操作 $R_i(x)$ が要求された場合には，$TS_i > TW_k(x)$ を満たす最大の書き込み時刻印を持つ版 k を選択し，読み出しを実行し，$TR_k(x)$ を $TR_k(x)$ と TS_i の大きい方に更新する．また，処理の後退復帰を実行する為に，各版には読み出し操作を行ったトランザクションが投入されたノード (親ノード)，時刻印の値をデータの履歴に記録しておく．読み出し操作実行前後のデータの履歴を図 2，図 3に示す．

2.2 書き込み操作

トランザクション T_i による書き込み操作 $W_i(x)$ は以下の手順で行う．

1) $TS_i > TW_k(x)$ を満たす最大の書き込み時刻印をもつ版 k を選択する．

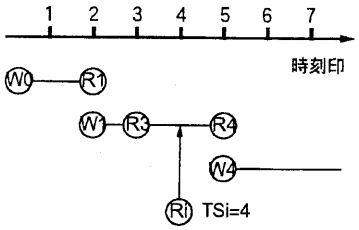

図 2: 読み出し実行前のデータの履歴

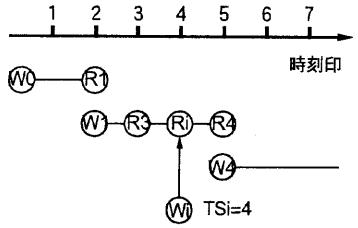

図 4: 書き込み実行前のデータの履歴

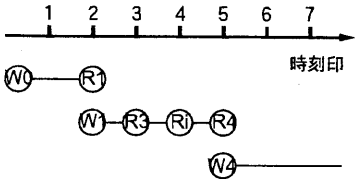

図 3: 読み出し実行後のデータの履歴

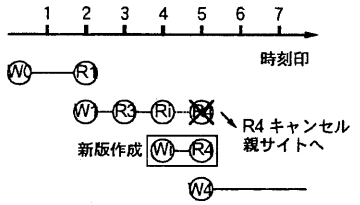

図 5: 書き込み実行後のデータの履歴

2) a) $TS_i \geqq TR_k(x)$ ならば無矛盾であり，新版 jを作成し $W_j(x)$ を TS_iとする．

b) $TS_i < TR_k(x)$ ならば時刻矛盾が発生する．まず，版 kの処理履歴から TS_iより大きな時刻印をもつ実行済読み出し操作をキャンセルする．次に新版 jを作成し，$TW_j(x)$ を TS_iとする．さらにキャンセルされた読み出し操作を再実行し，$W_i(x)$ が書き込んだ値を読み出し，結果を親ノードに返す．

図 4，図 5に矛盾が発生した場合の書き込み操作実行前後のデータの履歴を示す．

これらの図では，時刻印 4 をもつ書き込み操作 $W_i(x)$ と時刻印 5 を持つ実行済読み出し操作 $R_4(x)$ との間に時刻矛盾が発生している．このため，上記の手順に従って $R_4(x)$ をキャンセルし，$W_i(x)$ による新版作成後，$R_4(x)$ を再実行している．$R_4(x)$ の再読み出しの結果は親ノードに返される．

このように，不変時刻印方式は書き込み操作が矛盾を引き起こす場合に，トランザクションがアボートされることが無く，読み出し操作のキャンセル・再処理によって矛盾を解消することができる (すなわち，常に書き込み可である)．

2.3 後退復帰

矛盾発生により読み出し操作をキャンセルされたトランザクションの親ノードでは，トランザクションの処理系列をキャンセルされた読み出し操作まで後退復帰する．その後，再処理を行った結果，実行済書き込み済操作の値が変更された場合には，書き込み操作のキャンセル要求を出し再書き込みを行う．

トランザクション T_iによって実行済書き込み操作 $W_i(x)$ の再書き込みが要求されると，データ項目 x では処理履歴から $W_i(x)$ が作成した版を選択し，この版の履歴に実行済読み出し操作があればこれをキャンセルし，$W_i(x)$ の書き込み実行後，再読み出しを行う．

このように，再書き込み要求は新たに読み出し操作のキャンセルを伴う可能性があり，後退復帰の連鎖が起こりうる．しかしながらこの連鎖は有限であ

る．後退復帰の連鎖の様子を図6に示す．

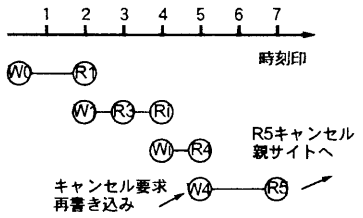

図 6: 後退復帰の連鎖

2.4 コミットメント制御

全節までに述べて来たように不変時刻印方式ではトランザクションの実行終了後も後退復帰される可能性がある．従って，トランザクション終了後直ちに結果をユーザに返すことはできない．ユーザに結果を返すことができるのは，そのトランザクションが後退復帰されないことが保証された後である．

そこで，トランザクションのコミットメント制御に仮コミットと本コミットという2つのフェーズを設ける．仮コミット状態とは，トランザクション終了後，後退復帰の可能性がある為，後退復帰されないことが保証されるのを待っている状態である．本コミット状態とは，トランザクションの後退復帰の可能性が無くなり，ユーザに結果を出力できる状態である，

矛盾が発生するのは，あるトランザクションが読み出し操作実行後，それよりも小さい時刻印をもつ書き込み操作が要求された時である．従って，実行中(アクティブ)のトランザクションの時刻印の最小値よりも小さい時刻印を持つトランザクションは，後退復帰されることは無い．また，読み出しステップのみからなるトランザクションは，他のトランザクションの後退復帰を引き起こすことが無い．

これらのことを考慮して，ローカル処理最小時刻印 LTA (*Local mininum Timestamp of Active transaction*)，グローバル処理最小時刻印 GTA(*Grobal mininum Timestamp of Active transaction*) を次のように定義する．

LTA_n : ノード n で処理中の書き込みステップを含む(或は含む可能性のある)トランザクションの時刻印の最小値

GTA : 全ノードの LTA の最小値

GTA は全システムで処理中の書き込みステップを含む(或は含む可能性のある)トランザクションの時刻印の最小値となる．すなわち，GTA より小さい時刻印を持つトランザクションは後退復帰される可能性が無いことが保証される．

各ノード n はトランザクション T_iを終了すると仮コミットし，LTA_nを次のように更新する．

$LTA_n = min$ (TS_k| T_kは書き込みステップを含む実行中のトランザクション)

また，仮コミット済のトランザクション T_iが後退復帰されると

$$LTA_n = min(LTA_n, TS_i)$$

とした後，T_iを再処理する．

各ノードは何らかの方法で GTA の値を決定し，GTA より小さい時刻印を持つトランザクションを本コミットし，結果をユーザに返す．

図7の例では，GTA は9であり，各ノードにおいて時刻印が9以前である操作は本コミット済となっている．また，時刻印が GTA と LTA の間である操作は仮コミット済となっている．例えば，ノード1では，時刻印7の読み出し操作と時刻印9の書き込み操作は本コミット済であり，LTA が13であるため時刻印11の読み出し操作と時刻印13の書き込み操作は仮コミット済となっている．

3 GTA の決定法

仮コミット済トランザクションを本コミットする際には，そのトランザクションが後退復帰されないことが保証されなければならない．前節では，GTA

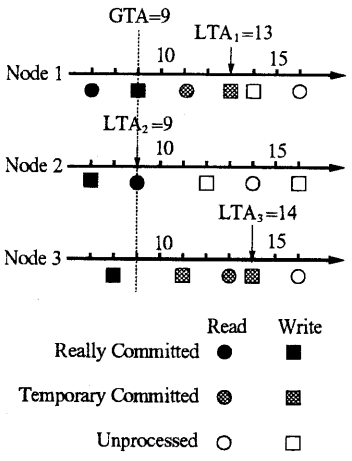

図 7: コミットメント制御

より小さい時刻印を持つトランザクションが，後退復帰されないことが保証されることを述べた．本節では、GTA を決定する手法について述べる。

GTA 決定法は集中管理と分散管理に大別できるが、今回は分散管理について考察する。

3.1 トークンを用いた分散管理

各ノードの LTA の値を書き込んだトークンを全てのノード間で巡回させれば，トークンの 1 巡毎に各ノードは全てのノードの LTA のを知ることができる．

しかし，トークンには全ノードの LTA の値を記録し巡回させるだけでは不十分である．これはトークン巡回の時間差によって誤った GTA を検出するノードが発生する可能性があるためである．ここで

GTA_n : ノード n がトークンから算出した GTA 決定値　($\leqq GTA$)

とする．

例えば，図 8に示すようにノード A, B, Cからなる DDBMS において，$LTA_A = 8$, $LTA_B = 13$, $LTA_C = 11$ のとき，ノード A からデータ $b(\in B)$ への書き込み要求 $W_i(b)$ が到着したとする．この時ノード Cを起源とする読み出し操作 $R_j(b)$ が既処理であるとする，この結果データ b において矛盾が発生し，$R_j(b)$ がキャンセルされ仮コミット済みトランザクション T_jは後退復帰されアクティブとなる．$W_i(b)$ の完了後 Ackが親ノード A に返されると $LTA_A = 12$, $LTA_C = 9$ となるが，更新した LTA の値をトークンに反映させるまでの時間差が問題となり，図 9に示すような状況では次にトークンを受け取るノード Bは $GTA_B = 11$ と誤って認識してしまう．

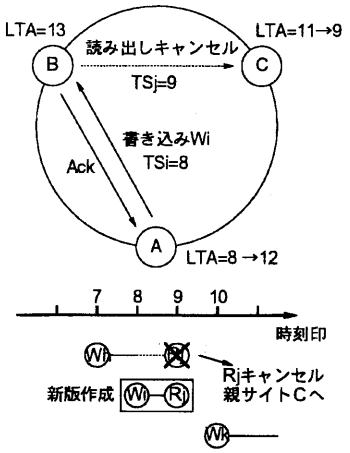

図 8: キャンセルの発生例

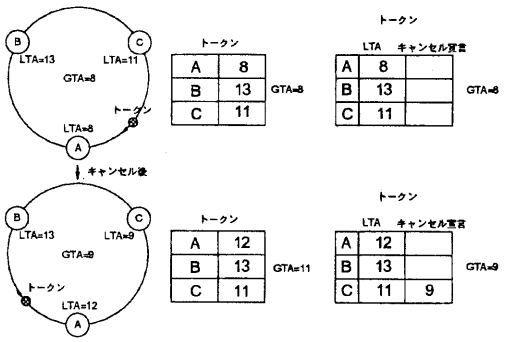

図 9: 誤ったＧＴＡの決定と正しい決定

そこで，トークンを LTA 部とキャンセル宣言部からなるものとし，キャンセル発生ノード (B) はキャンセルを誘引した書き込み操作が完了したことを知らせる Ack にキャンセルさせた読み出し操作の親ノード (C) と時刻印 (9) の情報を載せる．

この Ackを受け取ったノード A は，トークンが到着すると自ノードの LTA 部を 12 に更新し，ノード Cのキャンセル宣言部に 9 を登録する (図 9)．

全ノードの LTA 部とキャンセル宣言部を持つ GTA 決定トークンを巡回させる場合の，各ノード n の GTA_nの決定手順は次のようになる．

(1) 各ノード n はトークンを受け取ると，自ノードの LTA 部を更新し，キャンセルを誘発した場合は．該当するノードのキャンセル宣言部に登録する，
(2) 読み出し操作のキャンセル・後退復帰を受けたトランザクションが自ノードに存在すれば，トークンのキャンセル宣言部から該当する時刻印を削除する．
(3) トークンに記録された全ノードの LTA とキャンセル宣言の最小値を算出し GTA_nとする．

4 評価

本研究では試験システムを構築し，エミュレーションにより不変時刻印方式の性能評価を行う．試験システムは以下の様なものである．

- ノード数は 5
- データ項目は各ノードにつき 3 個
- トランザクションの到着間隔は全ノード均一で平均 $1/\lambda[sec]$ の指数分布に従う
- トランザクション長は 6
- 各トランザクションは複数のデータ項目に対して読み出しを行った後，1 個のデータ項目に対して書き込みを行う

評価結果を図 10に示す．これらの図において，x 軸はシステム負荷であり，y軸はスループットである．

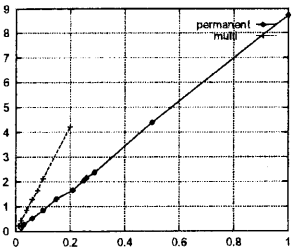

図 10: スループット特性 (トランザクション長= 6)

これらの結果より，高負荷になるにつれてスループットも増加しているのが分かる．

多版時刻印方式が負荷が 0.2 以上はロックアウトが発生して結果を返さないのに対して，不変時刻印方式では高負荷でも応答を返した。

また，低負荷時には多版時刻印方式の方がスループット特性は上だが，高負荷時には不変時刻印方式が多版時刻印方式を上回っていた．

5 まとめ

今回は，不変時刻印方式のスループット特性を評価した．比較対象の多版時刻印方式は低負荷時には良い結果を返したが，高負荷時には結果を返さなかったのに対し，不変時刻印方式は高負荷においても反応を返しただけではなく，スループットも良い値を返した．

参考文献

[1] 小林真也, 古川　誠, 中西　暉, 手塚慶一:“要求順サービス可能な時刻印方式について”, 電子情報通信学会論文誌 $(D), Vol.J74-D-I, No.3, pp.232-239(1991)$