分散システムにおける複製データの相互一貫性制御方式

6E－1

冨田 達也　吉田 隆一
九州工業大学 情報工学部

1 はじめに

データ多重化は，分散システムの目標の一つである障害透過性を実現するための一手段であるが，複製データ間の相互一貫性を保証するためにトランザクション間の同期が必要となる．現在までに提案されている相互一貫性制御方式は，通常の同期制御に加え，障害回復時に相互一貫性を保つための回復処理を必要とした．また，メッセージ通信量が多いことやデッドロックの発生などの問題があった．そこで，本稿では新しい相互一貫性制御方式を提案する．提案する方式は，基本的にはすべての複製データに対して，時刻印順でトランザクションの要求を処理することで，データ間の相互一貫性を保証する．そして，通常の同期制御が障害回復処理をサポートするため，障害回復時に付加的な処理を必要としない．これは，通常から複数のデータを読み出し，その中から正しいデータを選択することで実現される．また，メッセージ通信量も少なく，時刻印を利用するためデッドロックの問題も解決することができる．

2 システムモデル

本稿では，分散データベースシステムを考える．データを管理するノードには，トランザクション間の同期をとるためにスケジューラが用意され，トランザクションはデータの読み出し，および書き込み要求をこれに対して行う．

3 基本方針

まず，複製データ間の相互一貫性制御を考える．本研究では，基本的時刻印方式 [2] に基づいて相互一貫性制御を行なう．さらに，応答時間を短縮するために，以下に述べる新しい read-write semantics を利用する．あるトランザクションの読み出し集合を R，書き込み集合を W と表す時，$x \notin R \cap W$ であるデータ x に対する読み出し要求と，$x \in R \cap W$ であるデータ x に対する読み出し要求を区別する．前者を R_R，後者を R_U，また書き込み要求を W と表現すると，相互一貫性を保証するために時刻印順に処理されなければならない要求の対は表1のようになる．（′ 記号のある方が時刻印が小さいとする）

	R_R	R_U	W
R'_R	-	-	matter
R'_U	matter	-	matter
W'	matter	matter	matter

表 1: 実行順序に制約のある要求対

次に，障害回復処理について考える．現在，提案されている手法は，他ノードの複製データをコピーすることで障害回復を行う．この手法は単純であるように思えるが，データはコピーされている間も変更がなされるため，より複雑になる．また，コピーのオーバヘッドも大きい．本研究では，障害時に付加的な処理を必要としない相互一貫性制御方式を目標として，通常の同期制御における読み出しを複数のデータに対して行い，その中から正しいデータを選択することを考える．このために，quorum consensus 方式を利用する [1]．具体的には，二つの条件 $(1) w + r > N$, $(2) w > N/2$ を満足するような読み出しデータ数 r と書き込みデータ数 w を決定する．この条件により読み出しデータの中に常に正しいデータが含まれることが保証される．

4 制御方式の提案

すべてのトランザクションは，生成時にユニークな時刻印が割り当てられ，スケジューラは，各データに対して，処理された読み出し要求，書き込み要求の最大時刻印 R-MAX, W-MAX を管理する．また，データレコードには書き込みを行ったトランザクションの時刻印 W-TIME を格納するフィールドが設けられる．

4.1 トランザクションの実行

phase 1

すべての複製に対して読み出し要求を行い，r 個のデータの読み出しに成功すれば，その中で最大の W-TIME を

A mutual cosistency control method for replicated data on distributed systems

Tatsuya TOMITA, Takaichi YOSHIDA, Kyushu Institute of Technology

持つデータを読み出し値とする．この方式で，トランザクションは時刻印順にデータの更新を行うため，W-TIME が最大であるデータが正しいデータであることが理解される．また，r 個のデータの読み出しに成功しなければ，アボートする．

phase 2

phase 1 で得られた読み出し値をもとに書き込みデータ値を計算する．

phase 3

すべての複製に対して書き込み要求を行い，w 個以上のデータの書き込みに成功すれば，COMMIT メッセージを，また，不成功であれば，ABORT メッセージを複製を管理するすべてのスケジューラに対して送信する．後者の場合，メッセージ送信後，自身をアボートする．

上記のトランザクションの実行は，論理データが唯一しか存在しない場合について述べたものであり，複数の論理データに対して読み出し，書き込みを行う場合は，各論理データに対して r 個の読み出し，w 個の書き込みに成功しなければ，アボートする．

4.2 同期操作

スケジューラは次の規則に従って，時刻印 T のトランザクションの読み出し要求，書き込み要求を処理する．ここで，トランザクションの cascading rollback を防止するために，あるトランザクションの書き込み要求を処理した後，そのトランザクションが COMMIT されるまで，データに対してロックを行う．

R_R の処理

1. T<W-MAX ならば，トランザクションに FAIL メッセージを返す．
2. T>W-MAX ならば，
   (a) データに対してロックが行われていないならば，要求を処理し，読み出しデータをトランザクションに返す．また，R-MAX は，max (T,R-MAX) に更新される．
   (b) データに対してロックが行われているならば，要求をキューに入れる．

R_U の処理

1. T<max (R-MAX,W-MAX) ならば，トランザクションに FAIL メッセージを返す．
2. 1. 以外の場合は，R_R の 2. と同じ規則に従って処理される

W の処理

1. T<max (R-MAX,W-MAX) ならば，トランザクションに FAIL メッセージを返す．
2. T>max (R-MAX,W-MAX) ならば，
   (a) データに対してロックが行われていなければ，ロックを行うと同時にデータの更新を行い，SUCCESS メッセージをトランザクションに返す．また，W-MAX は，T に更新される．
   (b) データに対してロックが行われているならば，要求をキューに入れる．

スケジューラは，COMMIT メッセージ，または ABORT メッセージを受信した場合，ロックを解除する．次の要求は，キューの中で最小の時刻印を持つものが選択され処理される．

5 まとめ

本稿では，障害回復時に付加的な処理を必要としない複製データの相互一貫性制御方式を提案した．この方式は，すべての複製に対して，基本的にトランザクションを時刻印順に処理することによってデータの相互一貫性を保証することができ，デッドロックも防止することができる．また，SDD-1 で用いられている手法と比較してメッセージ通信量も大幅に削減することができる．本稿では基本的アルゴリズムの提案にとどまったが，今後，メッセージ通信量の減少，応答時間の短縮などについて，シミュレーションにより確認を行う必要がある．

参考文献

[1] Gifford D. K.: Weight voting for replicated data. *In Proceedings of the 7th Symposium on Operating Systems Principles*, 1979

[2] Philip A. Bernstein and Nathan Goodman: Concurrency Control in Distributed Database Systems, *Computing Surveys*, Vol.13, No.2, June 1981