データベースシステム 104−37
（１９９５．７．２０）

不変時刻印方式における集中型コミットメント制御の提案と検討

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．本研究では管理ノードによる集中的な*GTA*決定法の提案と，管理ノードに要求される処理能力に対する検討を行う．

Centrally Commitment Mechanism 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 propose a method how a central control node determines the least time stamp, and investigate how much performance is added of the central control node.

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．従来，不変時刻印方式のコミットメント制御は，各ノード間でトークンを巡回して行っていた．本稿では，コミットメント制御に管理ノードを用いる方式の提案と評価を行う．

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．

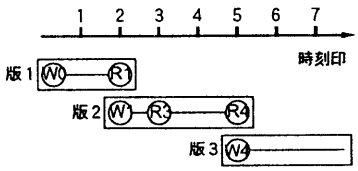

図1．版の管理

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．

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．

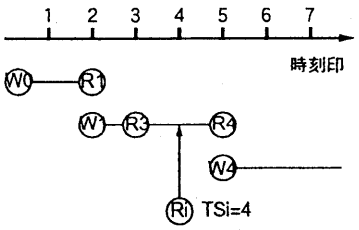

図2．読み出し実行前のデータの履歴

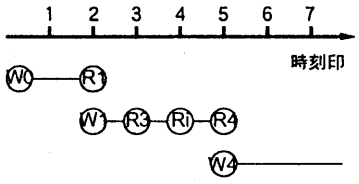

図3．読み出し実行後のデータの履歴

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．

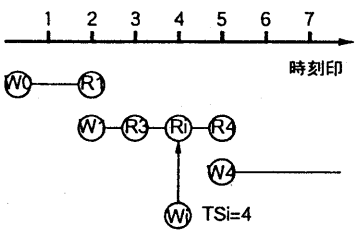

図4．書き込み実行前のデータの履歴

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．

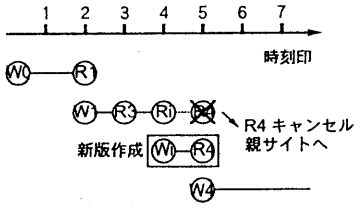

図5．書き込み実行後のデータの履歴

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．

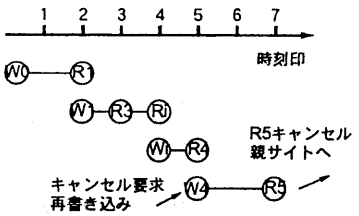

図6．後退復帰の連鎖

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．

$LTA_n = min\,(TS_k \mid T_k$ 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．グローバル処理最小時刻印の決定法

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 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

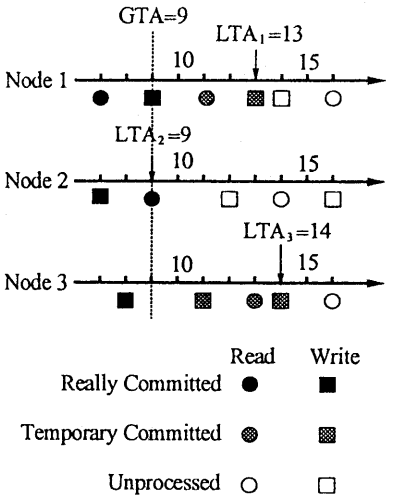

図7．コミットメント制御

ないことが保証されることを述べた．本節では，本稿で新たに提案する GTA の決定を管理ノードで集中的に行う手法に付いて述べる．

3．1 集中管理

中央に管理ノードを設置し，各ノードが LTA を更新する度に中央の管理ノードに知らせる．しかし，更新した LTA の値を知らせるだけでは不十分である．例えば，仮コミット済のトランザクションがキャンセルされた時には LTA の値が減少する．この時この変更を知らせるのが遅れれば実際の GTA よりも大きな GTA を算出してしまいキャンセルの可能性のあるトランザクションまでコミットされてしまう．

従って，キャンセル発生時には管理ノードに LTA だけではなく，キャンセルさせるトランザクションの時刻印も同時に送る必要がある．そこで，キャンセル発生時には，キャンセル発生ノードはキャンセルを誘引した書き込み操作の親ノードに対する書き込みの Ack に，キャンセルされた読み出し操作の親ノード名と，その操作の時刻印を付与して送る．トランザクションの読み出しや書き込みが終了したノードは，そのトランザクションを仮コミットし新たな LTA を決定する．

また，更新した LTA の値を，管理用プロセッサに通知するが，仮コミットしたトランザクションが他のノードのトランザクションのキャンセルを誘引した場合には，書き込みの Ack に付与されているキャンセルされたトランザクションの親ノードと時刻印も同時に送る．

一方，管理ノードは各ノードの LTA を記録しておく LTA 管理テーブルを持ち，各ノードからの LTA 更新通知メッセージによりテーブルの更新を行う．また，LTA 更新通知メッセージにキャンセルされたトランザクションに関する情報も含まれる場合には，キャンセルされたトランザクションの親ノードの LTA をキャンセルされたトランザクションの時刻印に変更する．

管理ノードは LTA 管理テーブルにより LTA の最小値である GTA を決定する．

集中管理では，正確な GTA の値を知ることができ，実装が容易であると言った利点もあるが，システムが大規模になると管理ノードへの通信量や処理量が大きくなり，高速化の障害となると考えられる．従って，集中管理を採用する際には管理ノードに高速な計算機を使い，できるだけ通信量を減らすといったことが必要となる．

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2か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

・管理ノードと他のノードの処理速度比を変化させる

評価結果を図１０，１１に示す．これらの図において，x 軸は平均到着時間間隔をトランザクションの投入から仮コミットまでの時間で割ったものであり，利用率の逆数に相当する．従って１に近い程高負荷である．また y軸は，トランザクションの投入から本コミットまでの時間をトランザクション投入から仮コミットまでの時間で割ったものである．これは，不変時刻印方式で必要なオーバヘッドである仮コミットから本コミットまでの時間の相対比を示すものであり，この値が大きい程応答時間に占めるオーバヘッドの割合が大きいことを意味する．

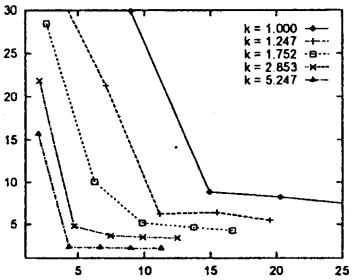

図１０．平均特性（トランザクション長＝２）

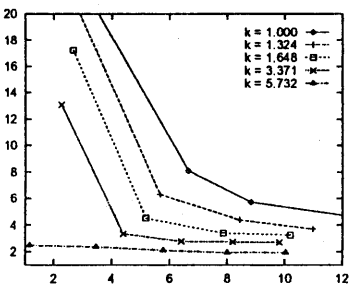

図１１．平均特性（トランザクション長＝６）

図１０はトランザクション長が２であるトランザクションの，図１１はトランザクション長が６であるトランザクションについての結果である．また図中の $k = 1, k = 1.247, k = 1.752, k = 2.853, k = 5.247$ は他のノードに対する管理ノードの速度比 kである．

これらの結果より，$k = 1$ つまり，管理ノードと他のノードの処理速度が等しい場合には，高負荷時には著しいオーバヘッドの増加がみられる．それに対し，k の増加に従って改善が見られた．

$k = 5.752$ の時には高負荷であっても，トランザクション長６のトランザクションでは，ほとんどオーバヘッドの増加が見られなかった．トランザクション長２のトランザクションでは，1/(利用率) $\doteqdot 2$ でオーバヘッドの増加が見られるが，それ以外ではオーバヘッドの増加は見られなかった．

５．まとめ

不変時刻印方式のコミットメント制御に必要な GTA を決定する方式として管理ノードにより集中的に行う方式の提案と，管理ノードに要求される処理能力に対する評価検討を行った．

今回の結果から，ノード数が５台程度では，管理ノードを他ノードの　５倍程度の処理速度にしてやればほぼ安定的に動作することが分かった．

しかしながら，ノード数が増加すると当然のことながら，管理ノードに対しより速い処理能力が要求される．今後これらのことについての検討を更に行う予定である．

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）, $Vol.J74-D-I, No.3, pp.232-239(1991)$