分散データベース・システムにおける同時実行制御方式の性能評価†

加藤 宣弘††

同時実行制御は，分散データベース・システムにおいて重要な技術課題のひとつであり，多くの制御方式が提案されている．本稿では，分散データベース・システムにおける同時実行制御方式のシミュレーションによる相対的な性能比較について述べる．性能を比較する方式は，3種類の2相ロック方式，2種類の時刻印順方式，1種類の楽観制御方式，1種類の先読みスケジューラである．特に，トランザクションを構成するデータ入出力操作の組合せとサイト間通信に要する時間が性能に与える影響について調べる．評価の結果，実験したすべての組合せにおいて先読みスケジューラが優れていることがわかった．また，先読みスケジューラを除くと，データを読んで書き換える操作が多い場合には2相ロック方式が，データを書き込む操作が多い場合には時刻印順方式が他の方式より優れていることがわかった．さらに，通信時間の変化に対して，楽観制御方式は影響を受けやすいことがわかった．

1. はじめに

最近，分散データベース・システム（分散 DBS）が注目を集めている．分散 DBS における重要な課題のひとつが同時実行制御である．同時実行制御に課された問題は，直列化可能性を保証しつつ最大のスループットを達成するようなトランザクションの実行スケジュールを生成することである．しかし，このような直列化可能性の判定はNP完全であり，上記の問題の最適解を多項式時間内に求めることは不可能である[1]．それゆえ，同時実行制御に関する様々な方式が提案されている．

これまでに提案されている同時実行制御方式は，次の4つに大きく分類できる．

①2相ロック方式[2]
②時刻印順方式[2]
③楽観制御方式[3]
④先読みスケジューラ[4),5]

これらの方式が生成する実行スケジュールが直列化可能であることは証明されているが，性能面からの有効性については様々な議論がある．特に，分散 DBS では考慮すべきパラメータが多いので性能予測が難しい．

分散 DBS における同時実行制御方式の性能を評価するアプローチとしては，解析的手法[6),7]とシミュレーション[8)~12]が代表的である．解析的手法では，モデル化の限界により厳密な性能評価を行うことができない．一方，シミュレーションでは，詳細なモデル化を行うことができる．したがって，本研究では後者のアプローチを選択する．

シミュレーションによる性能評価では，その結果がモデルやパラメータに大きく依存するので，同じモデルの上で相対的な比較を行うことが重要である．そこで本研究では，上記の同時実行制御方式のいずれかに分類される様々なアルゴリズムを同じ分散 DBS モデル上に実装し，相対的な性能比較を行う．

トランザクションを構成するデータベース操作の組合せによりアルゴリズムの優劣が変化すると予想されるので，本稿ではいくつかの組合せのそれぞれについて各アルゴリズムの性能を評価する．また，サイト間通信は分散システム特有の機構であり，集中システムでは見られない影響を同時実行制御に与えると予想されるので，通信に要する時間が各アルゴリズムの性能に与える影響についても調べる．

本稿では，2章で性能評価を行う同時実行制御方式について説明する．そして，3章でシミュレーションの手法を示し，4章で評価の結果と考察を述べる．

2. 同時実行制御方式

2.1 基本的事項

データベースの観点からはトランザクションを任意のデータベース操作の系列と見なすことができる．それぞれのデータベース操作は，データベースの値を読み書きするために，それを格納しているデータ領域にアクセスする．このデータ領域をデータ項目と呼ぶ．例えばページやレコードなどがデータ項目に対応す

† Performance Evaluation of Concurrency Control Algorithms in the Distributed Database System by NOBUHIRO KATO (Parallel and Distributed Computer Department, Information Systems Engineering Laboratory, TOSHIBA Corporation).

†† (株)東芝情報処理・機器技術研究所開発第1部

る．同時実行制御は，同じデータ項目に同時にアクセスする複数のデータベース操作の実行を制御し，トランザクションの直列化可能性を保証する．

本稿ではデータベース操作として次の3種類を考える．

①Read 操作：1つのデータ項目から値を読む操作．

②Write 操作：1つのデータ項目に値を書く操作．

③Rewrite 操作：1つのデータ項目から値を読み（Read フェーズと呼ぶ），値を書き換えて同じデータ項目に書く（Write フェーズと呼ぶ）操作．

これらのデータベース操作を総称して操作と呼び，1つのトランザクションを構成する操作の組合せをアクセス・パターンと呼ぶ．

トランザクションをコミットすることにより，その効果をデータベースにすべて反映させる．また，トランザクションをアボートすることにより，その効果をデータベースに全く反映させないようにする．同時実行制御の理由によりアボートしたトランザクションはある時間の後に再実行する．

2.2 同時実行制御方式

評価する7つのアルゴリズムについて説明する．

（1） A2PL (Aggressive Two-phase Locking)[2]

これは2相ロック方式に分類される．このアルゴリズムでは，Read, Write 操作および Rewrite 操作の Read フェーズを実行する直前に，アクセスするデータ項目にそれぞれ共有，排他，排他ロックをかけることを試みる（Rewrite 操作の Write フェーズはそのまま実行する）．ロックをかけることができれば操作を実行し，そうでなければ操作の実行を待たせる．ロックは，トランザクションのコミットあるいはアボート完了時にはずす．デッドロックは Wait-Die 方式により防止する．ただし，Wait-Die 方式とは，ロック待ち操作の時刻印がロックをかけている操作の時刻印より大きいとき，待ち操作の実行を拒否し，そのトランザクションをアボートするデッドロック防止方式である．

（2） C2PL (Conservative Two-phase Locking)[2]

これは2相ロック方式に分類される．このアルゴリズムでは，トランザクションの開始時に，そのトランザクションを構成する Read, Write, Rewrite 操作がアクセスするすべてのデータ項目にそれぞれ共有，排他，排他ロックをかけることを試みる．ロックをすべてかけることができれば，操作を次々に実行し，トランザクションのコミット完了時にロックをはずす．1つでもロックをかけることができなければ，ロックをはずし，トランザクションをアボートする．デッドロックは起こらない．

（3） 2V2PL (Two Version Two-phase Locking)[2]

これは2相ロック方式に分類される．このアルゴリズムでは，Read, Write 操作および Rewrite 操作の Read フェーズを実行する直前に，アクセスするデータ項目にそれぞれ R, W, R ロックという共有ロックをかけることを試みる．ただし，1つのデータ項目に対して複数の R ロックと1つの W ロックをかけることはできるが，複数の W ロックをかけることはできない．Rewrite 操作の Write フェーズを実行する直前には R ロックを W ロックに変換することを試みる．また，トランザクションをコミットする直前には，値を書いたデータ項目の W ロックを排他ロックに変換することを試みる．ロックをかける（変換する）ことができれば操作を実行し（処理を進め），そうでなければ操作を待たせる（処理を停止する）．ロックは，トランザクションのコミットあるいはアボート完了時にはずす．デッドロックは A2PL と同じように Wait-Die 方式により防止する．

（4） BTO (Basic Timestamp Ordering)[2]

これは時刻印順方式に分類される．このアルゴリズムでは，トランザクションの開始時に時刻印を付与する．Read 操作の実行時には，その時刻印 TS とデータ項目の Write 時刻印 WTS を比較する．TS>WTS ならば実行を認め，そうでなければ拒否する．Write 操作の実行時には，その時刻印 TS とデータ項目の Read 時刻印 RTS, Write 時刻印 WTS を比較する．TS>WTS または WTS>TS>RTS ならば実行を認め，そうでなければ拒否する．Rewrite 操作の Read フェーズは Read 操作，Write フェーズは Write 操作と同じように制御する．拒否された操作を含むトランザクションはアボートする．データ項目の Read, Write 時刻印は，それぞれデータ項目の値を読んだ操作，データ項目に値を書いた操作の時刻印の最大値である．

（5） MVTO (Multiversion Timestamp Ordering)[2]

これは時刻順方式に分類される．このアルゴリズムでは，Write 操作を実行する度にデータ項目の新しいバージョンを作り，その時刻印をバージョンの Write

時刻印とする．Write 操作の実行時にはその時刻印を超えない最大の Write 時刻印を持つバージョンを見つける．Read 操作の場合は，そのバージョンを Read する (Read 操作を拒否することはない)．Write 操作の場合は，その時刻印がバージョンの Read 時刻印より大きければ実行を認め，そうでなければ拒否する．Rewrite 操作の Read フェーズは Read 操作，Write フェーズは Write 操作と同じように制御する．拒否された操作を含むトランザクションはアボートする．ここで，Read 時刻印とは，そのバージョンを Read した操作の最大時刻印である．

(6) TBC (Timestamp Based Certification)[3]

これは楽観制御方式に分類される．このアルゴリズムでは，操作の実行時には何の制約も課さず，トランザクションのコミット直前に操作の承認を行う．Read 操作，Rewrite 操作の Read フェーズの実行時には，データ項目の値とともにそのバージョン識別子を Read する．操作の承認を行う直前にトランザクションに時刻印を与える．Read 操作は，データ項目のバージョン識別子に変更がなくかつそれより小さな時刻印を持つ Write 操作を承認していないとき承認される．Write 操作は，それより大きな時刻印を持つ Read 操作を承認していないとき承認される．Rewrite 操作は，上記の Read, Write 操作の承認手続きを両方満たすとき承認される．すべての操作が承認されたトランザクションはコミットし，そうでないトランザクションはアボートする．

(7) CS (Cautious Scheduler)[4),5]

これは先読みスケジューラに分類される．このアルゴリズムでは，データ項目を介した操作間の干渉を表す有向グラフを管理している．トランザクションの開始時に，それを構成する操作とそれぞれがアクセスするデータ項目を用いて有向グラフを変更する．さらに，操作の実行時には，有向グラフを変更してその時点で実行できるかを判定する．ある時点で実行を待たされた操作は，有限時間内に必ず実行できることが保証されているので，トランザクションをアボートする必要はない．トランザクションに付与された時刻印を用いて，各サイトにおける分散トランザクションの実行順序の一貫性を保証する．

3. シミュレーションの手法

3.1 分散 DBS モデル

シミュレーションには，ネットワークで接続された複数の計算機サイトからなる分散 DBS モデルを用いる．各サイトの DBS モデルは，それぞれ端末，TM (トランザクション・マネージャ)，SC (スケジューラ)，DM (データ・マネージャ)，CM (コミュニケーション・マネージャ) の5つのサーバから構成される．

端末はトランザクション処理を TM に要求し，TM からの応答を待つ．応答を受けると，ある思考時間の後に新しいトランザクション処理を要求する．TM は処理要求を受けたトランザクションを Read 操作，Write 操作，Rewrite 操作の Read フェーズ，Write フェーズに分解し，それぞれアクセスするデータ項目の存在するサイトの SC へその実行を要求する．また，TM はトランザクションのコミット，アボートを SC に要求する．アボートされたトランザクションは，TM によりある時間の後に再実行される．SC は操作の実行要求を受け，それぞれのアルゴリズムに従って操作を同時実行制御する．DM は SC からの要求を受けてディスク・メモリ間のデータベース操作を実行する．CM はサイト間通信を担う．異なるサイトの TM と SC はそれぞれのサイトの CM を介して要求，応答を送受する．

図1にあるトランザクションの処理の流れを示す．ただし，ホーム・サイトとはトランザクションを受け付けた TM が存在するサイトであり，リモート・サイトとはホーム・サイト以外でそのトランザクションの操作がアクセスするデータ項目の存在するサイトで

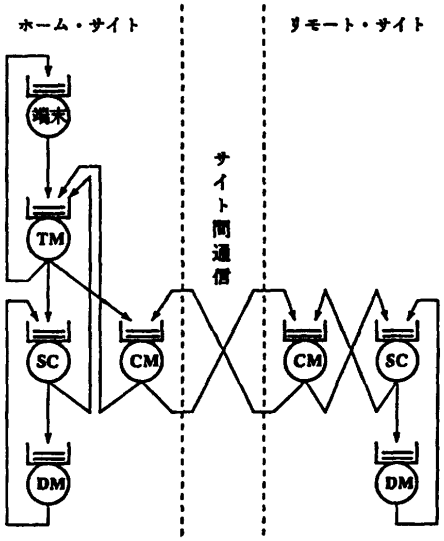

図1 あるトランザクションの処理の流れ
Fig. 1 Flow of a transaction processing.

ある．各サーバは FCFS (First Come First Served) に基づいてキューにある要求を処理する．なお，モデルは性能評価ツール eCceds[13] を用いて記述する．

3.2 シミュレーションの前提

シミュレーションは次の前提のもとに行う．これらの前提は，同時実行制御に関係しない部分を簡素化するだけでなく，制御方式による相対的な性能の差を顕著にするためのものである．

①各サイトの処理能力は等しい．
②双方向同時転送可能な回線でサイト同士を結ぶ．
③すべてのサイトの TM が端末からのトランザクション処理要求を受け付ける．
④トランザクションの各操作がどのサイトのどのデータ項目にアクセスするかは，トランザクションの受け付け時に決める．そのとき，各操作はある割合でリモート・サイトのデータ項目にアクセスするようにする．
⑤端末数により，同時実行するトランザクションの数を変化させる（データ競合を変化させる）．
⑥1つのトランザクションは，Read 操作，Rewrite 操作の Read フェーズ，Write フェーズ，Write 操作の順に直列に実行する．
⑦Read，コミット操作ではディスクにアクセスする．
⑧Write，アボート操作ではディスクにアクセスしない．
⑨1相コミット・プロトコルを用いる．
⑩サイト障害，通信障害は起きない．
⑪データの複製を持たない．

上記の②は，例えばネットワーク・トポロジが完全グラフであり，全2重通信方式を用いることを想定している．通信遅延（通信に要する時間）は評価結果に影響を与えるが，遅延の要因が何であるかは問題でないので，ネットワークの部分のモデル化は行わない．

③④より，各サイトの SC は自分のサイトの TM から要求された操作だけでなく，他のサイトの TM から要求された操作も同時実行制御する．他のサイトから要求された操作は制御されるまでに時間的遅れがあるために，特に時刻印順方式の性能に与える影響が大きいと予想される．

④の前半部分は，アルゴリズム C2PL, CS の性能に影響を与える．もしトランザクションの受け付け時に操作のアクセスするデータ項目が決まっていなければ，これらのアルゴリズムでは同時実行制御のためのデータ項目の粒度を大きくせざるを得ず，そのために性能低下を引き起こす可能性がある．

④の後半部分は，1つのトランザクションに対して1つのホーム・サイトと0個以上のリモート・サイトが対応することを示す．

⑥において直列とは，操作の確認応答があるまで次の操作の実行を要求しないことを言う．⑥に示すように操作を実行するのは，できるだけトランザクションの並列性を導き出そうという考えに基づく．

⑦⑧は，ディスク・メモリ間のデータベース操作に関する仮定である．Read 操作，Rewrite 操作の Read フェーズの実行時には，常にディスクにアクセスしメモリに値を読み出す．また，Write 操作，Rewrite 操作の Write フェーズにより書き込んだ値はメモリに保持する．保持した値は，コミット時にはディスクに書き込み，アボート時には消去する．

3.3 シミュレーション・パラメータ

表1にシミュレーション・パラメータ，**表2**にトランザクションのアクセス・パターンを示す．表2に示すアクセス・パターンのそれぞれについてシミュレーションを行う．

表1 シミュレーション・パラメータ
Table 1 Simulation parameters.

パラメータ	設定値
サイト数	16
サイトあたりの端末数	2～16
サイトあたりのデータ項目数	10
リモート・サイトのデータ項目にアクセスする操作の割合	50%
シミュレーション時間	20 sec
端末の思考時間	1 sec
トランザクション・アボート時の再実行待ち時間	1 sec
TM, SC のサービス時間	0.5 msec
ディスク・アクセス時の DM のサービス時間	20 msec
ディスク・アクセスしない時の DM のサービス時間	0.5 msec
CM のサービス時間	0～5 msec

表2 アクセス・パターン
Table 2 Access patterns.

アクセスパターン	Rewrite 操作数	Read 操作数	Write 操作数
pattern 1	3	0	0
pattern 2	1	3	1
pattern 3	1	1	3
pattern 4	0	4	2
pattern 5	0	2	4

4. 結果と考察

4.1 アクセス・パターンの与える影響

まず2相ロック方式に分類されるアルゴリズムについて，次に時刻印順方式に分類されるアルゴリズムについて，最後に2相ロック方式，時刻印順方式の中で最も優れた性能を示すアルゴリズム，および楽観制御方式，先読みスケジューラに分類されるアルゴリズムについて性能評価結果を示す．

（1） 2相ロック方式の性能評価

図2から**図6**までに，それぞれのアクセス・パターンにおける端末数に対するスループットの変化を示す．ただし，スループットとは単位時間あたりにコミットされるトランザクション数のサイト平均値である．また，端末数を増やすことにより，データ競合を激しくする．これらの図から次のことが言える．

①A2PL が最も高いスループットを示す．

②2V2PL は，Rewrite 操作のみの場合 C2PL より低いスループットを示すが，それ以外の場合 A2PL に次いで高いスループットを示す．

③Rewrite操作が少なくデータ競合が激しくない場合，2V2PL は A2PL と同じ位のスループットを示す．特に，Rewrite 操作がなく Read 操作が多い場合には，データ競合に関わらず，2V2PL は A2PL と同じ位のスループットを示す．

④データ競合が激しくなっても，C2PL は性能低下の幅が小さい．

⑤データ競合が激しくない場合，C2PL は低いスループットを示す．

2V2PL では，Rewrite 操作の Write フェーズの実行時にRロックをWロックに，トランザクションのコミット時にWロックを排他ロックに変換する必要がある．このようなロックの変換はデッドロックを起こす原因になるので，Write 操作，Rewrite 操作が多い場合にはトランザクションをアボートする割合が高くなる．したがって，Write 操作，Rewrite 操作が多い場合に 2V2PL のスループットが A2PL に比べ

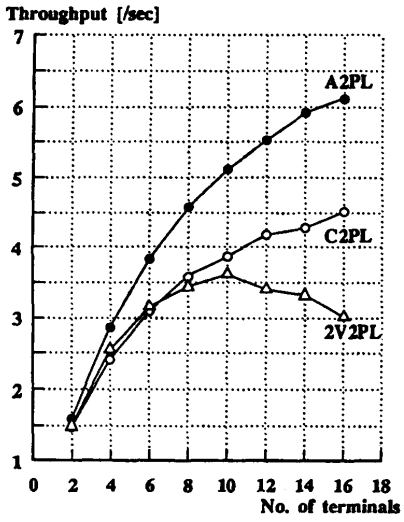

図 2 スループットの比較（2相ロック方式）
Fig. 2 Throughput, two-phase locking.
(Pattern 1, Communication time=2.5msec)

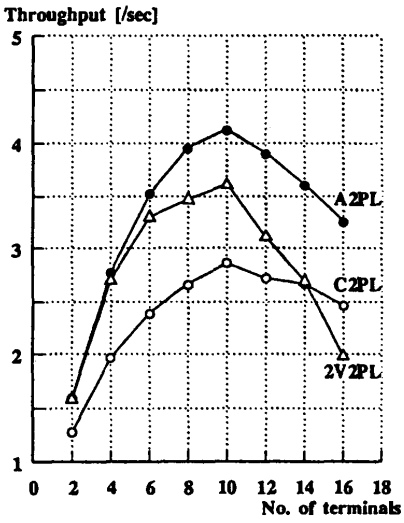

図 3 スループットの比較（2相ロック方式）
Fig. 3 Throughput, two-phase locking.
(Pattern 2, Communication time=2.5msec)

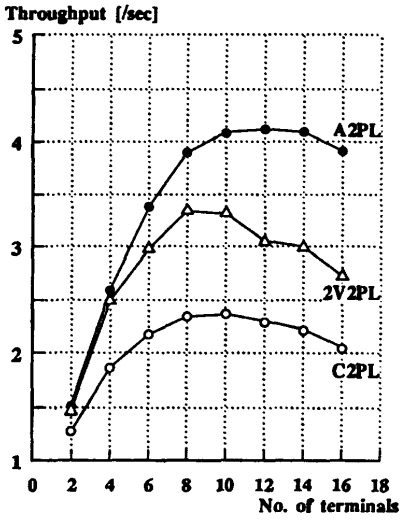

図 4 スループットの比較（2相ロック方式）
Fig. 4 Throughput, two-phase locking.
(Pattern 3, Communication time=2.5msec)

て低くなると考えられる．

上記の④は，C2PL が再開始主義 (Restart Policy) のアルゴリズムであることに起因する．また，C2PL ではあらかじめすべてのデータ項目にロックをかけるために，そのオーバヘッドにより上記の⑤のような結果となると考えられる．

（2） 時刻印順方式の性能評価

図 7 から**図 11** までに，それぞれのアクセス・パターンにおける端末数に対するスループットの変化を示す．これらの図から次のことが言える．

①Read 操作，Rewrite が多い場合より，Write 操作が多い場合のほうが，2つのアルゴリズムのスループットの差が顕著である．

②Rewrite 操作がある場合，MVTO が BTO より低いスループットを示すことがある．

MVTO は，Read 操作，Rewrite 操作の Read フェーズの実行を拒否しないという特長を持つ．それゆえ，Read 操作，Rewrite 操作が多い場合に MVTO は BTO より高いスループットを示すと予想されるが，評価結果は上記の①のようになる．このような結

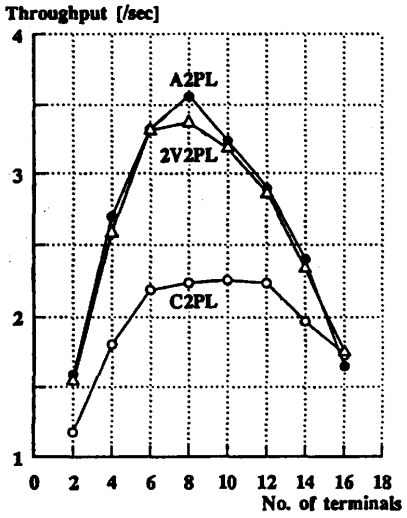

図 5 スループットの比較（2相ロック方式）
Fig. 5 Throughput, two-phase locking.
(Pattern 4, Communication time=2.5 msec)

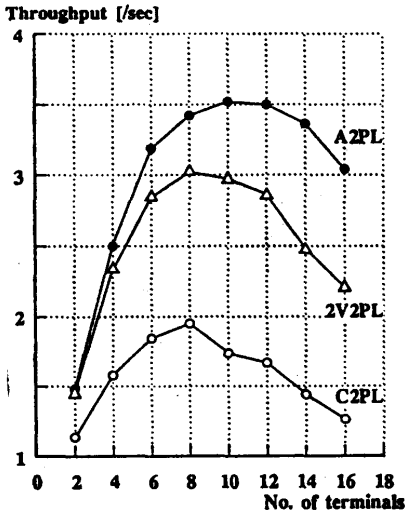

図 6 スループットの比較（2相ロック方式）
Fig. 6 Throughput, two-phase locking.
(Pattern 5, Communication time=2.5 msec)

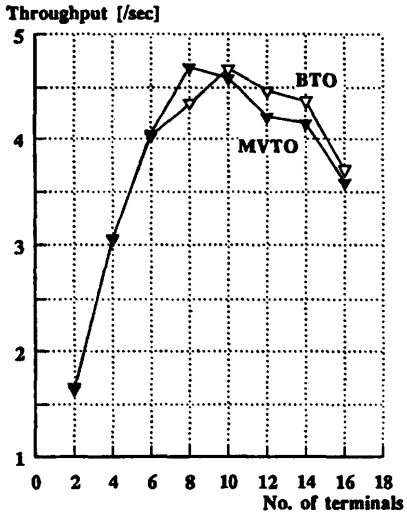

図 7 スループットの比較（時刻印順方式）
Fig. 7 Throughput, timestamp ordering.
(Pattern 1, Communication time=2.5 msec)

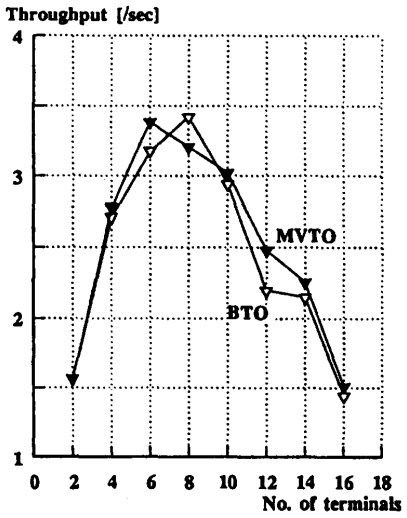

図 8 スループットの比較（時刻印順方式）
Fig. 8 Throughput, timestamp ordering.
(Pattern 2, Communication time=2.5 msec)

果になった理由は，次のように考えることができる．

Write 操作が多い場合，BTO ではデータ項目の Write 時刻印を小さな時間間隔で更新し，MVTO では新しいバージョンを小さな時間間隔で作成することになる．このとき BTO では少し遅れて要求された（少し小さな時刻印を持つ）Read 操作，Rewrite 操作の Read フェーズの実行を拒否する可能性が高くなる．しかし，MVTO ではそれらの実行を拒否しない．したがって，MVTO のほうがトランザクションをアボートする割合が小さくなり，より高いスループットを示すことになる．

一方，Read 操作が多い場合，BTO ではデータ項目の Write 時刻印を大きな時間間隔で更新し，MVTO では新しいバージョンを大きな時間間隔で作成する．このとき BTO では少し遅れて要求された（少し小さな時刻印を持つ）Read 操作，Rewrite 操作の Read フェーズの実行も認める可能性が高くなる．ゆえにトランザクションをアボートする割合は両者であまり差がなくなる．さらに，MVTO では Write 操作，Rewrite 操作の Write フェーズの実行だけを拒否するので，トランザクションのアボートが遅れ，無駄な処理を行うことになる．したがって，この場合には両者のスループットにあまり差が出ない．

上記の②は，データ競合のために拒否される Rewrite 操作を BTO では Read フェーズで拒否し，MVTO では Write フェーズで拒否することに起因する．つまり，MVTO のほうがトランザクションのアボートが遅れるからである．

（3） 同時実行制御方式の性能評価

図 12 から**図 16** までに，それぞれのアクセス・パターンにおける端末数に対するスループットの変化を示す．これらの図から次のことが言える．ただし，上記の結果から 2 相ロック方式としては A 2PL を，時刻印順方式としては MVTO を選択する．

①CS が最も高いスループットを示す．

②Rewrite 操作が多い場合，A 2PL, MVTO, TBC の順に高いスループットを示す．

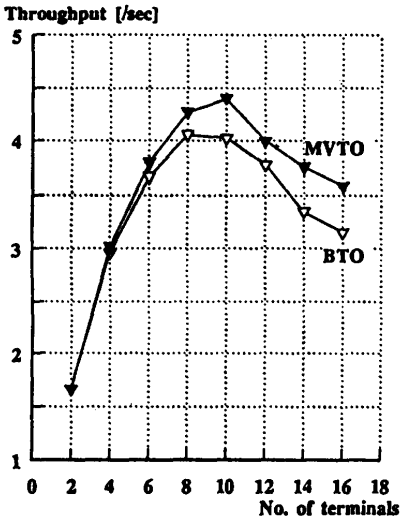

図 9 スループットの比較（時刻印順方式）
Fig. 9 Throughput, timestamp ordering.
(Pattern 3, Communication time=2.5 msec)

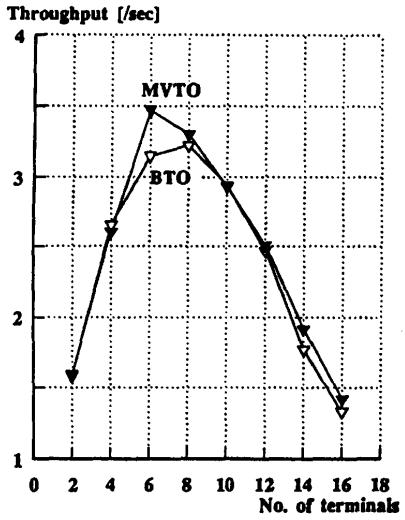

図 10 スループットの比較（時刻印順方式）
Fig. 10 Throughput, timestamp ordering.
(Pattern 4, Communication time=2.5 msec)

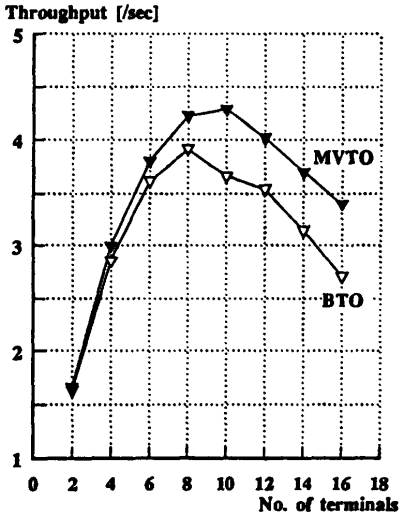

図 11 スループットの比較（時刻印順方式）
Fig. 11 Throughput, timestamp ordering.
(Pattern 5, Communication time=2.5 msec)

③Write 操作が多い場合，MVTO，TBC，A2PL の順に高いスループットを示す．

④Read 操作が多くデータ競合が激しい場合，MVTO は他の方式より低いスループットを示す．

⑤TBC はアクセス・パターンによる影響を受けにくい．

CS は他のアルゴリズムより柔軟なスケジューリングを行い，デッドロックやアボートを起こさないので，データ競合が激しい場合に優れた性能を示す．

Rewrite 操作が多い場合には Rewrite 操作によるデータ競合が多くなる．このとき，A2PL では一方の Read フェーズの実行を待たせるかあるいは拒否し，MVTO では一方の Write フェーズの実行を拒否し，TBC ではコミット実行を拒否する．つまり，TBC，MVTO，A2PL の順にトランザクションのアボートが遅れるので，②のような結果になると考えられる．

Write 操作が多い場合，上記のように MVTO では Read 操作，Rewrite 操作の Read フェーズの実

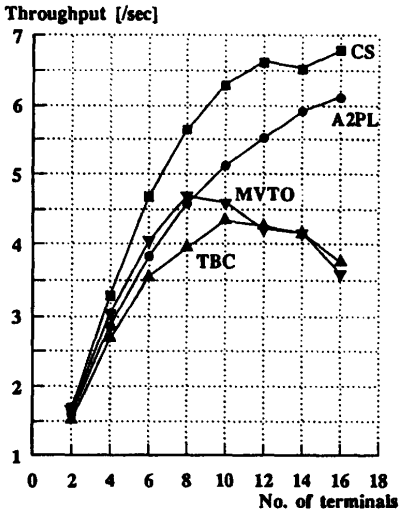

図 12 スループットの比較（同時実行制御方式）
Fig. 12 Throughput, concurrency control. (Pattern 1, Communication time=2.5 msec)

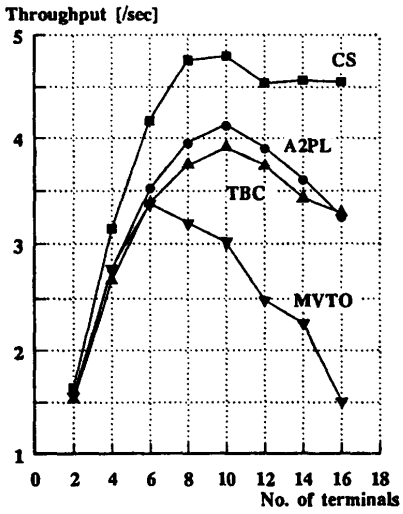

図 13 スループットの比較（同時実行制御方式）
Fig. 13 Throughput, concurrency control. (Pattern 2, Communication time=2.5 msec)

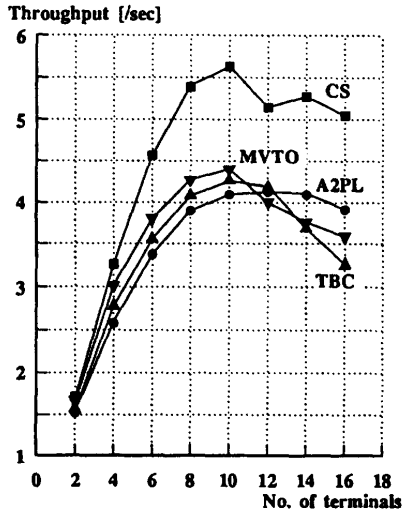

図 14 スループットの比較（同時実行制御方式）
Fig. 14 Throughput, concurrency control. (Pattern 3, Communication time=2.5 msec)

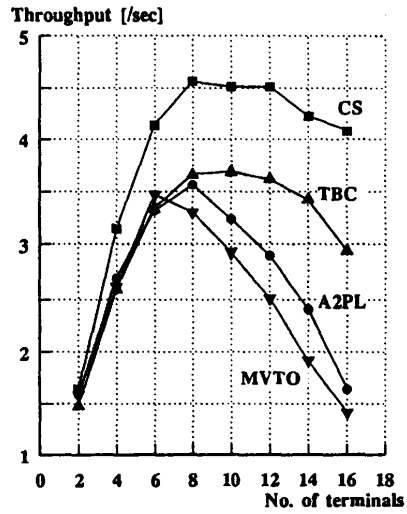

図 15 スループットの比較（同時実行制御方式）
Fig. 15 Throughput, concurrency control. (Pattern 4, Communication time=2.5 msec)

行を拒否しないという特長が生かされるので，③のようにMVTOが高いスループットを示す．A2PLが低いスループットを示すのは，Write操作が多いと排他ロックのかかっているデータ項目が多く，実行を拒否される操作が多くなるためと考えられる．

Read操作が多い場合，MVTOではその特長が生かされないだけでなく，データ競合が激しくなってもWrite操作，Rewrite操作のWriteフェーズの実行時まで操作を拒否しないため，トランザクションのアボートが遅れるので，④のような結果となる．

TBCではトランザクションに時刻印を与える時点と操作の同時実行制御を行う時点が近いので，操作を時刻印順に直列化しやすく，トランザクションをアボートする割合が低くなる．また，トランザクションをアボートする時点はすべての操作を実行した後であり，一定している．このような性質により⑤のような結果になると思われる．

図17から図20までにそれぞれ平均応答時間，1秒以内に応答するトランザクション数，DMの使用率，CMの使用率の端末数に対する変化を示す．これらの

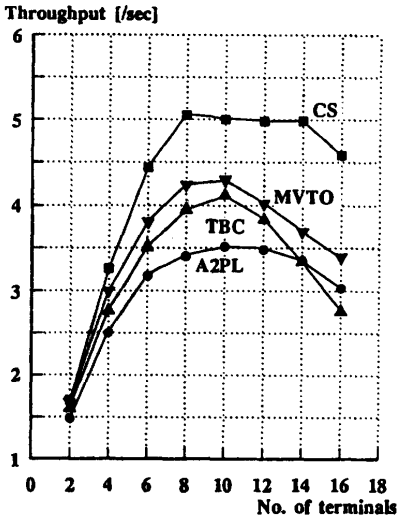

図16 スループットの比較（同時実行制御方式）
Fig. 16 Throughput, concurrency control.
(Pattern 5, Communication time=2.5 msec)

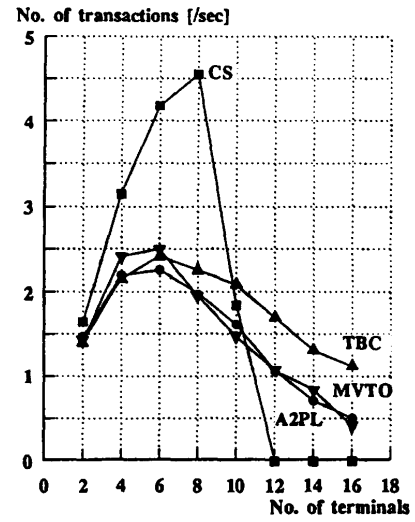

図18 1秒以内に応答するトランザクション数の比較
Fig. 18 No. of trans. responding within 1 sec.
(Pattern 2, Communication time=2.5 msec)

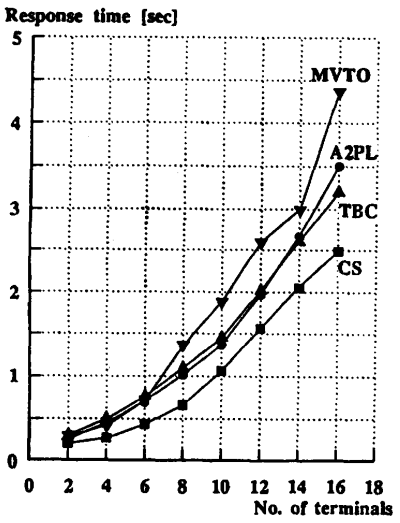

図17 平均応答時間の比較
Fig. 17 Average response time.
(Pattern 2, Communication time=2.5 msec)

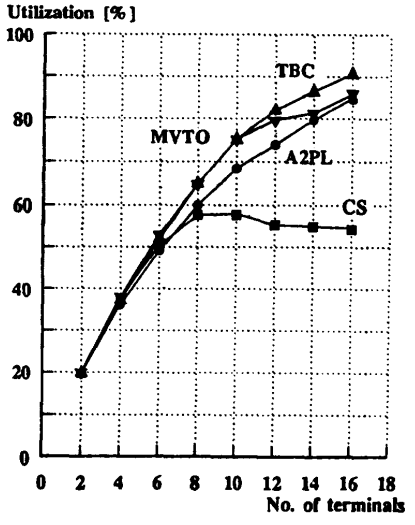

図19 DMの使用率の比較
Fig. 19 DM utilization.
(Pattern 2, Communication time=2.5 msec)

図より次のことが言える.

①CS では，平均応答時間が1秒を超えると，1秒以内に応答するトランザクション数が激減する.

②DM, CM の使用率は TBC が最も高い.

③CS では DM, CM の使用率が過負荷状態で一定になる.

上記の①は，CS においてトランザクションの応答時間の偏差が小さいことを表している．これは，CS ではトランザクションをアボートしないことに起因すると思われる．TBC はトランザクションのアボートが遅れるので，②のように資源を浪費してしまう．CS では過負荷状態になると一定以上のトランザクションの実行を停止するため，③のような結果になる．なお，ここでは pattern 2 の場合についてのみ述べたが，他のアクセス・パターンでも同じような結果が得られた.

4.2 通信時間の与える影響

図 21 から**図 23** までに，それぞれスループット比率の通信時間に対する変化を示す．ここでスループット比率とは，通信時間が0である場合のスループットに

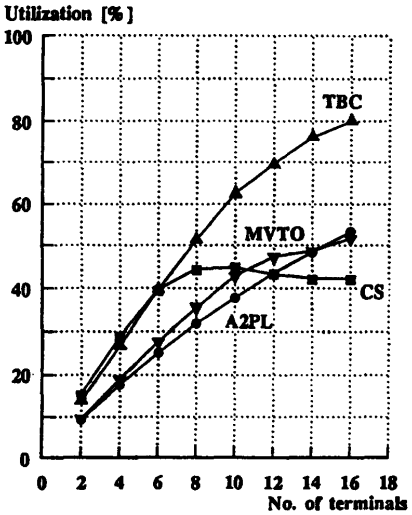

図 20 CM の使用率の比較
Fig. 20 CM utilization.
(Pattern 2, Communication time=2.5 msec)

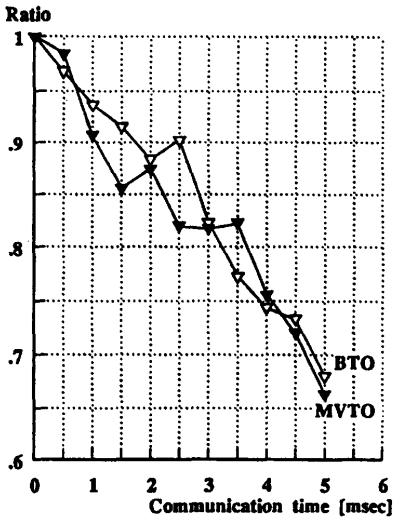

図 22 通信時間の影響（時刻印順方式）
Fig. 22 Effect of communication time.
(Pattern 2, No. of terminals=8)

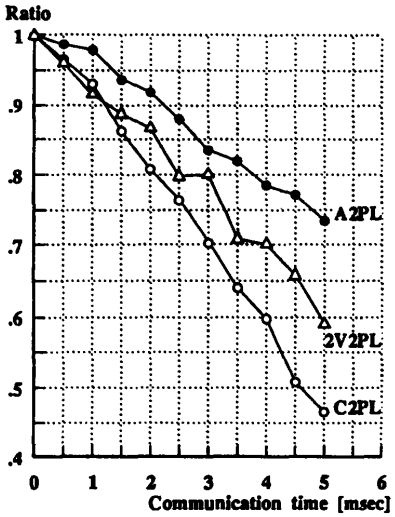

図 21 通信時間の影響（2相ロック方式）
Fig. 21 Effect of communication time.
(Pattern 2, No. of terminals=8)

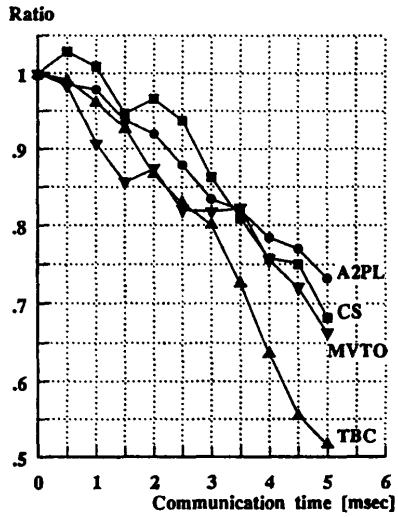

図 23 通信時間の影響（同時実行制御方式）
Fig. 23 Effect of communication time.
(Pattern 2, No. of terminals=8)

対する比率のことである．ここでは pattern 2，端末数が8である場合について示す．これらの図より次のことが言える．

①2相ロック方式の中では，C2PL が最も影響を受けやすく，A2PL が最も受けにくい．

②BTO と MVTO が受ける影響は同じ程度である．

③TBC は A2PL, MVTO, CS より影響を受けやすい．

アルゴリズム上の制約から，C2PL，2V2PL は A2PL に比べて通信回数が多くなる．そのため C2PL，2V2PL ではデータ項目にロックをかけている時間が A2PL より大きくなり，①のような結果となる．

TBC, CS もまた A2PL に比べて通信回数が多くなる．TBC では，通信時間が大きくなると，操作を実行してから承認するまでの時間が大きくなり，アボートされる割合が高くなる．CS では，通信時間の増加により実行を遅らす操作は増えるが，その操作を拒否しないのであまり影響を受けない．

5. おわりに

7種類の同時実行制御アルゴリズムについてシミュレーションよる性能評価を行った．アクセス・パターン，通信時間がトランザクション処理性能に与える影響を調べた結果，次のことが明らかになった．

①評価したすべてのアクセス・パターンにおいて，先読みスケジューラ (CS) が優れた性能を示す．

②Rewrite 操作が多い場合は2相ロック方式 (A2PL) が，Write 操作が多い場合は時刻印順方式 (MVTO) が相対的に優れた性能を示す．

③楽観制御方式 (TBC)，2相ロック方式 (C2PL, 2V2PL) は通信時間の影響を受けやすい．

本稿では，データの複製を持たない場合について分散 DBS における同時実行制御方式の性能評価を行ったが，今後の課題としてデータの複製を持つ場合について性能評価を行うことが挙げられる．また，サイト障害や通信障害が起こる場合について評価することも今後の課題である．

参考文献

1) Ibaraki, T., Kameda, T. and Minoura, T.: Serializability with Constraints, *ACM Trans. Database Syst.*, Vol. 12, No. 3, pp. 429-452 (1987).

2) Bernstein, P. A., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, pp. 1-166, Addison-Wesley (1987).

3) Sinha, M. K.: Timestamp Based Certification Schemes for Transactions in Distributed Database Systems, *Proc. ACM SIGMOD Conf.*, pp. 402-411 (1985).

4) 原島秀次，茨木俊秀：先読みスケジューラによる分散型データベースシステムの並行処理制御，信学論 D, Vol. J70-D, No. 6, pp. 1140-1148 (1987).

5) Katoh, N., Ibaraki, T. and Kameda, T.: Efficient Implementation of Cautious Schedulers, CSS/LCCR TR 90-13, Simon Fraser Univ. (1990).

6) Li, V. O. K.: Performance Models of Timestamp-Ordering Concurrency Control Algorithms in Distributed Databases, *IEEE Trans. Comput.*, Vol. C-36, No. 9, pp. 1042-1051 (1987).

7) Ozsu, M. T.: Modeling and Analysis of Distributed Database Concurrency Control Algorithms Using an Extended Petri Net Formalism, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 10, pp. 1225-1240 (1985).

8) Sevcik, K. C.: Comparison of Concurrency Control Methods Using Analytic Models, *Proc. IFIP Conf.*, pp. 847-858 (1983).

9) Carey, M. J. and Livny, M.: Distributed Concurrency Control Performance: A Study of Algorithms, Distribution, and Replication, *Proc. 14th VLDB Conf.*, pp. 13-25 (1988).

10) Garcia-Molina, H.: Performance Comparison of Two Update Algorithms for Distributed Database, *Proc. 3rd Berkeley Workshop on Dist. Data Mgmt. and Comp. Networks*, pp. 108-119 (1978).

11) Ries, D. R.: The Effects of Concurrency Control on the Performance of a Distributed Management System, *Proc. 4th Berkeley Workshop on Dist. Data Mgmt. and Comp. Networks*, pp. 75-112 (1979).

12) Thanos, C., Bertino, E. and Carlesi, C.: The Effects of Two-Phase Locking on the Performance of a Distributed Database Management System, *Performance Evaluation*, Vol. 8, pp. 129-157 (1988).

13) 柿元：シミュレーションによる性能評価用ツール eCceds，第41回情報処理学会全国大会論文集，6 D-3, No. 4, pp. 105-106 (1990).

(平成3年11月5日受付)
(平成4年9月10日採録)

加藤 宣弘（正会員）

昭和37年生．昭和61年京都大学工学部電気工学第二学科卒業．昭和63年同大学院工学研究科修士課程修了．同年(株)東芝入社．データベース管理システムの研究開発に従事．電子情報通信学会会員．